Sumário

1. HISTÓRIA DO Python:

Criador: Guido Van Rossum

Origem do nome Python: Inicialmente Guido V. Rossum quis homenagear um programa de humor americano "Monty Python's Flying Circus", porém, o nome foi associado a um tipo de cobra, por uma editora que inicialmente publicou os primeiros livros da linguagem e que sempre associava a capa dos livros a um animal.

Sistemas Operacionais: Amiga, NetBSD, MacOS, Linux (todas já possuem o Python na instalação). Logicamente no Windows também podemos usá-la, porém deve ser baixada e instalada, ou seja, entre os sistemas operacionais mais famosos a linguagem não vem pré-instalada. Assim sendo, a linguagem **Python é Multiplataforma**, pois podemos desenvolver softwares em diversos sistemas operacionais.

Uso de Python: Para desenvolver sistemas operacionais, IOT (Internet das Coisas) através do RaspBerryPi (lê-se "respberiPai"), inclusive o "Pi" é referência ao Python.

Outras possibilidades com Python

O MinecraftPi é um projeto que basicamente é uma biblioteca que se instala em um RaspberryPi, para poder ensinar programação Python para jovens, com possiblidade de criação até de jogos.

2. CARACTERÍSTICAS GERAIS DO PYTHON:

LINGUAGEM DE PROPÓSITO GERAL: Pode ser usada para programar o que você desejar (Por exemplo, o PHP é voltado especificamente para desenvolvimento para internet);

FÁCIL E INTUITIVO: Comandos simples, sintaxes fáceis;

MULTIPLATAFORMA: Programas escritos em Python funcionam basicamente em qualquer sistema operacional, em celulares e até em TV´s;

Batteries included: a linguagem já vem com a essência de bibliotecas e recursos instalados, porém podemos instalar outras quando necessário.

CÓDIGO ABERTO (LIVRE): Qualquer programador pode baixar o Python para estudar, programar e distribuir seus programas;

Extremamente ORGANIZADA: Todo programador deve ser organizado, obedecer a indentações de estruturas para programar. Com isso, todo programador, mesmo que não organizado acaba se tornando, pois é uma exigência para quem programa nesta linguagem

ORIENTADA A OBJETOS: A linguagem é orientada a objetos, tudo que se programa nesta linguagem é encarado como um objeto. **Até uma simples variável é um objeto**. Com o andamento dos estudos você verá como isso funciona.

REPLETO DE BIBLIOTECAS: Existem bibliotecas do Python, por exemplo, para:

- Programar para celular;
- Criação de jogos;
- Criar telas de sistemas;
- Etc...

3. PRINCIPAIS ÁREAS DE APLICAÇÕES DO PYTHON:

- Inteligência artificial
- Biotecnologia
- Computação 3D (computação gráfica)
- Ciência de Dados
- Microcontroladores

4. QUEM USA PYTHON?

Exemplos de algumas grandes empresas que usam o python:

- AIR CANADA – Reserva de assentos para voo são desenvolvidas em Python
- BitTorrent – Transferência de arquivos entre usuários sem que estes arquivos precisem estar num servidor. Exemplo: redes “peer to peer” (para a par, ou ponto a ponto);
- Rede Globo - Globo.com quando você pesquisa por vídeos.
- Google – Robôs que fazem buscas de dados quando procuramos algo no google, são conhecidos como ***CRAWLERS, também conhecidos como Spider ou bot (robôs).***
- YOUTUBE – também feito em Python
- Nasa
- Industrial Light & Magic (Criadora do Star Wars em suas produções de renderizações na computação gráfica)

5. FERRAMENTAS CRIADAS COM Python:

- Blander - produzir projetos em computação gráfica
- Ginp – funciona como photoshop
- RaspberryPi – Placa micro controladora
- Arduino – Não é linguagem padrão deste micro controladora, porém também é compatível com Python.

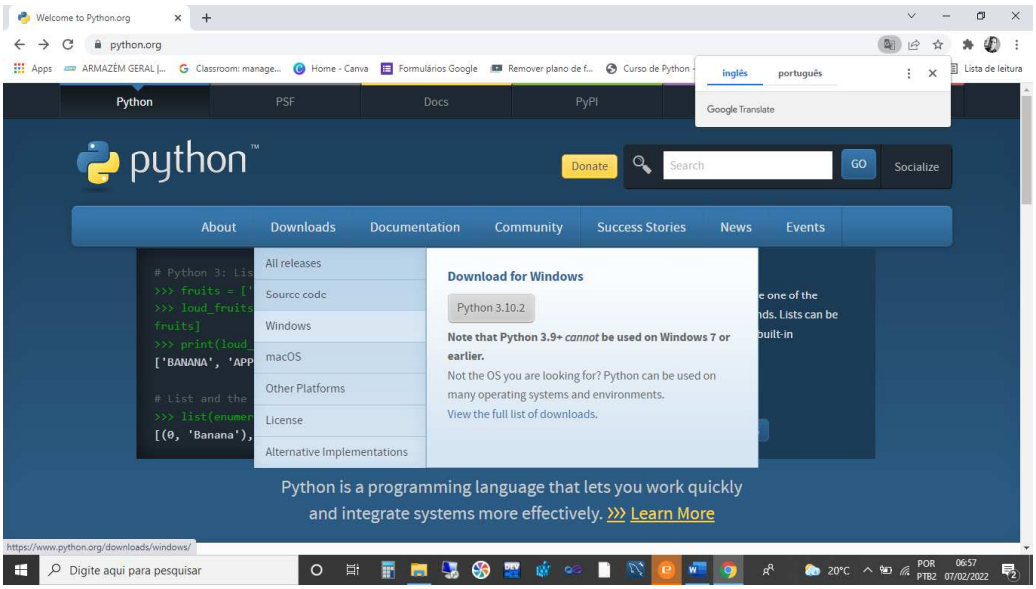
6. JOGOS CRIADOS EM PYTHON:

- EVE: Naves espaciais
- FRETS ON FIRE: Tocar guitarra com teclado
- JEWEL QUEST (Lê-se Jiuel Quest)
- Civilization IV
- BattleField 2 (lê-se barolfield 2)

7. DOWNLOAD E INSTALAÇÃO DO PYTHON e do IDLE

Para baixar o PYTHON, o IDLE (Ambiente de Desenvolvimento Integrado (IDE)) e suas bibliotecas, para o Windows basta entrar no site **python.org**. Na aba **Download,** você encontrará disponível sempre a última versão do software para baixar. Neste exemplo iremos baixar o Python para Windows. Você deve se atentar a qual versão do Windows você tem no seu computador. O Python é um software capaz de gerar programas para multiplataformas, assim sendo, você poderá baixá-lo para o sistema operacional que desejar. No exemplo abaixo, veja exemplo de como baixar e instalar uma versão.

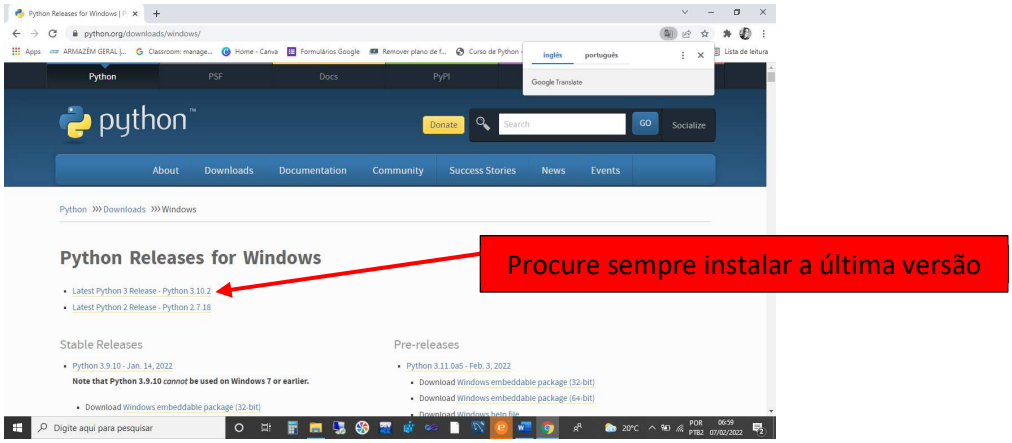
Siga os passos abaixo:



ATENÇÃO!!!! COMO ESTAMOS EXEMPLIFICANDO, AS VERSÕES DISPONÍVEIS NESTAS IMAGENS PODEM SER DIFERENTES. VOU BAIXAR E INSTALAR A VERSÃO 3.10.2

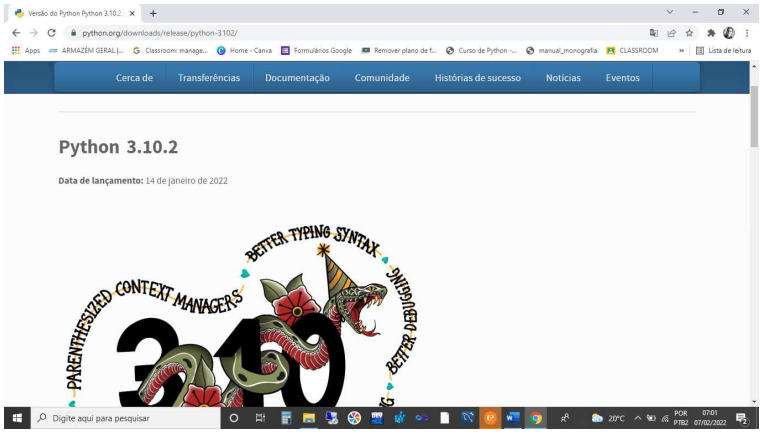
Na tela acima, clique em “Downloads” e, em seguida, “WINDOWS”. A partir da versão 3.9+ não funciona no Windows 7.

Surgirá então a seguinte tela:

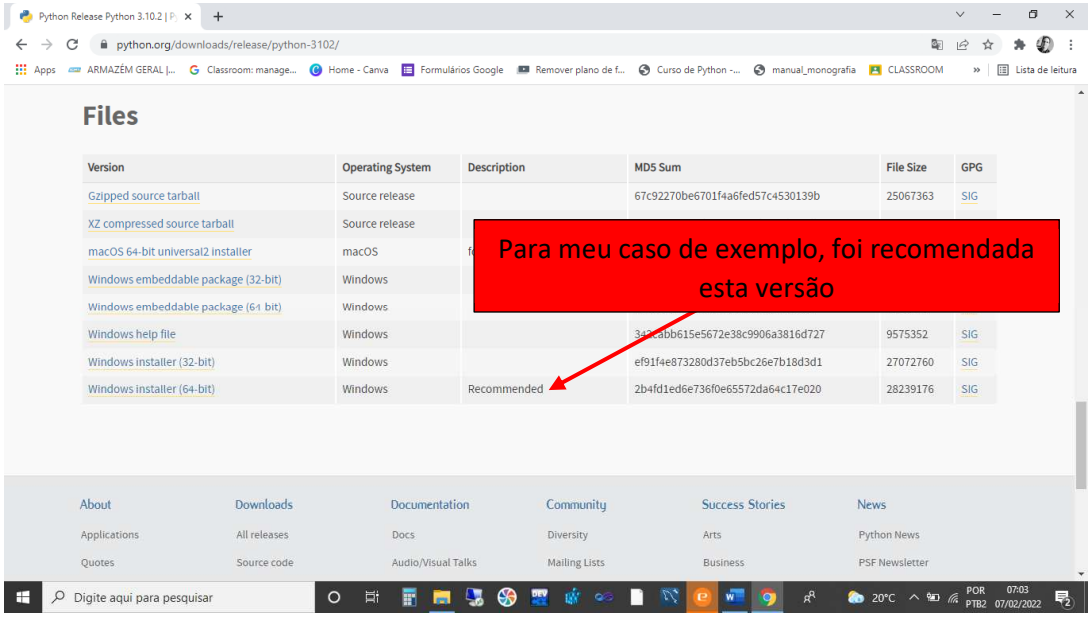


Clique em "LATEST PYTHON RELEASE – PYTHON 3.10.2" (OBSERVAÇÃO: ESTA VERSÃO É A ÚLTIMA QUE ESTAVA DISPONÍVEL NA OCASIÃO DE ESCRITA DESTA APOSTILA. ASSIM SENDO, VERSÕES MAIS ATUALIZADAS TENDEM A SURGIR NESTE LOCAL)

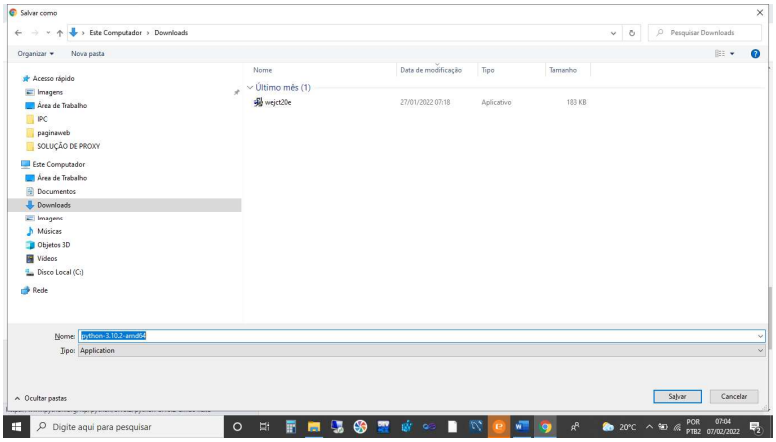
Surgirá então a seguinte página:



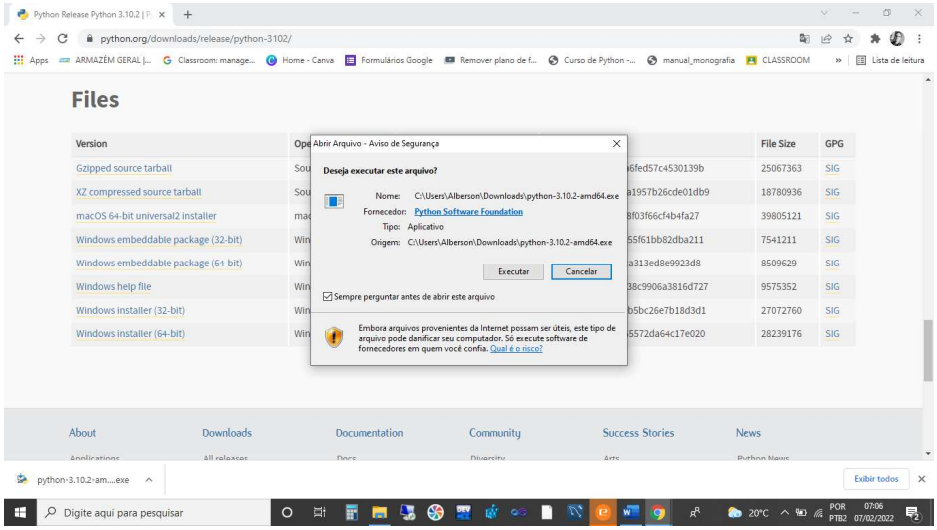
Para certificar-se da versão do Windows que você tem instalado e baixar a versão correta, vá ao final desta página acima e verifique a listagem das versões do python que estão disponíveis para instalação. Veja a seguir:



No meu caso vou clicar em "WINDOWS INSTALLER (64-BIT)", o que fará aparecer a seguinte tela:

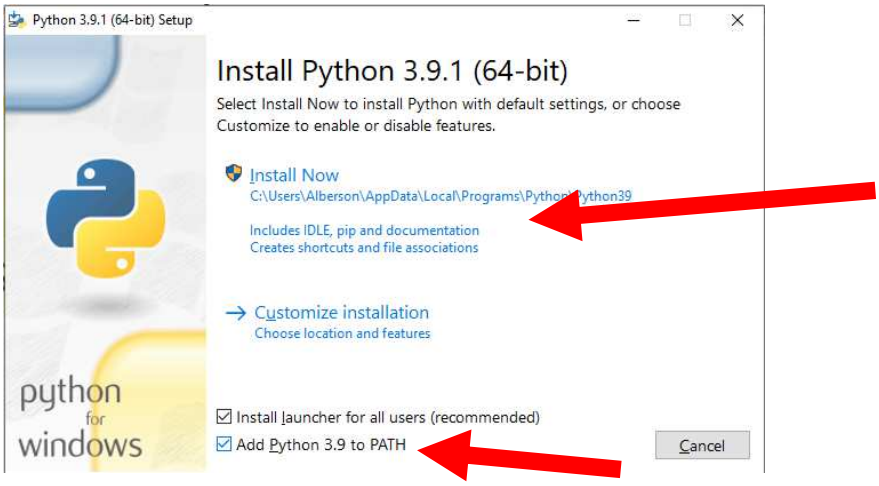


Escolha na tela mostrada, o local que deseja baixar o python e aguarde o final do download. Após baixado, clique no instalador. Surgirá então a seguinte tela:

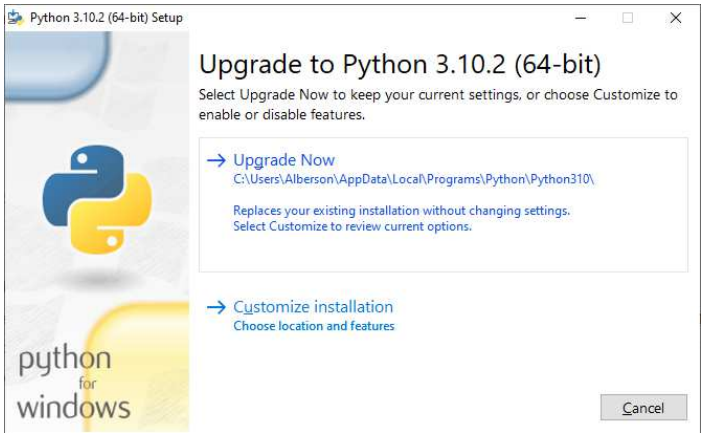


Agora basta clicar em “EXECUTAR” para iniciar a instalação

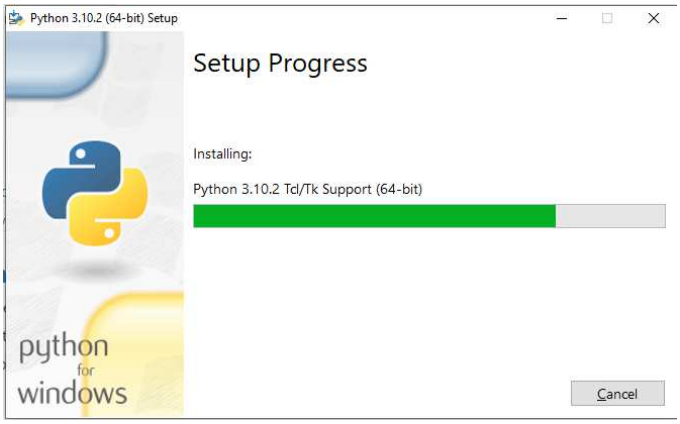
Quando instalei a primeira versão do python no meu computador marquei a opção “Add Python 3.9 to PATH” (conforme mostrado a seguir). Esta marcação permitirá você utilizar os comandos do PYTHON na tela de comando do Windows(“cmd”) no seu computador.



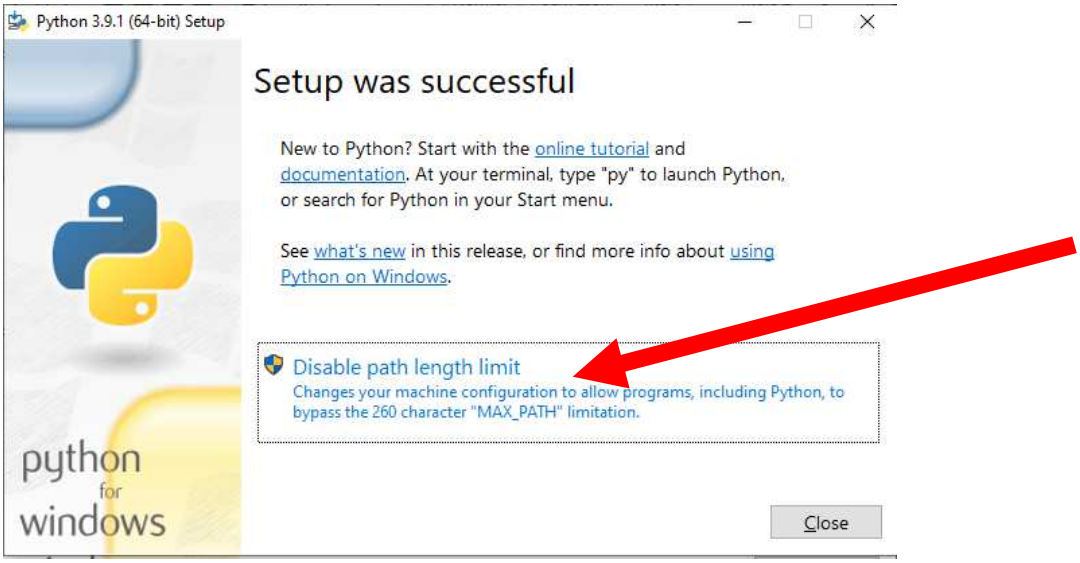
Como estou exemplificando a instalação numa versão atual, poderá aparecer a seguinte tela. Clique em “Upgrade Now”, observe abaixo, por exemplo:



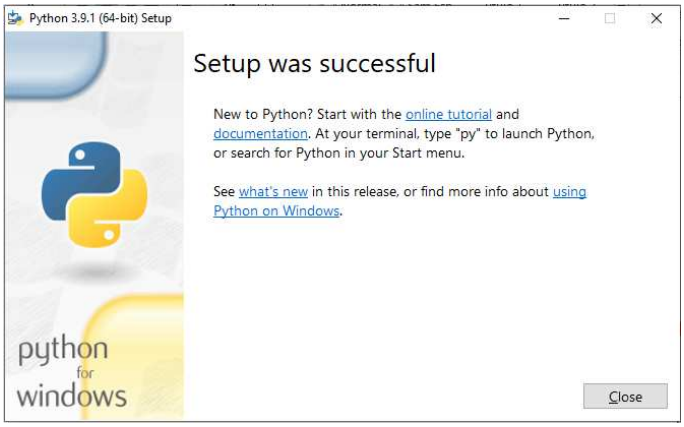
Logo em seguida, clique em “Install Now” que será iniciada a instalação, conforme a tela a seguir:



Quando fizer a instalação do python pela primeira vez, surgirá a tela abaixo. Recomenda-se desabilitar as limitações de comandos, clicando em “Disable path length limit”, na tela abaixo:



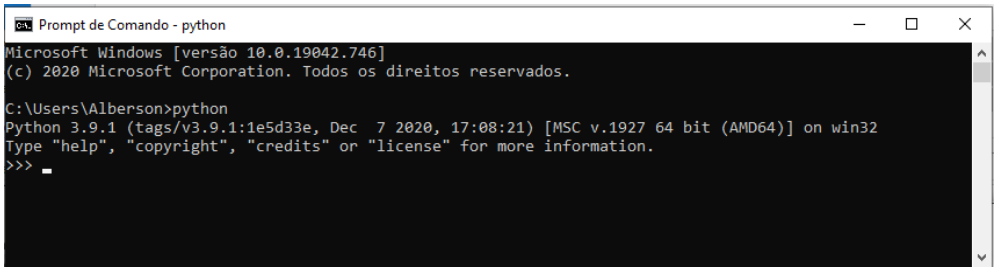
Terminada a instalação ou Upgrade, a seguinte tela aparecerá:



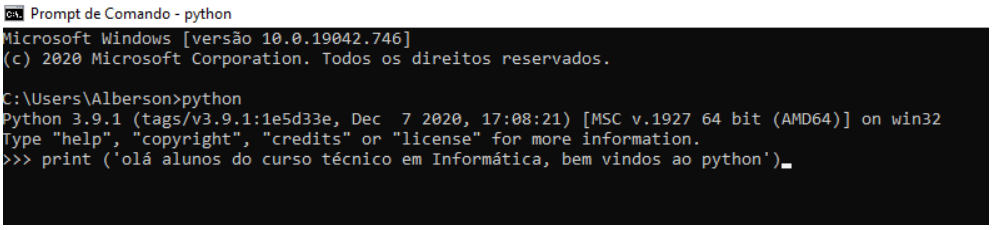
7.1 TESTANDO A INSTALAÇÃO REALIZADA COMO LINHA DE COMANDO

Agora basta entrar no tela de comando do Windows digitando “cmd”, veja figura a seguir :

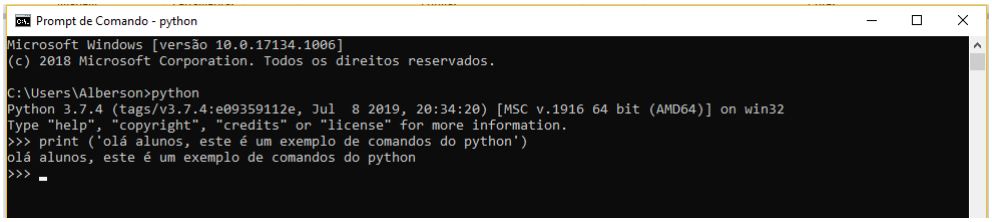
Deste ponto para frente manterei para exemplos a versão que tenho instalada no meu equipamento. Ao entrar na janela de comandos do windows, basta digitar no prompt de comando “python”, conforme exemplo a seguir:



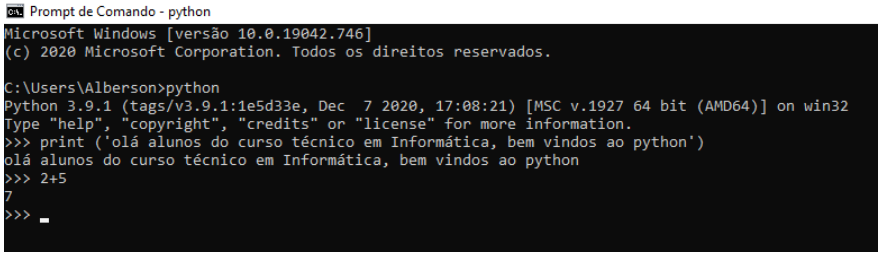
Para fins de teste, observe a seguir, o teste do comando “**print”** do python, exibindo uma mensagem qualquer:



Ao pressionar o <enter> no final da linha do print a mensagem será exposta, conforme resultado abaixo:

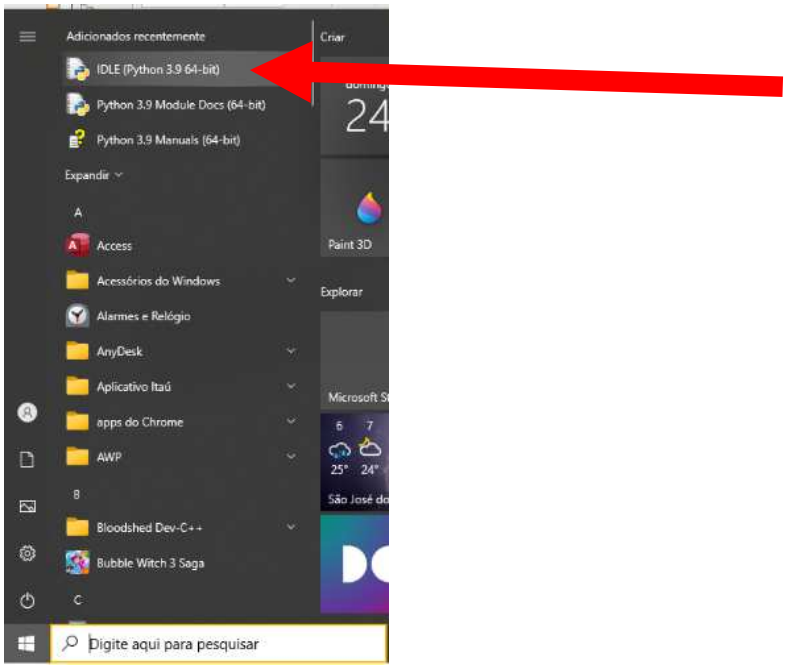


Um outro exemplo é mostrado a seguir, onde se pode realizar um cálculo através da digitação de “2+5”. Quando pressionado <enter> o resultado é exibido ao usuário, veja:

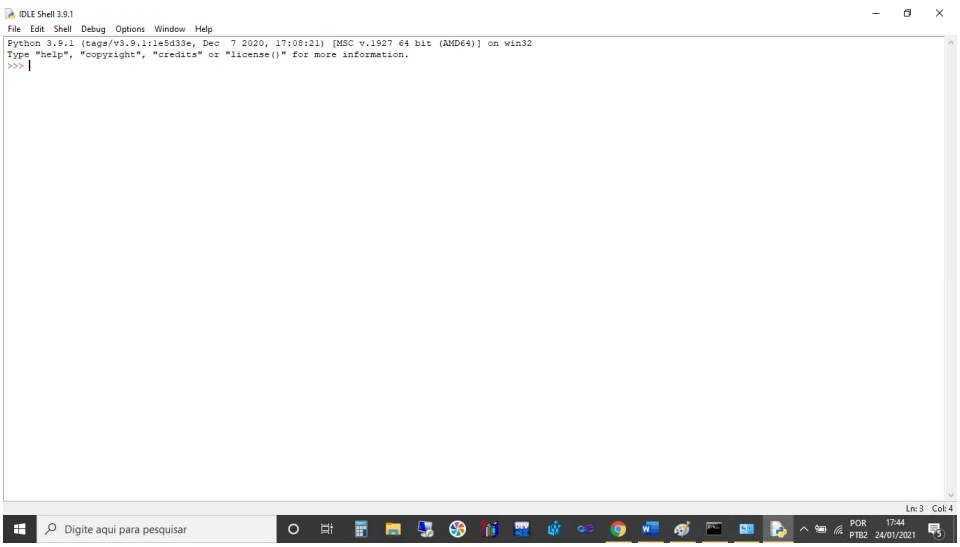


7.2 IDE DO PYTHON (IDLE) – MODO INTERATIVO

Para acessar a IDE do python basta abrir o IDLE, software que foi instalado quando instalamos o python. Acesse-o da seguinte forma:

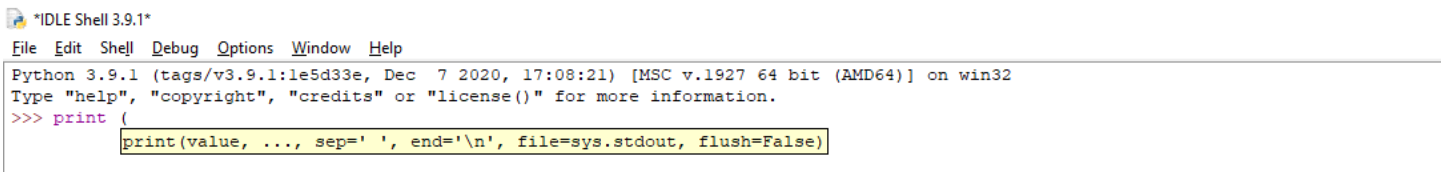


Ao clicar no IDLE a seguinte IDE será carregada. Nesta tela trabalharemos em **MODO INTERATIVO**, ou seja, após a digitação de um comando e pressionando <enter> os comandos já serão executados, linha a linha:

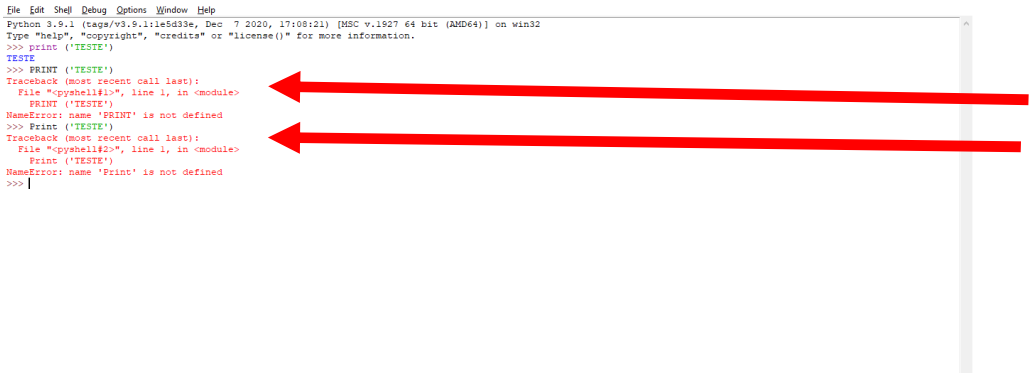


A tela exibida acima é o ambiente de execução de programas Python. Este ambiente é utilizado para não precisarmos abrir a tela de comandos do Windows (cmd) conhecida como tela de terminal.

Repare que no IDLE Shell, quando digitamos um comando, será exibida a sintaxe do comando em questão, conforme exemplo a seguir:



No próximo exemplo, você perceberá que o python é **CASE SENSITIVE**. Veja o que acontecerá quando você digita o comando print com letras maiúsculas e ou minúsculas. Observem os erros em vermelho quando o comando não consegue ser interpretado.

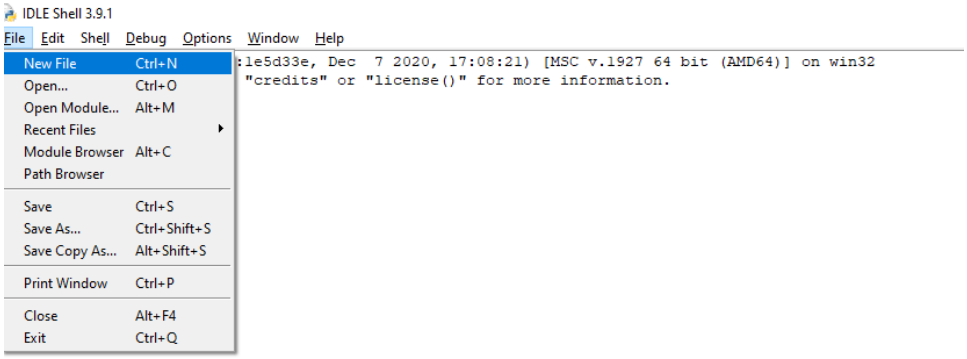


Seguindo com exemplos, observe o uso de algumas operações aritméticas, tal como: soma, subtração, multiplicação, divisão e resto de divisão.

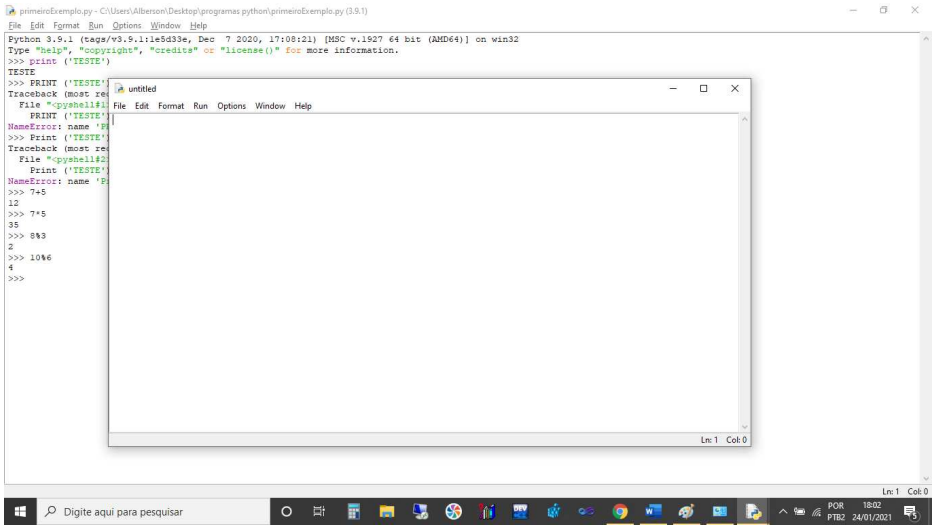
7.2.1 CRIANDO E GRAVANDO, ABRINDO, IMPRIMINDO E EXECUTANDO SEUS PROGRAMAS PYTHON (.PY) – MODO DE CRIAÇÃO DE SCRIPT

7.2.1.1 CRIANDO UM PROGRAMA NOVO:

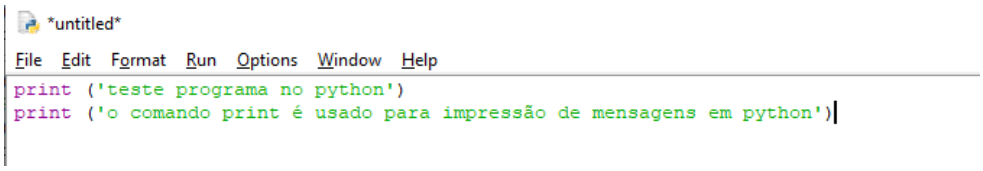
Você deve ter reparado, que para os testes realizados até agora usamos o IDLE SHELL, porém, é melhor utilizarmos sempre a área de **CRIAÇÃO DE SCRIPTS**. Para tal devemos usar os seguintes passos:



Surgirá então a seguinte janela intitulada “UNTITLED”, que é a área de **<u>criação de um script</u>** (script = programa = código), conforme figura a seguir:

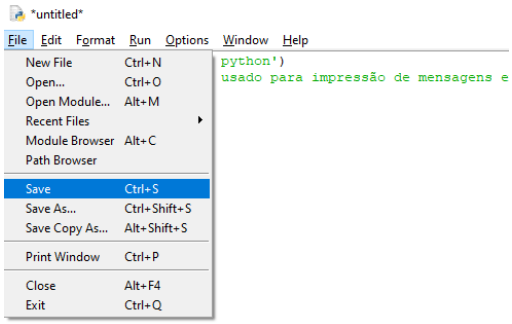


Na janela acima, digite seus programas, salve-os e execute-os, veja exemplo:

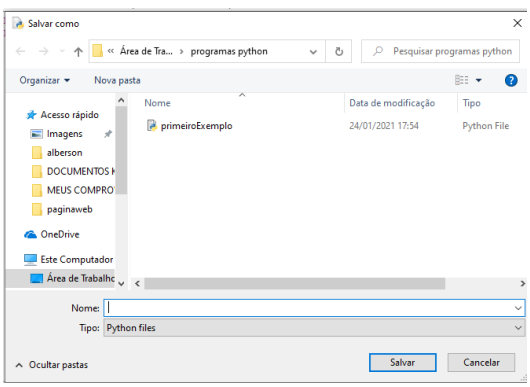


7.2.1.2 GRAVANDO O PROGRAMA CRIADO(DIGITADO):

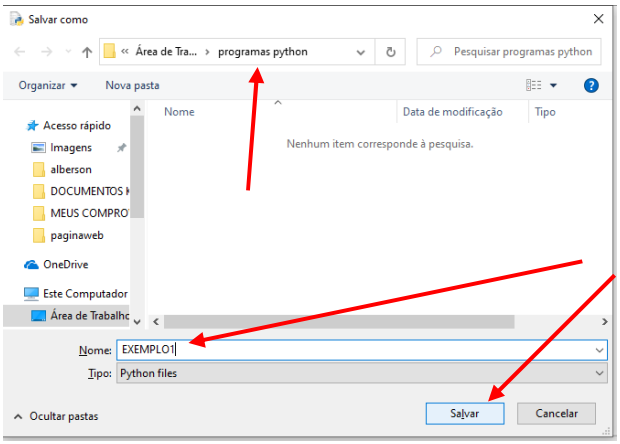
Após digitar todo seu programa, vamos salvá-lo. Para isso siga os passos abaixo:



Em seguida, surgirá a janela para escolhermos a pasta e unidade de disco na qual desejamos gravar o programa, veja:

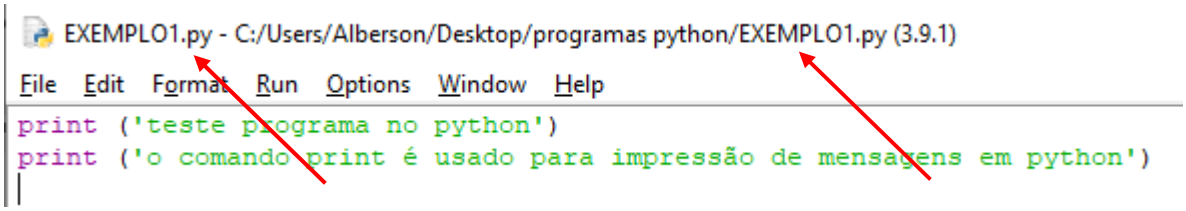


Assim como qualquer janela de diálogo do Windows, crie uma pasta qualquer na unidade de disco de sua preferência para gravar seus programas. Após digitar o nome do programa desejado, clique em “salvar”. Para exemplificar, façamos o seguinte: Vou nomear nosso pequeno programa como “EXEMPLO1.py”. Aliás vale ressaltar que os programas python possuem a extensão “.py”. Mantenha o TIPO de arquivo sempre como “PYTHON FILES”.

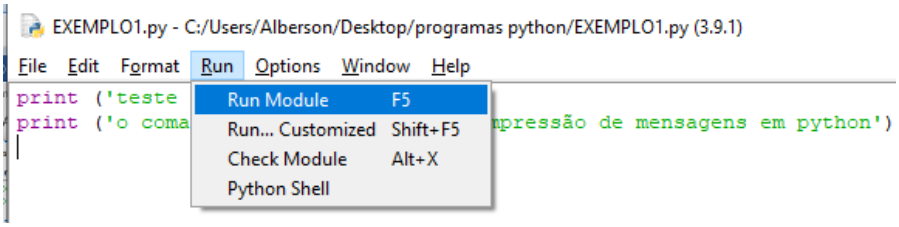


7.2.1.3 RODANDO, EXECUTANDO UM PROGRAMA ABERTO:

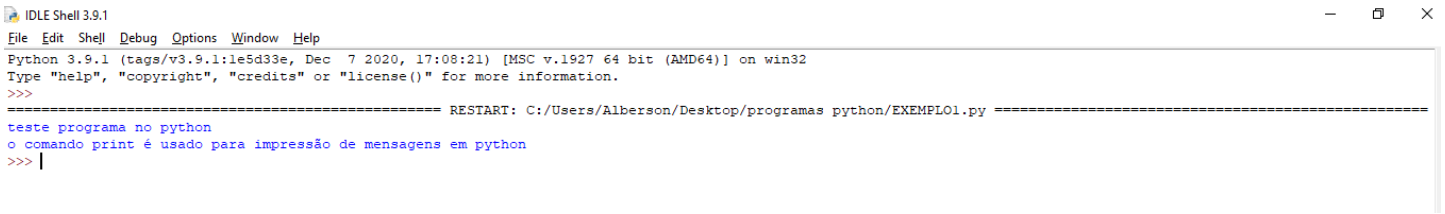
Repare que após gravar/salvar seu programa, a linha de título da janela da área de criação de scripts intitulada como “UNTITLED” mostrará o nome do programa python e o caminho (pasta) onde ele foi gravado, veja:



Para rodar (testar) seu programa basta seguir os passos abaixo:

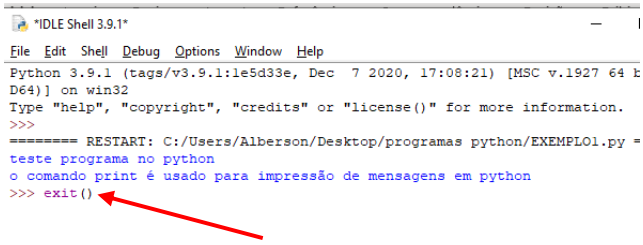


Ao invés de usar o menu “Run”, você também poderá optar por pressionar a tecla “F5” que também fará o programa ser executado. Veja resultado de execução do nosso pequeno programa de exemplo:

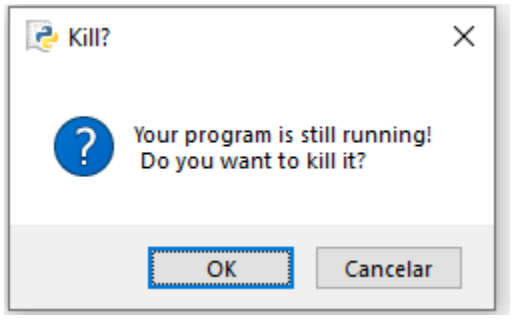


Perceba acima que os comandos não são mais mostrados no IDLE Shell, somente o resultado da execução das linhas do comando print. Logicamente você deve ter percebido que o IDLE Shell foi chamado para mostrar o programa rodando.

Caso você digite “exit()” no prompt do IDLE Shell e pressionar <enter>, a janela será fechada e você poderá voltar para janela de script que se mateve aberta atrás do IDLE Shell:



Surgirá a seguinte pergunta: “Seu programa ainda está rodando! Você quer matá-lo?”. (“matá-lo” quer dizer parar a execução do programa)

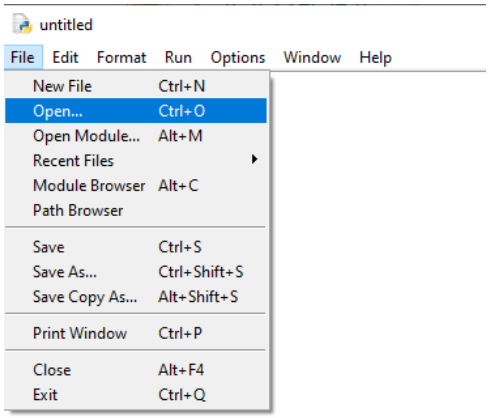


Se clicar em “OK”, a execução será encerrada.

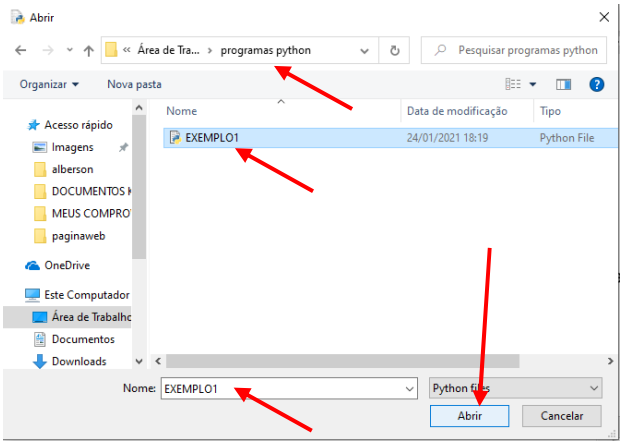
Se clicar em “Cancelar”, permanecerá no IDLE SHELL.

7.2.1.4 ABRINDO UM PROGRAMA PYTHON GRAVADO:

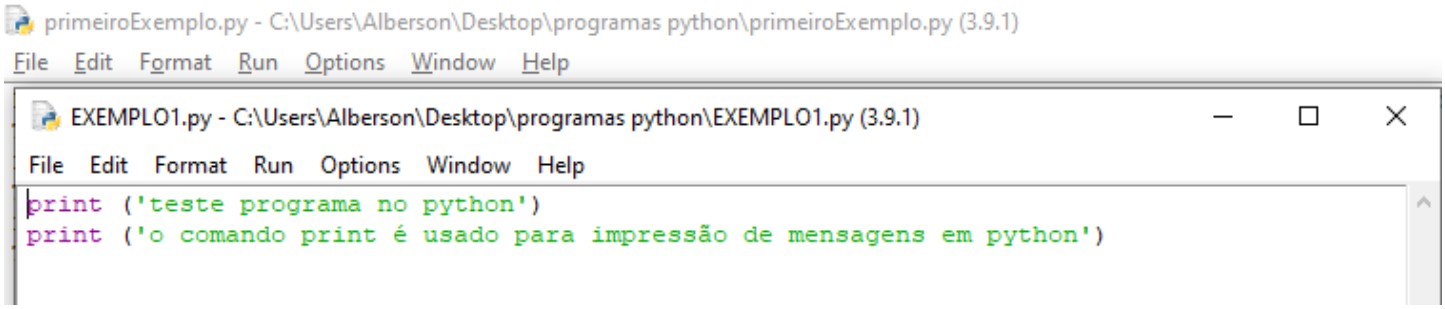
Para abrir um programa python, anteriormente gravado, basta seguir os passos abaixo:



Surgirá uma janela de diálogo onde devemos selecionar unidade de disco e a pasta onde o arquivo ".py" foi gravado. Ao encontrá-lo dê selecione-o e clique no botão "Abrir", veja:



Ao clicar no botão "Abrir" seu programa será carregado na área de Script, conforme próxima figura:



7.3 DIFERENÇAS ENTRE MODO INTERATIVO E MODO DE SCRIPT DO IDLE

Em resumo, o MODO INTERATIVO serve para testarmos linhas de comandos do Python. Já o MODO SCRIPT deve ser usado sempre que for necessário a criação de um programa completo (SCRIPT).

8. OUTROS EDITORES DE CÓDIGO PYTHON

Para programar em Python, vimos anteriormente o IDLE no módulo interativo e de script. Há, porém, outros editores (IDE) que permitem a criação de programas python. Podemos usar alguns editores de texto e também alguns editores FREE ou pagos. Vale informar que não testei todos estes editores, somente alguns deles. A seguir estão relacionados os mais utilizados por programadores e que podem facilitar o trabalho de desenvolvimento dos seus programas:

8.1 EDITORES DE TEXTOS:

Há vários editores de textos que podem ser usados para edições de programas Python. Destaco aqui os dois mais interessantes. Entretanto, acredito que o próprio editor IDLE é mais recomendável. Em todo caso, segue dois nomes bem conhecidos:

Atom

Atom é open source e feito pelo Github e com suporte para várias linguagens, dentre elas o Python. É integrado ao Git e Github, sendo possível mexer com o Git e Github através da interface do editor de texto. Ótimo para iniciantes.

Visual Studio Code
O VSCode é open source e free, desenvolvido pela Microsoft. Suporta inúmeras linguagens de programação.
EDITORES DE CÓDIGOS (FREE): Os editores de código Python são mais recomendáveis do que o uso de editores de texto. A saber, segue alguns mais conhecidos entre os desenvolvedores:

IDLE: (RECOMENDO USO DESTA FERRAMENTA PARA DESENVOLVIMENTO E TESTE DOS SEUS PROGRAMAS)
A IDLE vem com o Python. É feita com Tkinter e se você se acostumar pode lhe ajudar bastante. É bem simples de ser usada também.

PyCharm community (RECOMENDO USO DESTA FERRAMENTA PARA DESENVOLVIMENTO E TESTE DOS SEUS PROGRAMAS)
É desenvolvido pela companhia JetBrains. Esta edição é liberada sob a licença da Apache. É multiplataforma. Essa IDE fornece análise de código, um depurador gráfico, um testador de unidade integrado, integração com sistemas de controle de versão (VCSes), e suporta desenvolvimento de web com Django.

NetBeans
Analogamente ao Eclipse, o NetBeans também oferece suporte ao Python através de plugins

NINJA-IDE
Do acrônimo recursivo: "Ninja-IDE Is Not Just Another IDE", é uma IDE multiplataforma de desenvolvimento integrado. NINJA-IDE é executado em Linux/X11, Mac OS X e sistemas operacionais de desktop Windows, e permite aos desenvolvedores criarem aplicações para diversas finalidades, utilizando todas as ferramentas e utilitários de NINJA-IDE, tornando a tarefa de escrever software mais fácil e agradável.

Eclipse
Diferente de todos os outros. Pesado, grande, monstruoso mas muito poderoso. É feito em Java e é ideal para desenvolvimento Java. Mas existem plugins para se desenvolver em Python com ele (e detalhe: atualmente é um brasileiro quem o mantém) que é o **ppydev**.

EasyEclipse
EasyEclipse é open source e hospedado pela Sourceforge que fornece muitas distribuições empacotadas do Eclipse pré-configuradas com plug-ins para Python, Ruby, etc.

DrPython
Usa wxPython. Criado para ser utilizado em escolas.

8.2 EDITORES PYTHON PARA CELULAR:

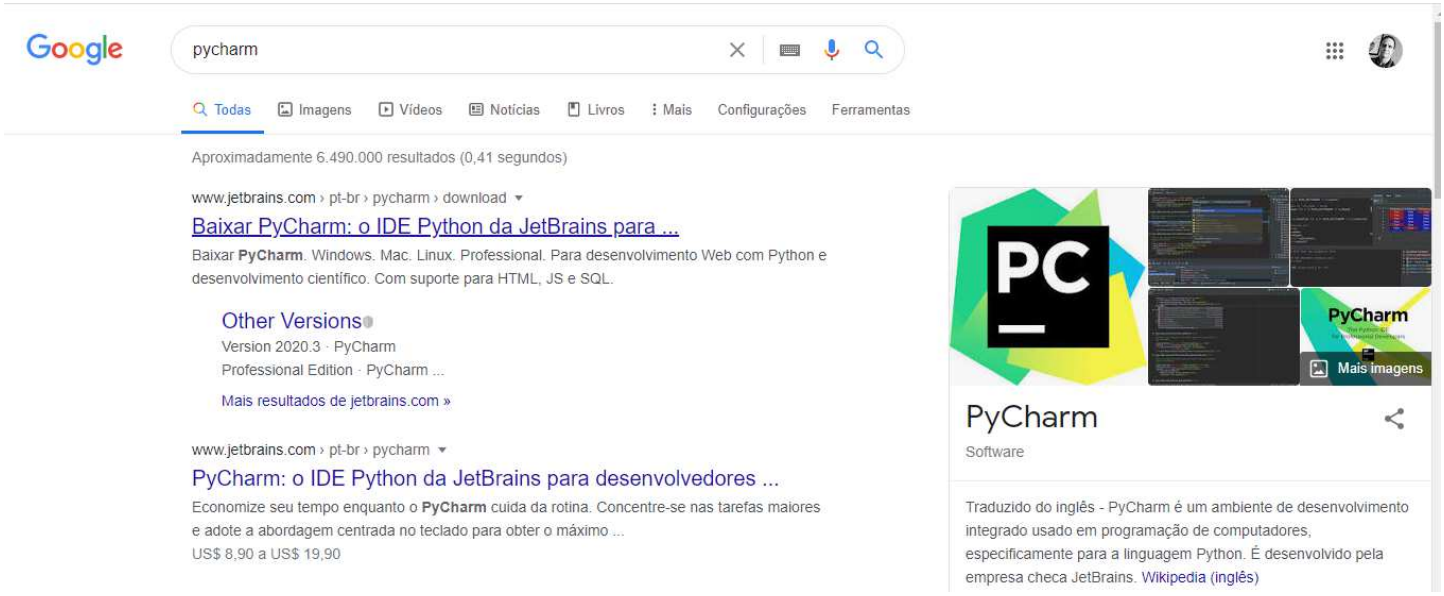
Existem na playstore do seu celular alguns APPS com os quais podem ser desenvolvidos e testados seus programas PYTHON. Testei o **QPython3** e funcionou. Ressalto que para teste dos programas desenvolvidos, você deve colocar a extensão **".PY"** **após o nome do arquivo**, pois caso não o faça o QPython não identificará como sendo um programa python e não rodará seu programa. Faça o teste você também !!!

Existem também alguns EDITORES PAGOS, mas não vamos mencioná-los nesta apostila.

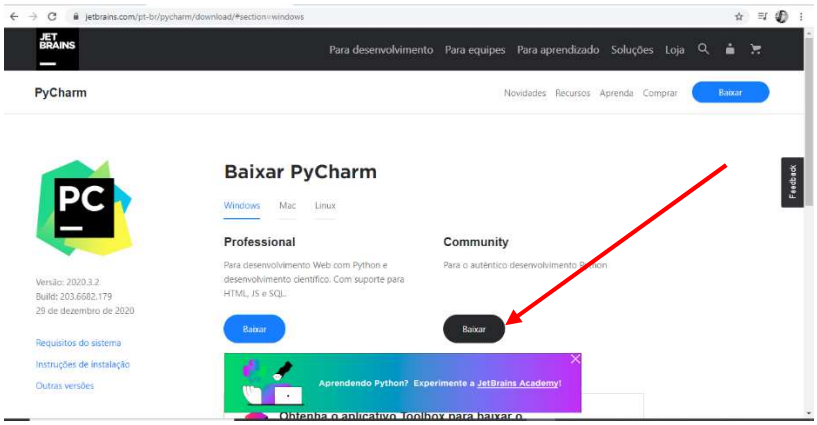
9. INSTALANDO E USANDO A IDE PYCHARM

Para termos uma ferramenta alternativa para desenvolvimento dos nossos programas deste curso vamos instalar o PYCharm. Para isto, siga os passos a seguir:

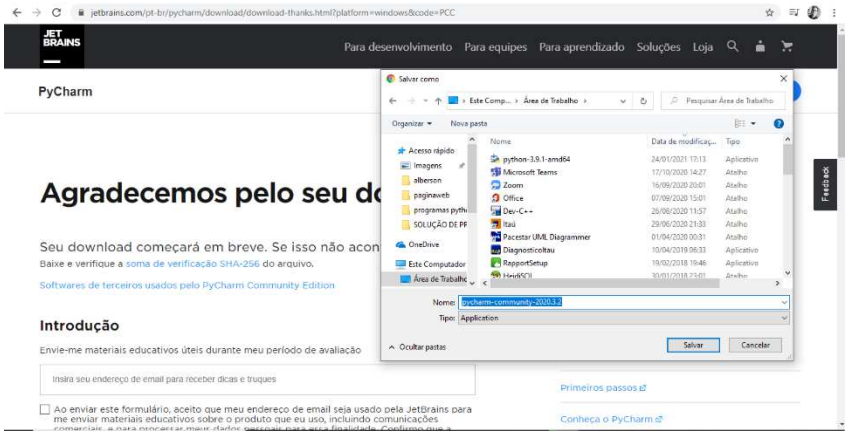
a. Procure no site de busca do seu navegador um PYcharm, conforme imagem a seguir:



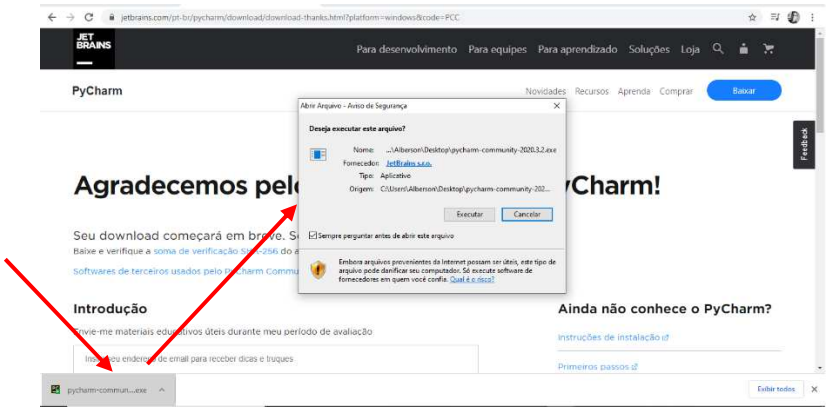
b. Vamos clicar no primeiro link de resultado da página de busca mostrada acima. Logicamente, pode ocorrer de seu resultado de busca ser diferente. O que importa na verdade é procurar pela área de download da empresa JETBRAINS, desenvolvedora do PYCharm. Será aberta, então, a seguinte página da empresa citada:



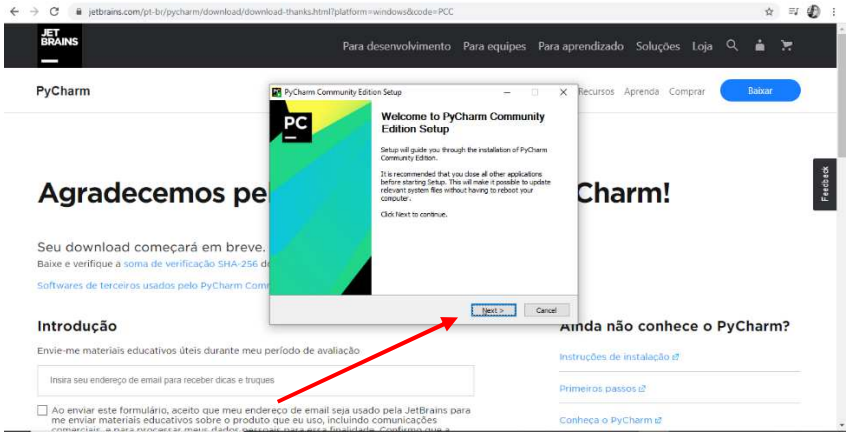
c. Conforme indicado pela seta vermelha, baixe a versão da comunidade (COMMUNITY), pois a versão Professional é paga e esta é para uso livre. Após clicar no botão “baixar” (preto) surgirá a seguinte página e lhe será solicitado o local para gravar o arquivo de instalação, que será baixado, veja:



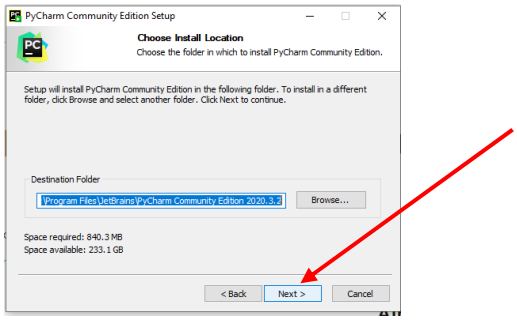
d. No exemplo acima, vamos baixar o arquivo de instalação na área de trabalho. Após o download realize a instalação. Veja exemplo:



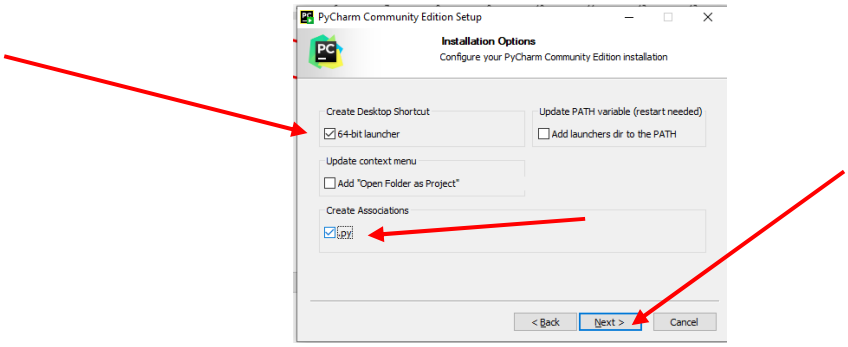
e. Basta então, clicar no botão “Executar”, para dar início a instalação. Surgirá uma tela solicitando autorização para alterar configurações. Responda que permite. Diante dessa resposta incisar-se-á a instalação, conforme mostrado a seguir:



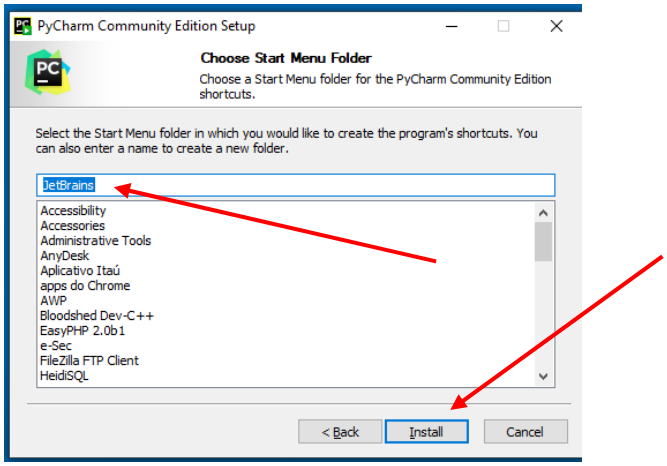
f. Clique no botão “Next” na janela acima. Surgirá então:



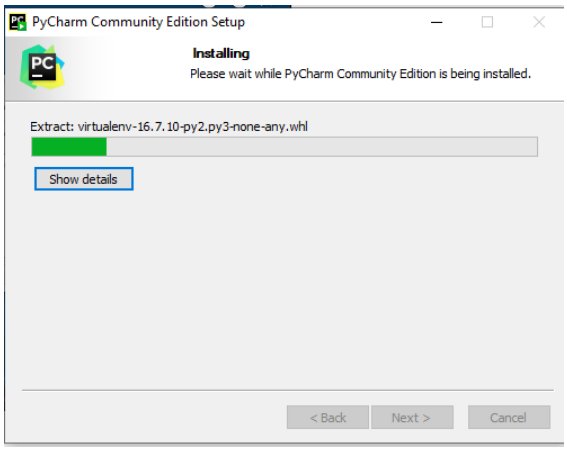
g. Nesta tela acima clique novamente em “Next”, para surgir a próxima tela:



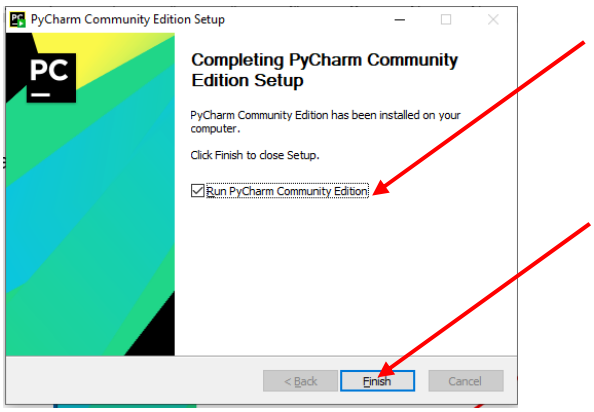
h. Selecione as opções da tela e clique em “Next” novamente. Surgirá então:



i. Mantenha “JetBrains” selecionado e clique em Install, para inicio da instalação do PyCharm:

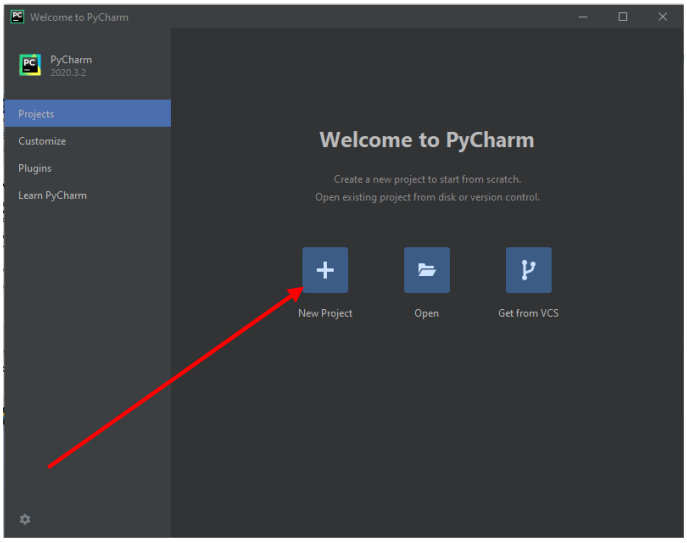


j. Ao término da instalação, surgirá:



k. Selecione “Run PyCharm Community Edition” e clique em “Finish”. Surgirá:

l. Mantenha “Do not Import Settings” marcada e clique em “Ok”. Se tudo deu certo o PYCharm será carregado e a tela inicial aparecerá como mostrada a seguir:

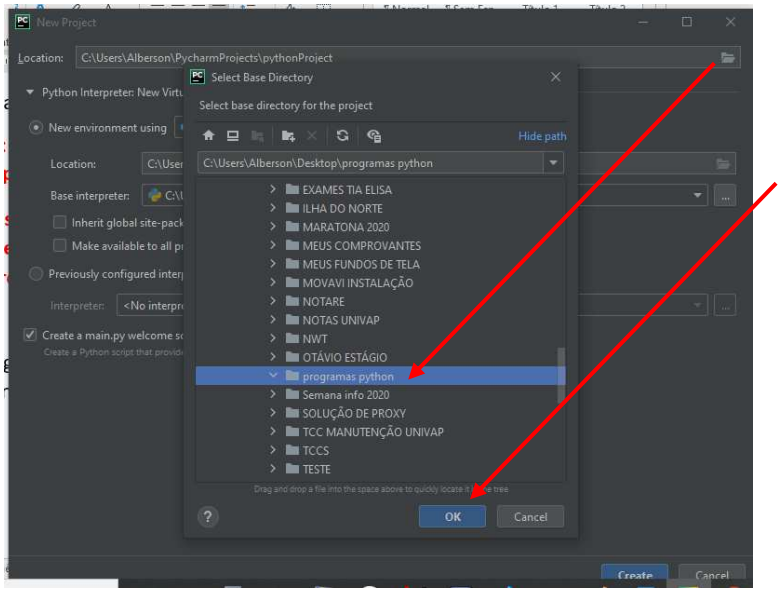


m. Para efeitos de teste, vamos clicar no botão “+” para criar novo projeto.

OBSERVAÇÕES: Um projeto nada mais é que uma pasta no seu HD. Ela armazenará todos os códigos de vários programas que serão criados. Por exemplo: Você poderá ter um projeto para guardar:

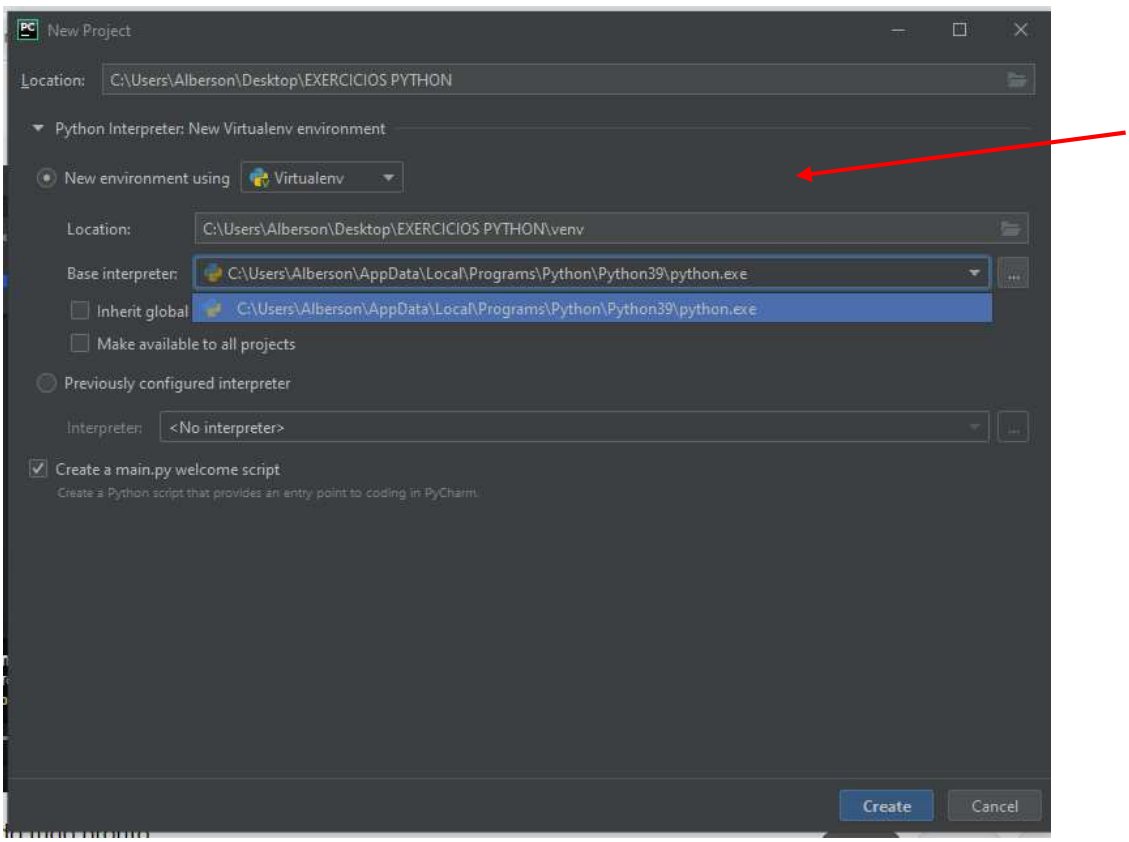
- **todas as suas listas de exercícios de cada bimestre;**
- **ou projetos bimestrais da nossa matéria;**
- **ou exercícios de aulas;**
- **etc...**

n. Defina então a pasta (ou crie-a) para armazenar os programas python que iremos desenvolver. No meu caso indicarei uma pasta existente na minha área de trabalho chamada “EXERCÍCIOS PYTHON”, veja:

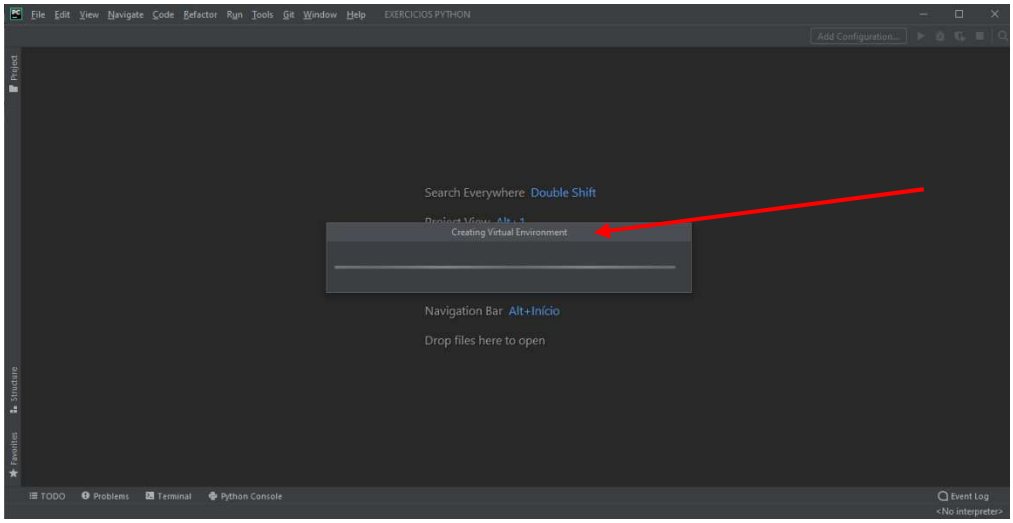


Observação: Perceba acima, que ao clicar no ícone pastas, posso estar selecionando a pasta onde salvarei meus programas. Ao definir a pasta, clique em “Ok”.

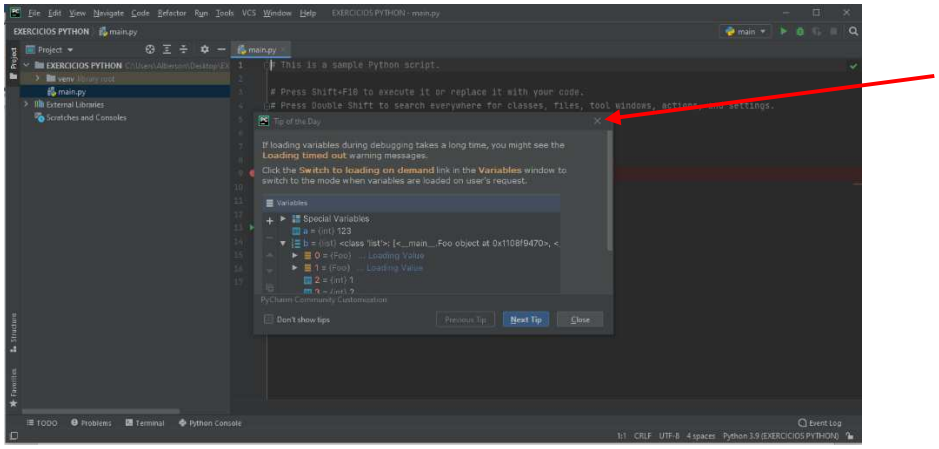
o. O próximo passo é definirmos qual a versão do Python irá gerar os programas que você vai desenvolver na pasta “EXERCÍCIOS PYTHON”. As versões do python podem ser escolhidas no campo “Base Interpreter”. Quando você tiver várias versões do Python baixadas e instaladas no seu computador, elas aparecerão neste local. Após escolher a versão basta clicar no botão “Create”. Seguindo estes passos você está criando o projeto “EXERCÍCIOS PYTHON”. Veja:



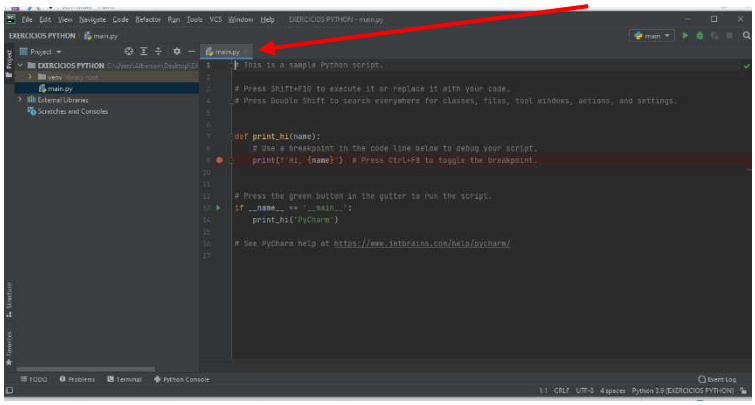
p. Após clicar no botão “Create”, surgirá a seguinte tela, indicando a criação do seu projeto.



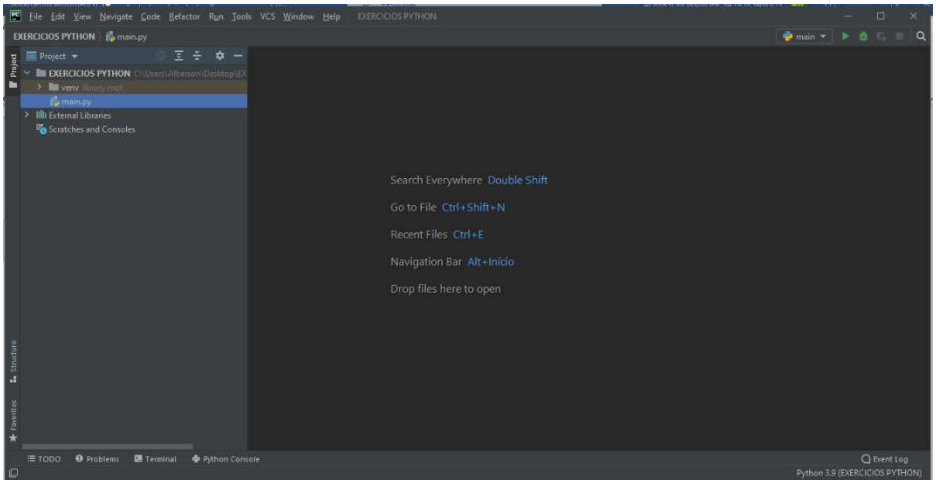
Após aguardar alguns momentos, uma tela parecida com a figura abaixo irá aparecer para você:



q. Pode fechar a janela “Tip of The Day”, a qual lhe dá dicas do dia. Sua tela deverá se parecer com a mostrada abaixo:

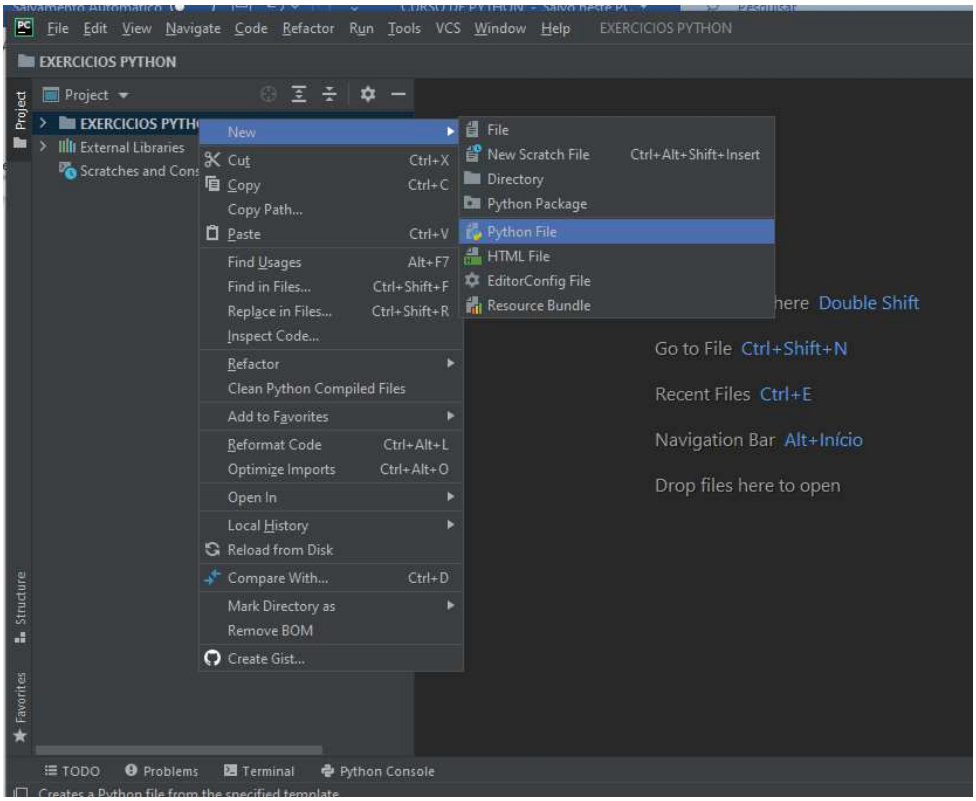


r. Pode fechar também o programa “main.py”, pois trata-se de um exemplo de programa python. Sua tela deverá ter a seguinte aparência:

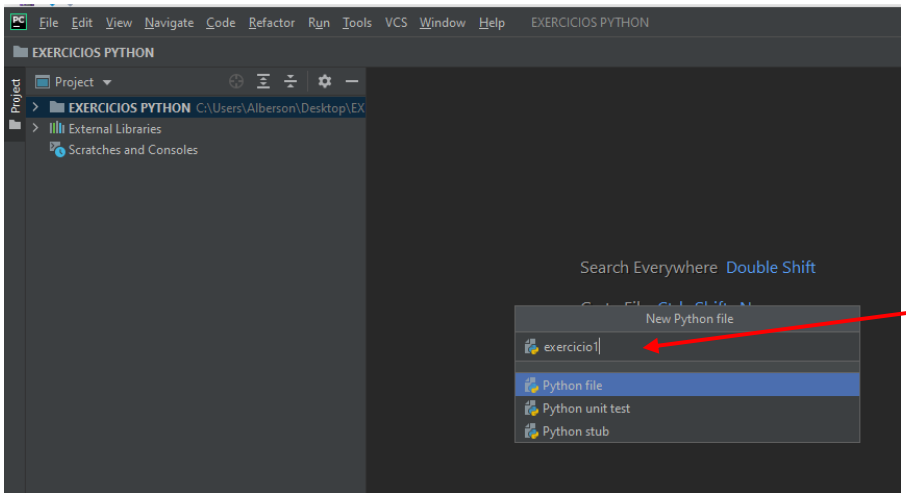


9.1 NOSSO PRIMEIRO PROGRAMA NO PYCHARM:

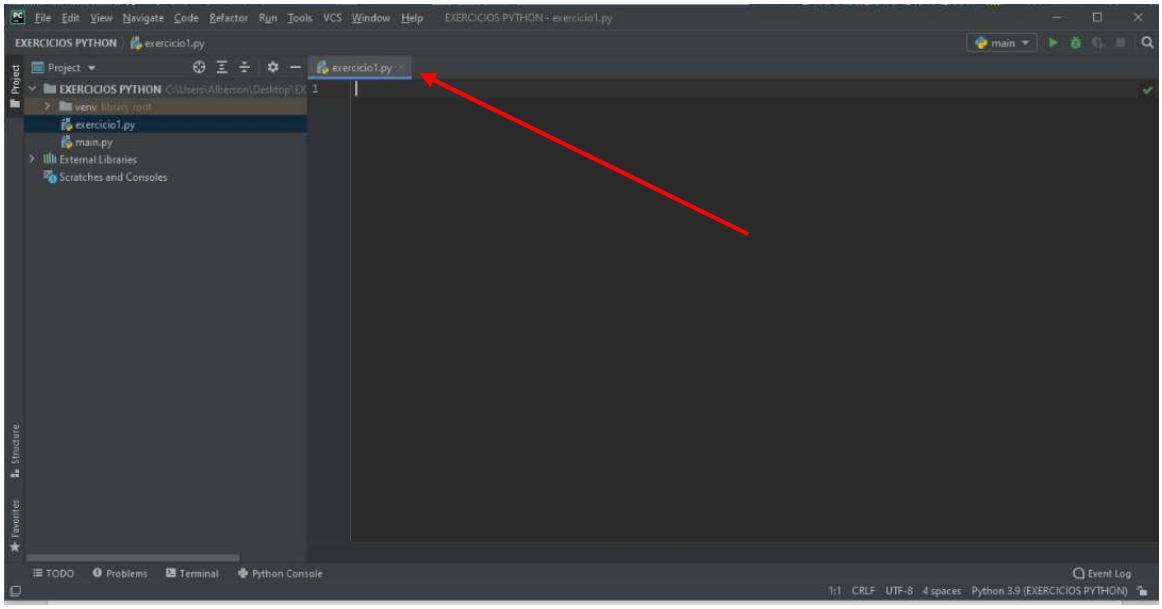
Para exemplificar a criação de um primeiro programa no PyCharm, clique com botão DIREITO do mouse sobre o nome do projeto “EXERCÍCIOS PYTHON” e siga os passos abaixo:



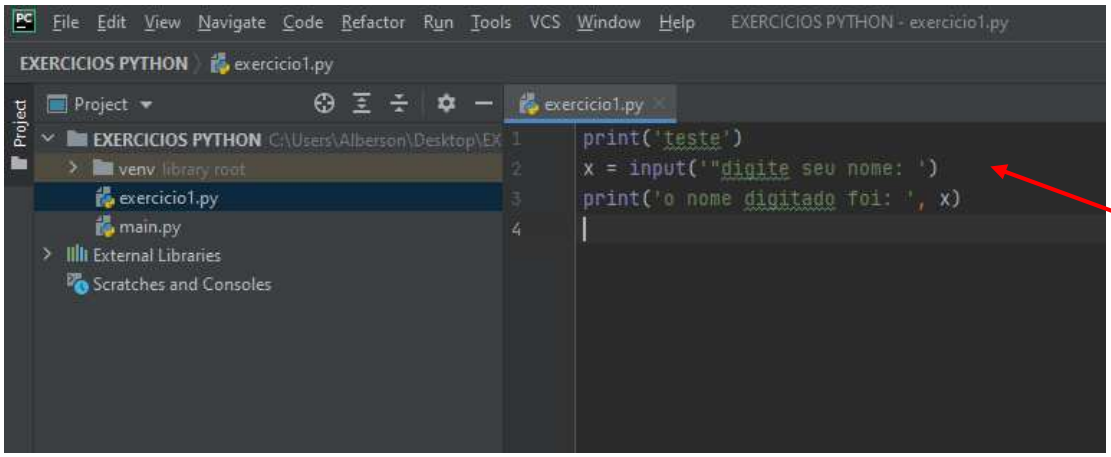
Na próxima janela que aparecer, informe o nome do seu primeiro script Python, conforme exemplo abaixo e pressione <enter>:



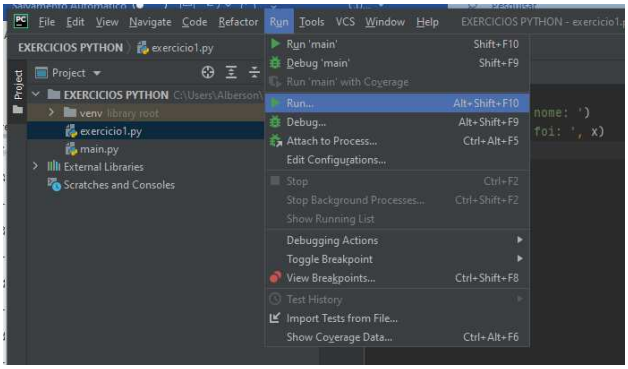
Deverá aparecer a área para digitação do nosso primeiro programa:



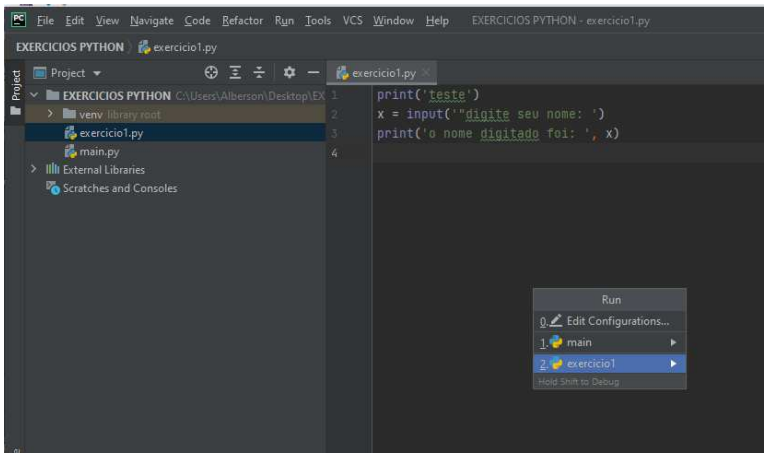
Para exemplificar vamos digitar um simples programa Python, conforme próxima figura:



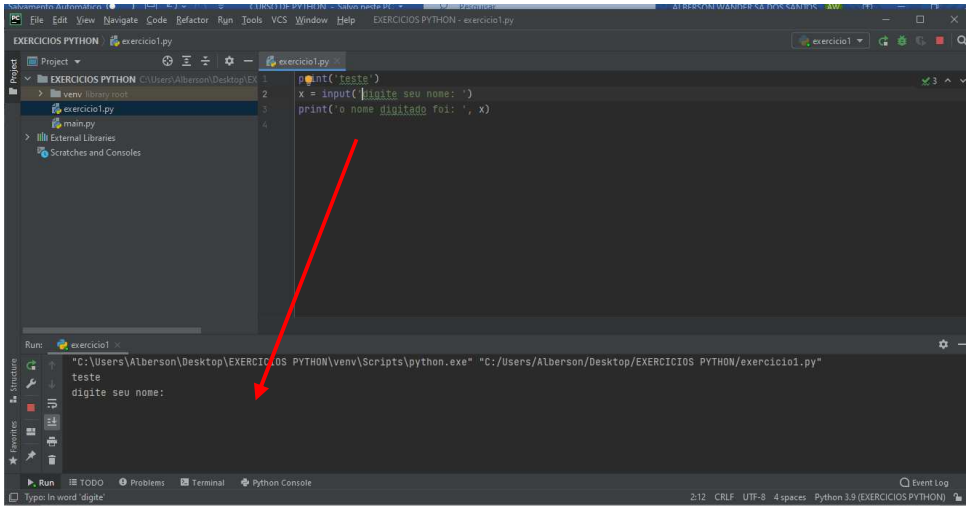
Repare acima, que o programa “exercicio1.py” foi criado dentro da pasta de projeto “EXERCÍCIOS PYTHON”. Agora vamos testar este programa. Para isso siga os passos da próxima figura:



Quando clicar em “Run...”, será mostrada a janela para você selecionar qual programa deve ser rodado; para nosso caso o Pycharm irá questionar se deseja rodar o “main.py” ou o “exercicio1.py”. Vamos selecionar “exercicio1.py”:

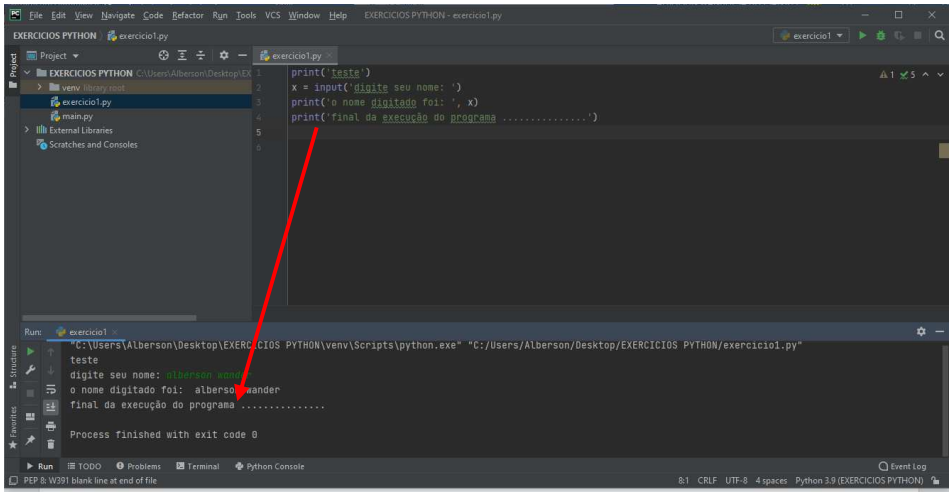


A tela de execução será mostrada da seguinte forma:

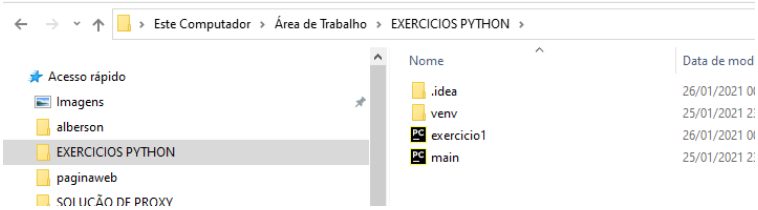


Respondendo ao que o programa está solicitando, digitando meu nome “ALBERSON WANDER”, teremos então o seguinte resultado:

Quando precisar alterar o código, basta fazê-lo e, novamente, mandar executá-lo. Veja um exemplo:



Só para efeitos de curiosidades, minha pasta na área de trabalho já possui o projeto de exemplo “main.py” e também o programa que criei, chamado “exercicio1.py”, perceba:

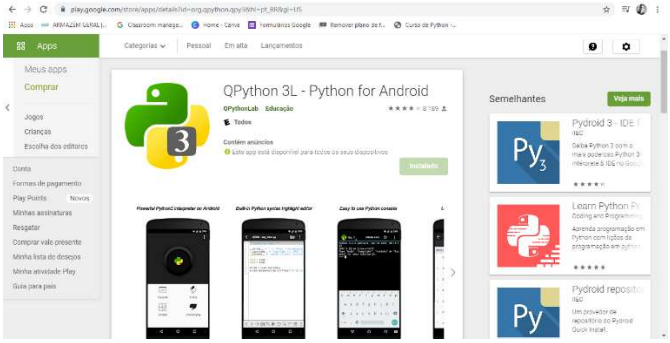


Com este tópico entendido, você será capaz de usar mais esta ferramenta para desenvolver os projetos em Python.

Você poderá usar a IDE que mais acha fácil para desenvolver seu programa python. Como citado anteriormente existem vários editores no mercado.

Porém, para as aulas de laboratório usaremos somente o IDLE. Vale dizer que os programas feitos no IDLE do Python ou no PyCharm podem ser abertos em ambas IDE´s.

10. USE O QPYTHONL NO SEU CELULAR ANDROID – IDE PARA DESENVOLVER NO CELULAR



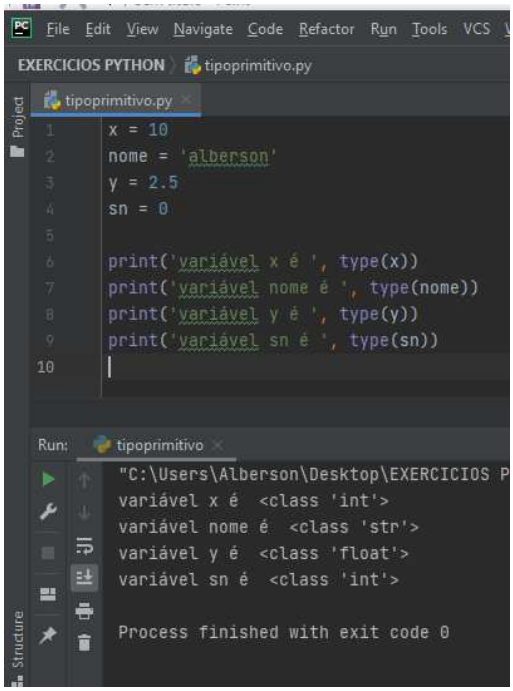
Baixe na PlayStore do seu telefone o app acima para treinar o desenvolvimento no seu celular. Você poderá executar em modo INTERATIVO e modo de SCRIPT (PROGRAMAÇÃO), tal como no IDLE.

LINGUAGEM PYTHON

11. VARIÁVEIS E TIPOS PRIMITIVOS:

Em Python **não precisamos CRIAR** variáveis, basta usá-las no código. Dependendo do valor atribuído a variável ela assumirá o tipo de dado a ela atribuída, por exemplo, INTEIRO, REAL, BOOLEANO OU CARACTERES (int(), float(), bool() ou str(), respectivamente).

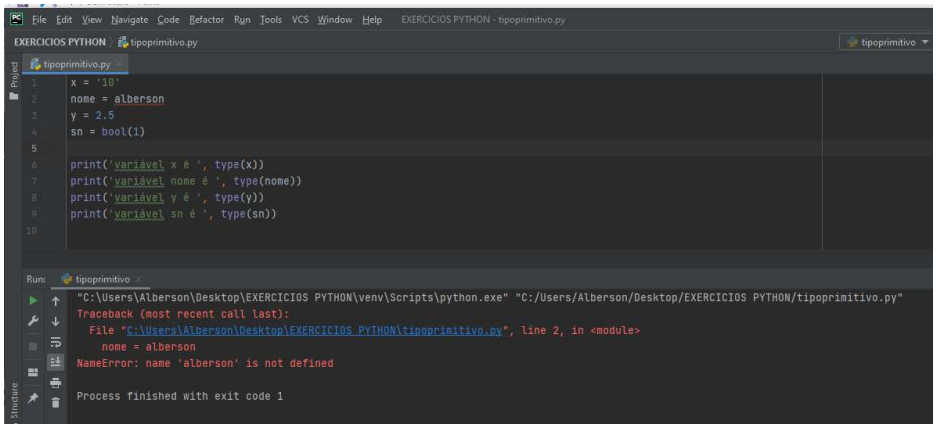
Vejamos exemplos abaixo:



Vamos entender!!!

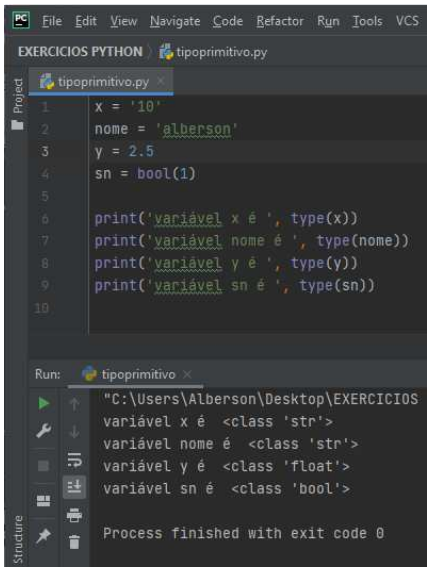
No pequeno programa acima, fiz a impressão do tipo de dado de cada variável. O TIPO DE DADO foi definido pelo dado atribuído a cada variável. Perceba que os números não estão postos entre aspas, assim não numéricos int ou float. O float é definido com o separador de casas decimais, que no Python é o "." (ponto, não use vírgula)

Veja um exemplo interessante:



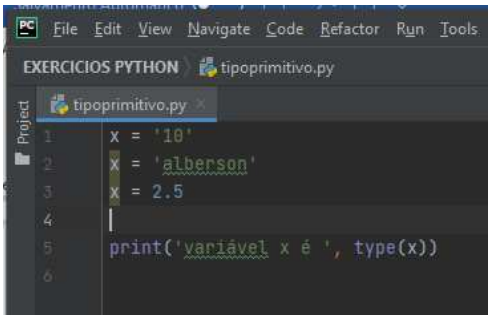
Este programa deu erro porque a variável ***nome*** está recebendo um dado que não é definido; o Python estava esperando que ***alberson*** fosse uma variável que tivesse guardando um dado qualquer. Este programa não rodou por este motivo.

Arrumando a linha com erro teremos:

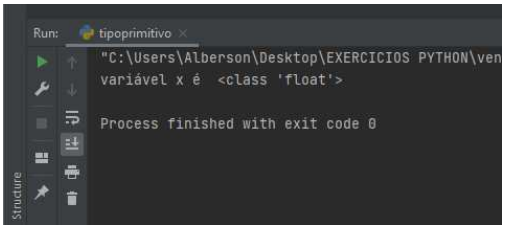


CURIOSIDADE: REPARE NO CÓDIGO ACIMA QUE A VARIÁVEL ***sn***, foi convertida. O valor 1 agora será booleano, daí o tipo bool.

*Será que você é capaz de responder qual é o tipo de dado da variável **x**?*



Imagino que você tenha entendido, pois foi impresso a informação que ***x*** é do tipo float!!! Veja abaixo:



12. CONVERSÃO DE TIPOS USANDO FUNÇÕES int(), float(), bool() e str():

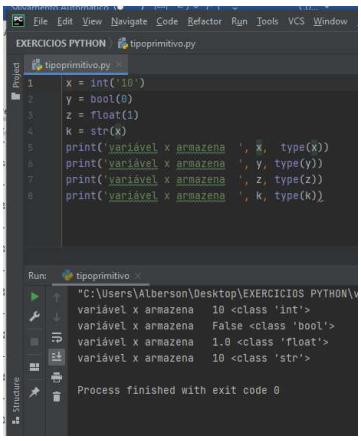
int() – Converte um dado para o tipo inteiro.

float() – Converte um dado para o tipo real.

bool() – Converte um dado para o tipo booleano(ou lógico).

str() – converte um dado para o tipo caractere (string)

Exemplos:



13. OPERADORES ARITMÉTICOS DO PYTHON:

Operação	Operador
adição	+
subtração	-
multiplicação	*
divisão	/

13.1 OUTROS OPERADORES USADOS EM EXPRESSÕES ARITMÉTICAS:

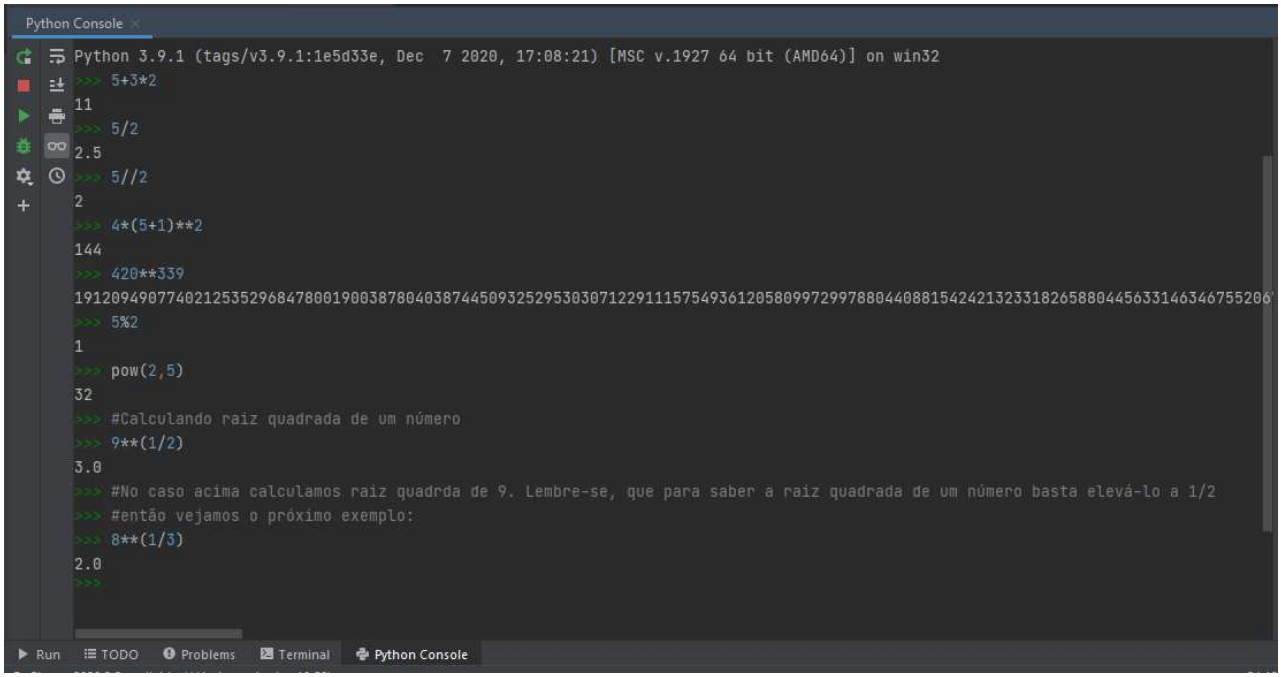
Operação	Operador
Exponenciação	**
Parte inteira do resultado da divisão	//
Módulo (Resto de Divisão)	%

14. EXPRESSÕES ARITIMÉTICAS E ORDEM DE RESOLUÇÕES:

ORDEM DE PRECEDÊNCIA DOS OPERADORES ARITMÉTICOS:

OPERADOR	ORDEM DE RESOLUÇÃO NA EXPRESSÃO
()	1º
**	2ª
*, /, //, %	3ª
+, -	4ª

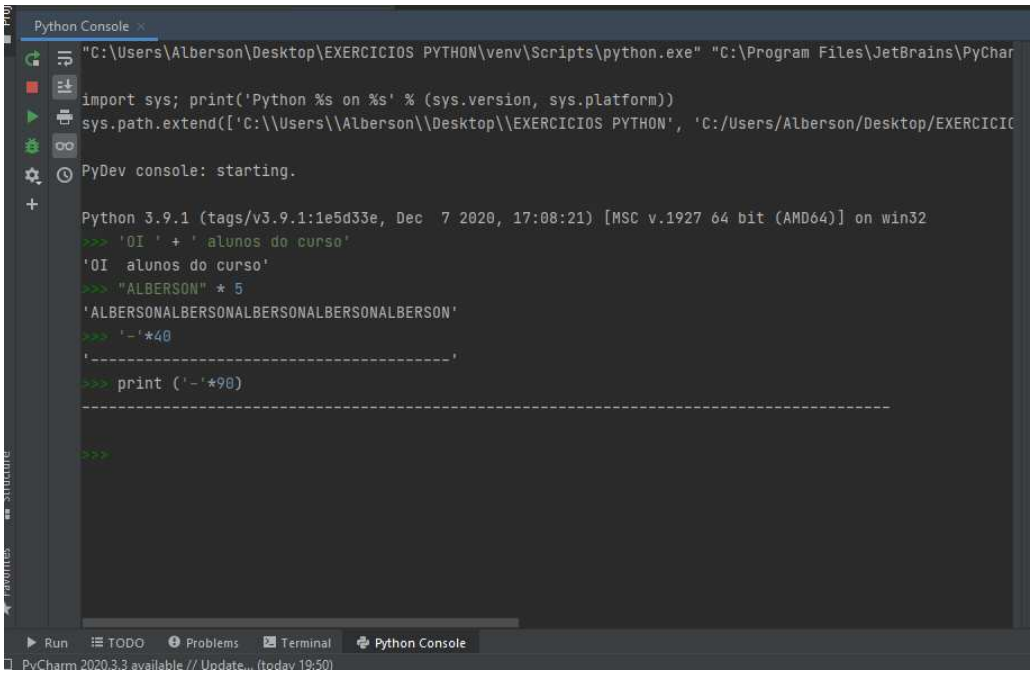
Vejamos alguns exemplos de uso de operadores aritméticos no PYTHON:



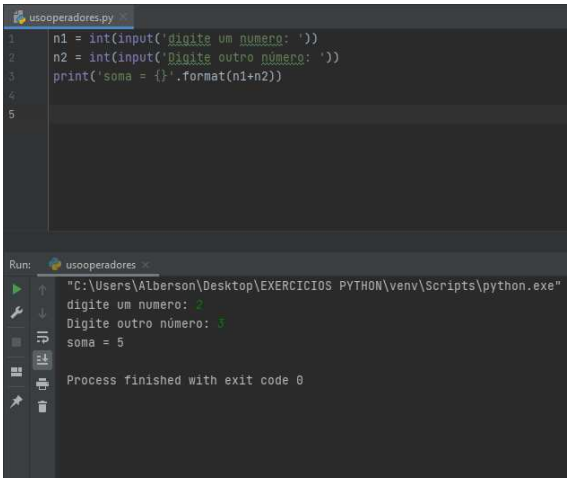
Os exemplos de cálculos acima merecem algumas considerações, a saber:

1) 5//2, resultou em 2, pois foi ignorado a parte fracionada do resultado;
2) 420**339, resultou em um número extremamente grande, cuja memória foi capaz de armazenar;
3) Calculo da raiz quadrada, ou raiz cúbica de um número, elevamos um número a ½ ou 1/3, respectivamente, como feito na matemática.
4) É bom ressaltar que no PYTHON a MEMÓRIA DO COMPUTADOR é o limite para o armazenamento de resultado de cálculos. Assim sendo, poderemos armazenar números extremamente grandes, sem a necessidade de informarmos à linguagem um tipo de dado específico para tal.

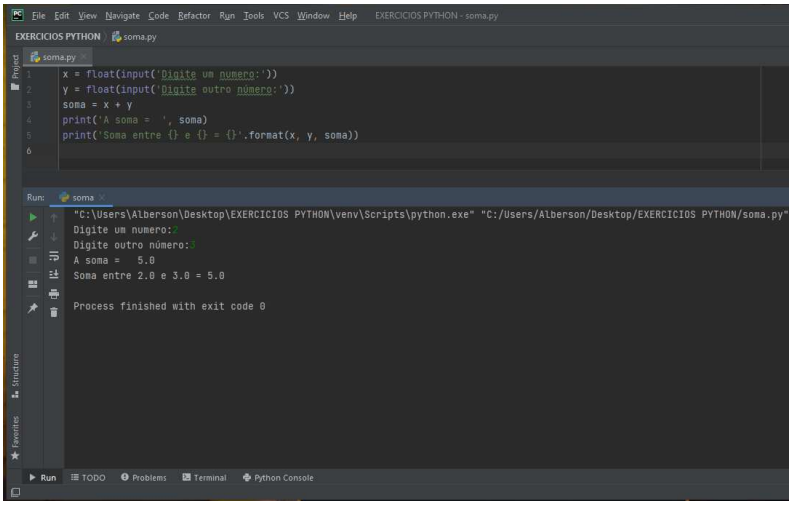
VEJA QUE INTERESSANTE OS EXEMPLOS ABAIXO, USANDO ALGUNS OPERADORES ARITMÉTICOS COM STRINGS !!!



Vejamos alguns exemplos de usos de operadores aritméticos em programa Python:



Vamos então fazer um programa para fazer operações matemáticas usando conversores de dados em expressões aritméticas, usando os operadores mostrados:



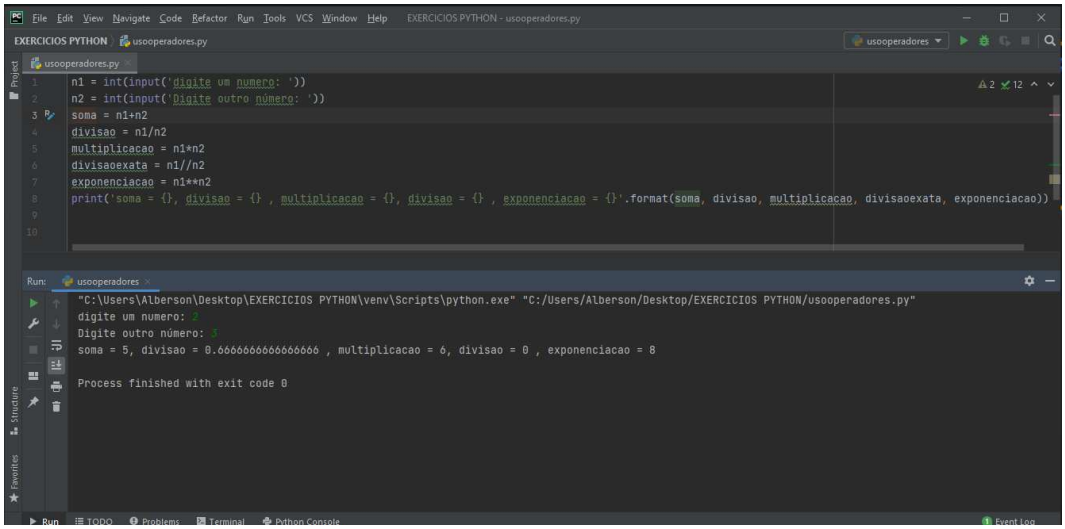
OBSERVAÇÕES GERAIS SOBRE O PROGRAMA ACIMA:

a) Na linha 1 e 2 do programa, percebam que os dados digitados pelo usuário serão alterados convertidos para float;
b) Na linha 5 do programa, observem que dentro das aspas do comando print usamos três '{}' e que fora das aspas usamos a função .format(), a qual imprimirá no locais respectivos das '{}' os valores armazenados nas variáveis x, y e soma;
c) OUTRA SINTAXE QUE FUNCIONARIA PARA LINHA 5 DO PROGRAMA:

```
print(f'Soma entre {x} e {y} = {soma}')
```

EXPLICANDO O COMANDO ACIMA:

O f colocado antes da mensagem, no comando print acima, indica que deve ser formatado os valores da vairáveis x, y e soma. REPARE QUE NÃO FOI NECESSÁRIO ISOLAR ESTAS VARIÁVEIS FORA DAS ASPAS (''), POIS OS COLCHETES FAZEM O PYTHON ENTENDER QUE ESTAMOS TRATANDO DA INCLUSÃO DE UMA VARIÁVEL



CURIOSIDADE !!!!!!!!!!!

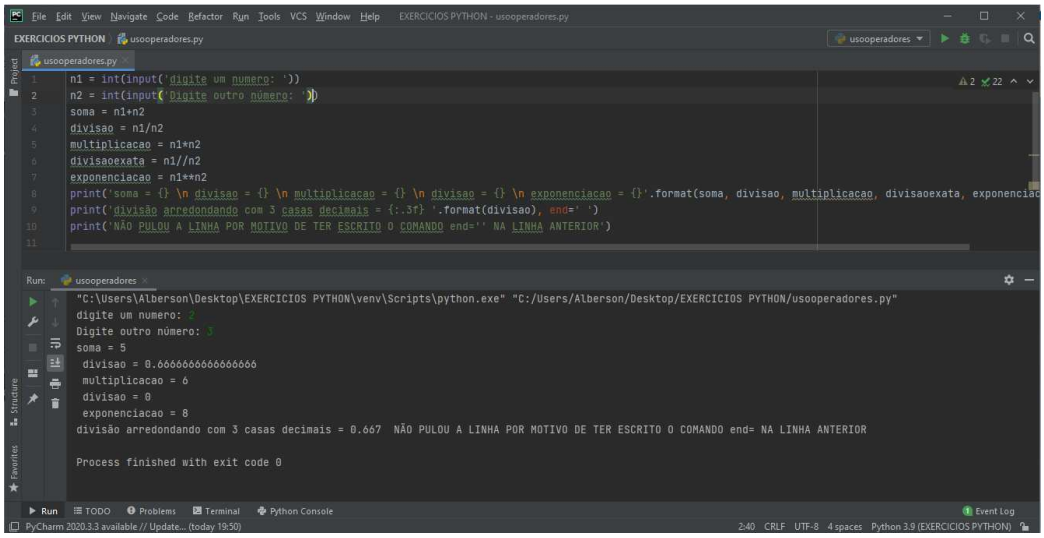
VOCÊ PODE TAMBÉM DETERMINAR A QUANTIDADE DE CASAS DECIMAIS DO RESULTADO DA DIVISÃO DO PROGRAMA ACIMA, PARA ISSO USE DENTRO DAS {}, NO COMANDO *print* A SEGUINTE SINTAXE:

```
print('divisão arredondando com 3 casas decimais = {:.3f} '.format(divisao))
```

{:.3F} SIGNIFICA PARA MOSTRAR O RESULTADO DA DIVISÃO FORMATADO COM 3 DÍGITOS FLUTUANTES, VEJA O RESULTADO QUE SERÁ EXIBIDO:

divisão arredondando com 3 casas decimais = 0.667

Veja também como funciona a sintaxe para quebra de linha e também para juntar linhas de resultados no python, usando função print():



OBSERVE NO RESULTADO DA EXECUÇÃO DO PROGRAMA ACIMA:

,end = ‘ ’, após o parâmetro .format(), na linha 9, a PRÓXIMA LINHA FOI IMPRESSA NA FRENTE DO CONTEÚDO IMPRESSO POR ESTA LINHA DE COMANDO

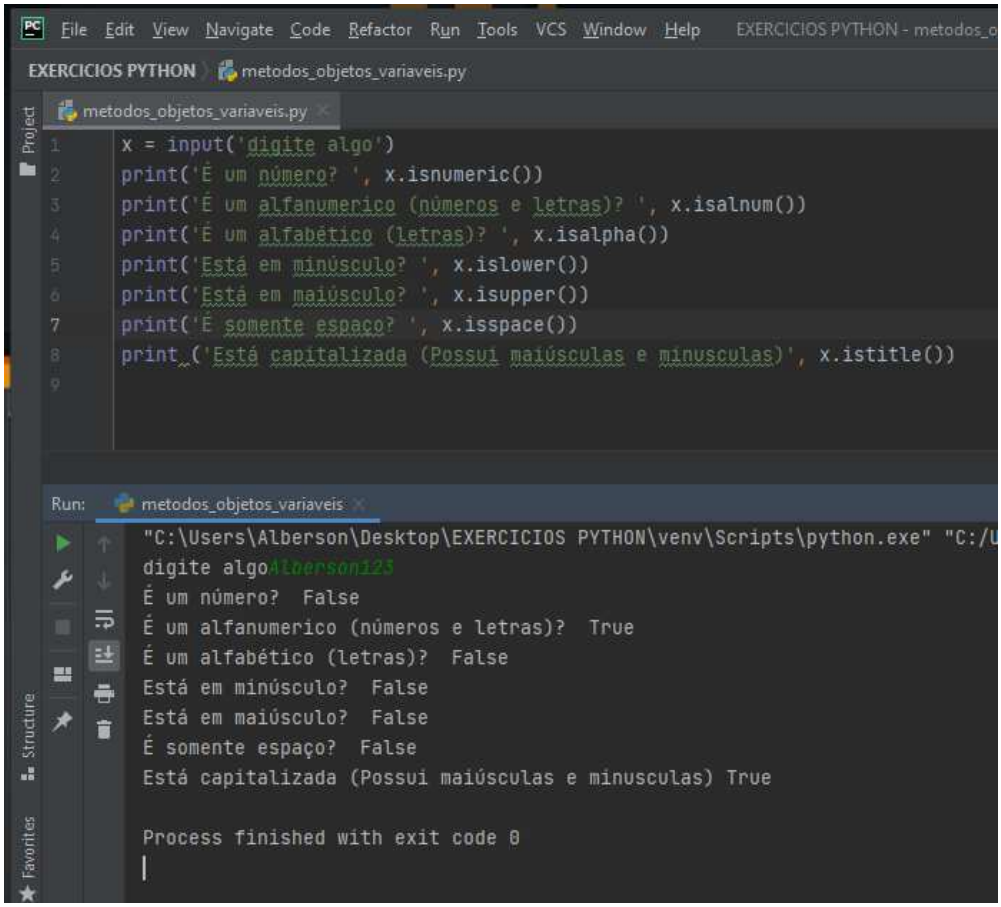
\n, escrito dentro das aspas ‘’ do comando print, na linha 8, forçou quebra da linha impressa exatamente nos pontos onde foram escritos.

15. MÉTODOS DE OBJETOS

VOCÊ SABIA QUE UMA VARIÁVEL EM PYTHON TAMBÉM É UM OBJETO?

VEJA QUE INTERESSANTE!

Vamos conhecer ALGUNS MÉTODOS DE UM OBJETO STRING criado num programa, veja uma relação de métodos que podem ser usados:



MÉTODOS USADOS ACIMA:	SIGNIFICADO, OU RETORNO:
isnumeric()	Verifica se o objeto guarda um valor numérico (int ou float)
isalnum()	Verifica se o objeto guarda um valor alfanumérico (números e letras)
isalpha()	Verifica se o objeto guarda um valor alfabético (somente letras)
islower()	Verifica se o objeto está guardando somente letras minúsculas
isupper()	Verifica se o objeto está guardando somente letras maiúsculas
isspace()	Verifica se o objeto está guardando somente sequência de espaços
Istitle()	Verifica se o objeto está guardando letras maiúsculas e minúsculas, ou seja, se está **capitalizada**

IMPORTANTE: VOCÊ PERCEBEU O USO DOS MÉTODOS?

LEIA ABAIXO A EXPLICAÇÃO COM ATENÇÃO:

O que conhecíamos como simplesmente VARIÁVEIS até então, agora é um OBJETO.

REPAROU QUE OS MÉTODOS USADOS RETORNARAM TRUE(SIM) OU FALSE(NÃO)?

ATENÇÃO: NEM SEMPRE TEREMOS ESTES RETORNOS.

O que é um MÉTODO em POO (PROGRAMAÇÃO ORIENTADA A OJETO)?

Os métodos, em sua essência, nada mais são que procedimentos (vistos em linguagens estruturadas). OS MÉTODOS SÃO APLICADOS AOS DADOS ARMAZENADOS POR CADA OBJETO

Nos exemplos temos a seguinte linha de programa, por exemplo:

```
print('É um número? ', x.isnumeric())
```

Neste caso:
x é OBJETO, ou seja, você deve ter percebido então que uma variável é um objeto em python.
isnumeric() é um método que verifica se o DADO armazenado pelo objeto x é um número.

Como saber se estou usando um MÉTODO ou um ATRIBUTO do OBJETO?

Resposta:

OS '()' PARENTESES ESCRITOS AO FINAL DA SINTAXE, INDICAM QUE ESTAMOS TRATANDO DE UM MÉTODO

É BOM SABER!!!!

TODO OBJETO TEM CARACTERÍSTICAS E FUNCIONALIDADES, OU SEJA, ELE TEM ATRIBUTOS E MÉTODOS, respectivamente.

16. MÓDULOS (BIBLIOTECAS)

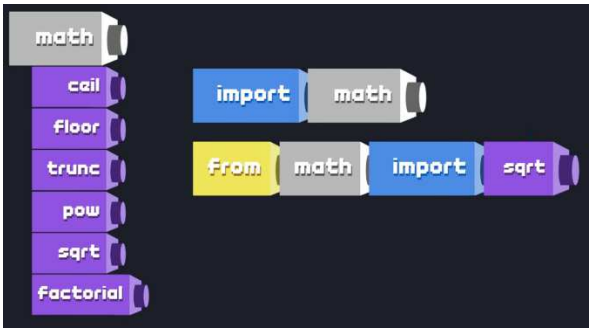
Comandos:

***import* <biblioteca>**

***from* <biblioteca> *import* <funcionalidade>**

Veja exemplo abaixo:

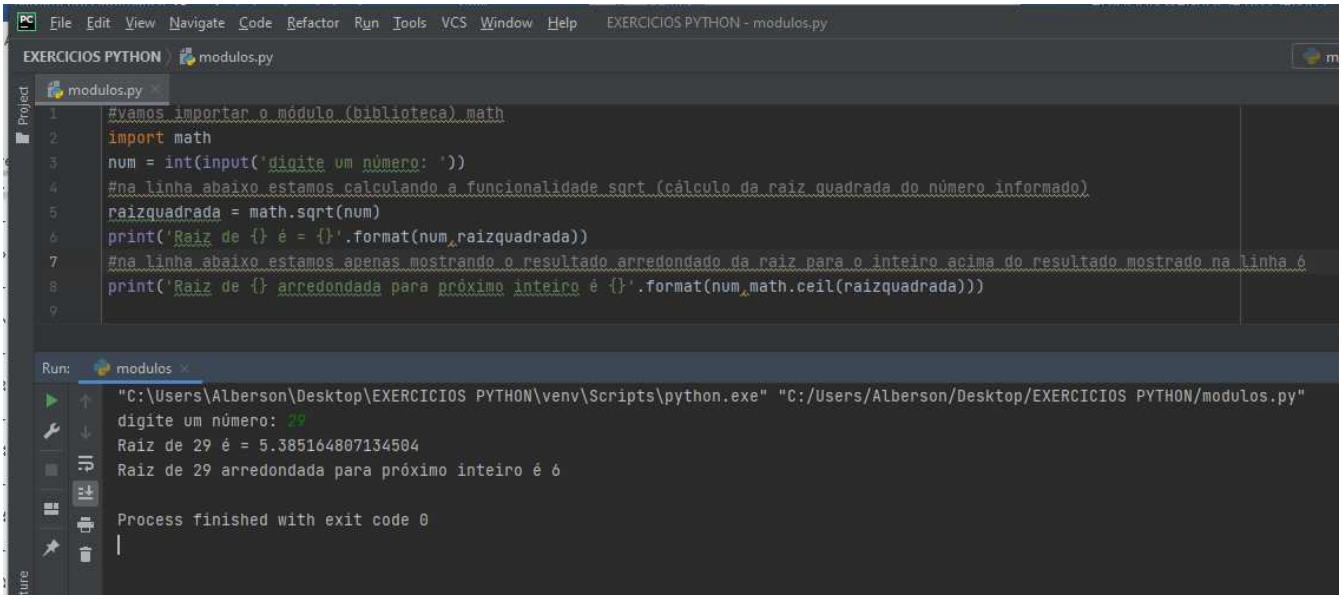
Temos a biblioteca math, usada normalmente para uso de algumas funções extras de cálculos matemáticos. Perceba na figura a seguir que a referida biblioteca possui várias funcionalidades, a saber:



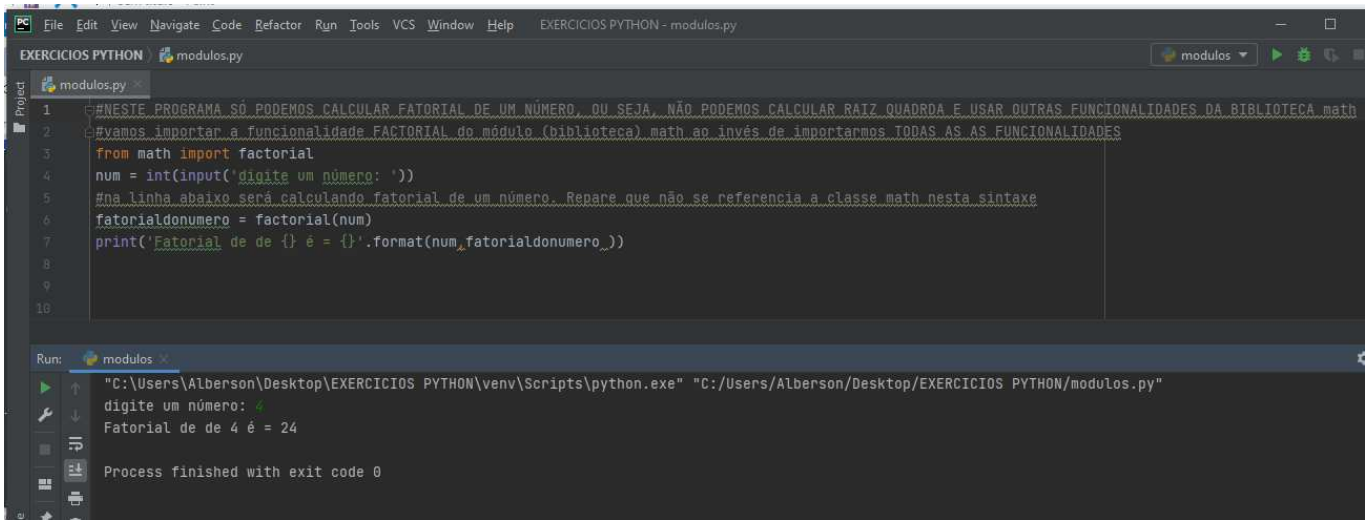
ceil – usada para arredondamento para o maior inteiro de um número fracionado
floor – usado para arredondamento para o menor inteiro de um número fracionado
trunc – usado para ignorar parte fracionária de um número
pow – usada para cálculo de potenciação
sqrt – usada para cálculo de raiz quadrada de um número
factorial – usada para cálculo de fatorial de um número

De acordo com a figura anterior, podemos importar, POR EXEMPLO, TODAS AS FUNÇÕES MATEMÁTICAS da biblioteca ***math***, ou então, as que eu desejar.

EXEMPLO 1:

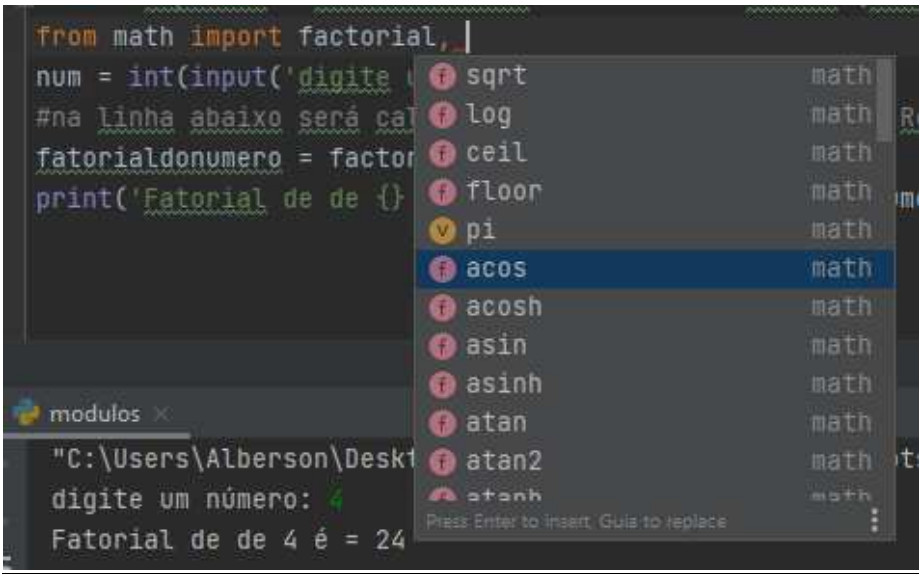


EXEMPLO 2:



EXEMPLO 3:

Podemos importar mais de uma funcionalidade de uma biblioteca, basta separá-las por ‘,’ (vírgula), veja:



Se pressionar <ctrl><barradeespaço> serão mostradas todas as funcionalidades de uma classe que você pode importar, conforme na figura acima.

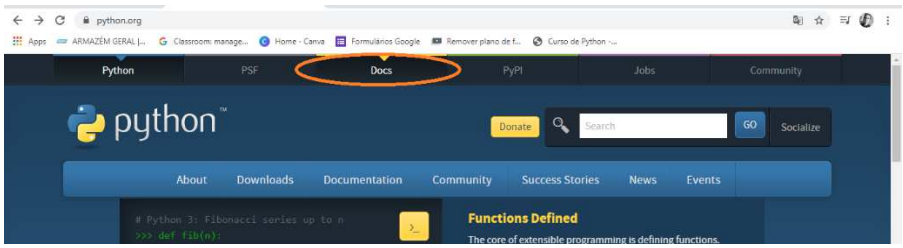
LEMBREM-SE!!

Quando usamos SOMENTE IMPORT podemos trabalhar com qualquer funcionalidade(função - método) de uma bilblioteca (classe)

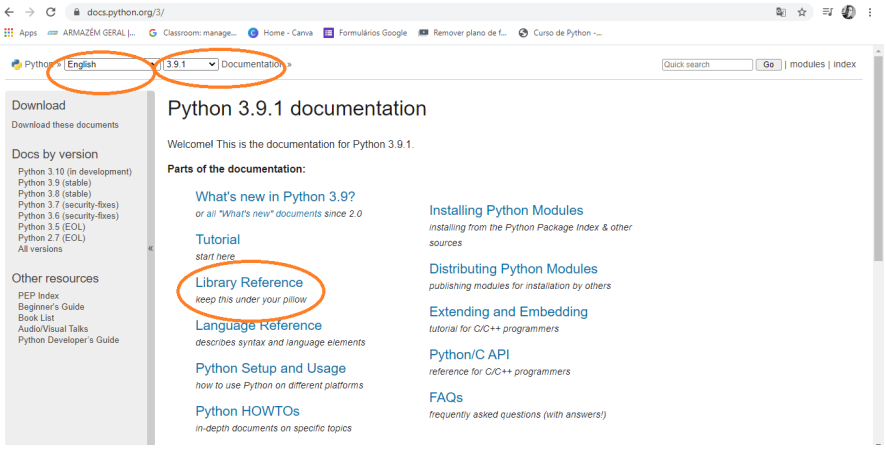
Quando usamos ‘from’ vamos importar somente as funcionalidades(função - método) que especificarmos de uma biblioteca (classe).

Para conhecer as classes do python que podemos importar para nossos programas, entre no site python.org e siga os seguintes passos:

1) Entre no menu “Docs”, mostrado a seguir:

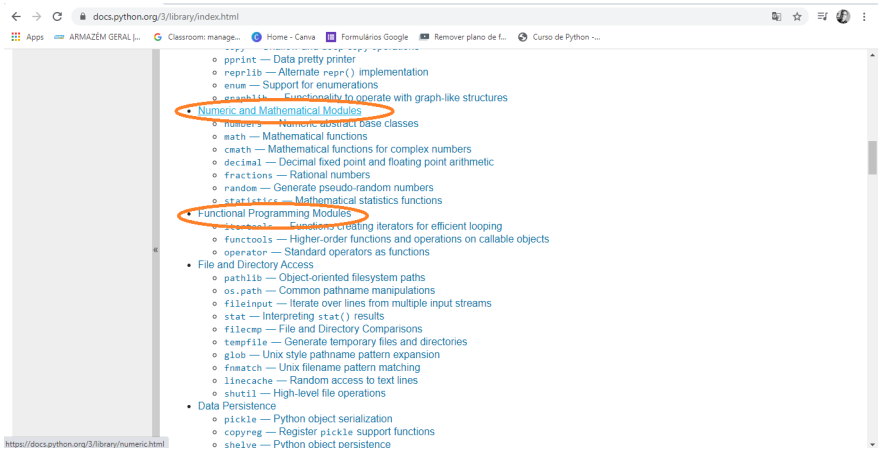


2) Surgirá a seguinte página:



Escolha o idioma que você preferir, a versão do Python que está usando e clique em ***Library Reference*****,** conforme figura acima

3) Uma nova página será carregada. Procure pelo tópico que informa sobre módulos, como exemplo abaixo:

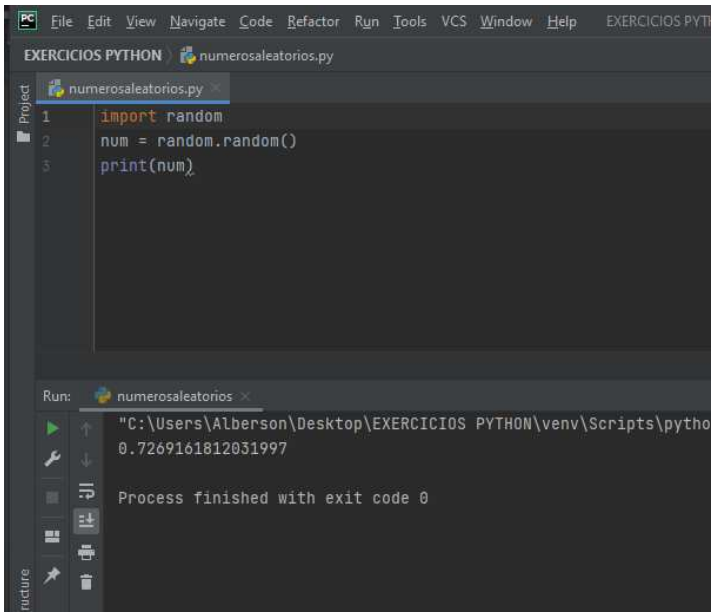


Veja que que nas bibliotecas numéricas e matemáticas acima existe por exemplo a CLASSE random que pode ser importada no seu programa.

ATENÇÃO: PODE ACONTECER DO SEU PYTHON INSTALADO NÃO CONTER ESTA BIBLIOTECA, VOCÊ PODERÁ BAIXÁ-LA FACILMENTE.

EXEMPLO 4:

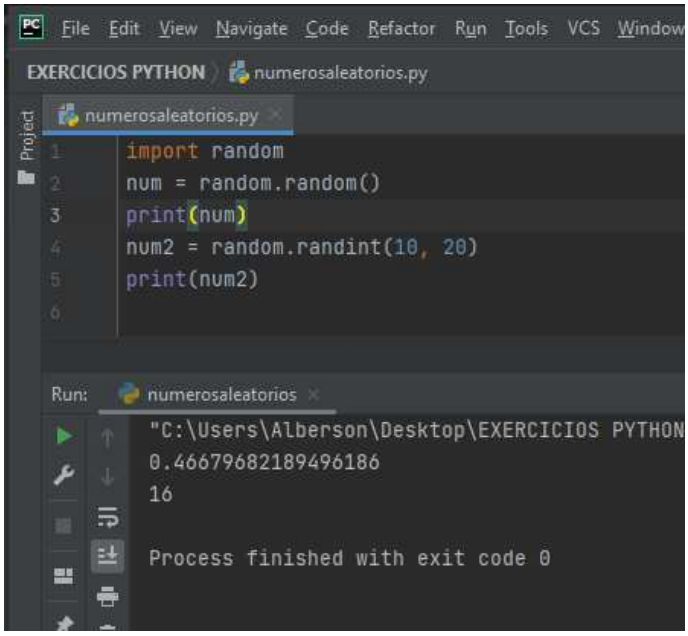
Observe o exemplo abaixo onde estamos gerando números aleatórios entre 0.0 e 1.0 flutuante.



Usamos a classe ***random***. Em seguida, na linha 2, usamos o método ***random()*** desta classe para armazenar em ***num*** o número gerado automaticamente pelo computador. Por fim, o número gerado foi mostrado.

LEMBRE-SE QUE O COMPUTADOR SORTEARÁ UM NÚMERO QUALQUER E MOSTRARÁ AO USUÁRIO, desta forma o número exibido no seu computador, quando fizer teste, certamente será diferente do exibido acima.

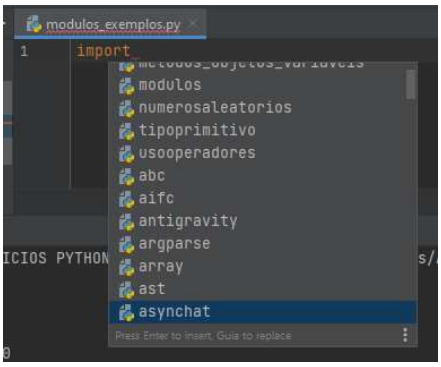
EXEMPLO 5:



Neste exemplo estamos usando o método ***randint*** para gerar um número inteiro entre limites informados como parâmetros. Neste exemplo foi gerado um número entre 10 e 20, no caso foi exibido pelo computador o número ***16.***

EXEMPLO 6:

Outra forma de sabermos quais as classes podemos importar é pressionando simultaneamente as teclas ***<ctrl><espaço>***, após a digitação do comando ***import***, veja o que acontecerá



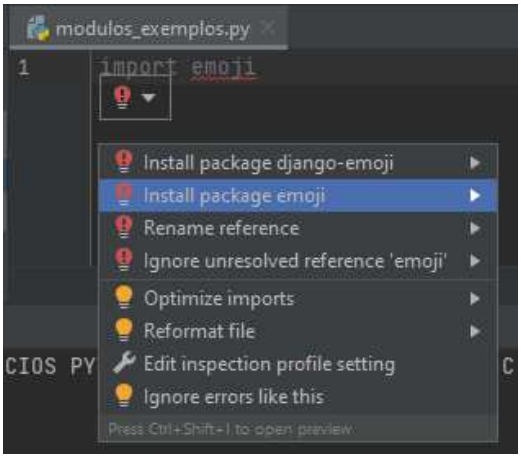
Será exibida uma lista com todas as bibliotecas disponíveis para importação, no programa que você for desenvolver.

EXEMPLO 7:

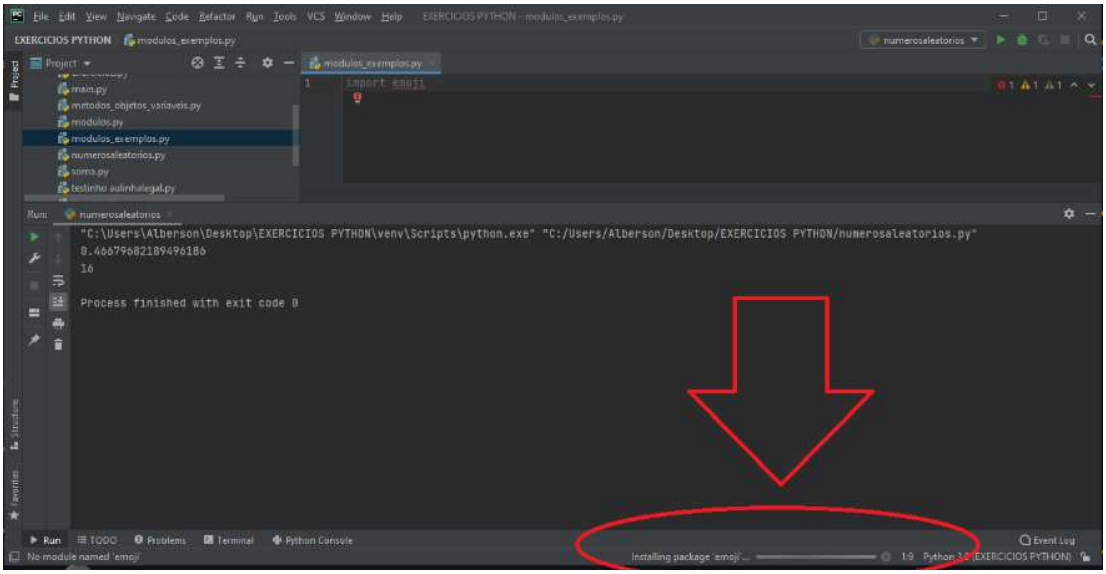
É IMPORTANDE MENCIONAR!!!
VOCÊ DESENVOLVEDOR PODE USAR BIBLIOTECAS CRIADAS POR OUTROS DESENVOLVEDORES DA COMUNIDADE PYTHON.
BASTA BAIXÁ-LAS DENTRO DO SEU PROGRAMA.
Lógico que você deverá estudar as sintaxes de uso das bibliotecas desenvolvidas por terceiros.

16.1 INSTALAÇÃO DE PACOTES DE BIBLIOTECAS EXTERNAS DA COMUNIDADE DE DESENVOLVIMENTO

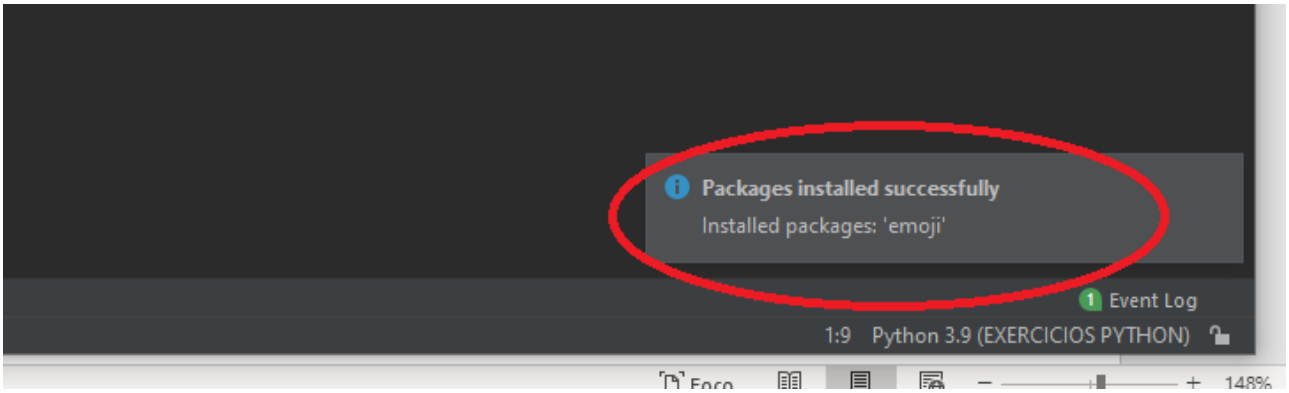
Usando o PyCharm, podemos importar bibliotecas desenvolvidas pela comunidade. Basta digitar o nome da biblioteca (POR EXEMPLO VAMOS USAR a biblioteca ***emoji***). Siga abaixo:



Veja que a classe ***emoji*** não está disponível no meu Python. Usando o PYCharm, basta clicar na lâmpada vermelha aparece quando você clica na palavra digitada ***emoji***. Logo depois clique em ***"install package emoji"***. Repare no rodapé da IDE do Pycharm quando a instalação for iniciada:

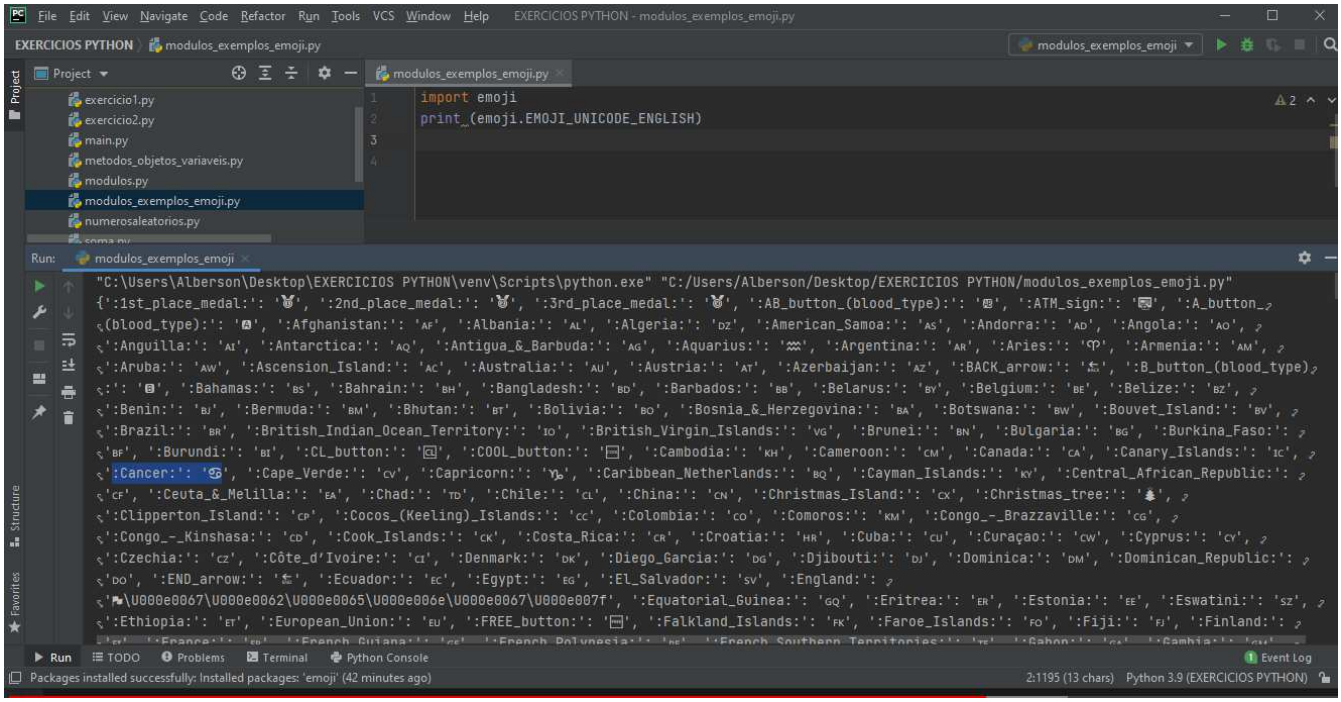


Quando o pacote for instalado você verá a seguinte mensagem:

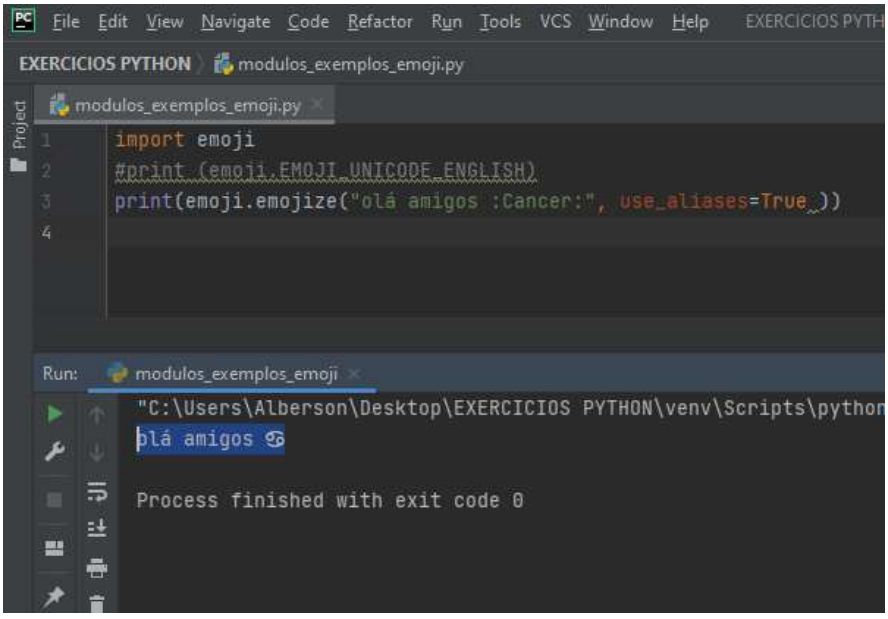


16.2 VERIFICANDO BIBLIOTECAS QUE SEU PYTHON POSSUI

Agora basta criar um programa para sabermos os emojis que podemos usar!!!! Veja exemplo:



Agora vamos usar a seguinte programação:



Observe na figura:

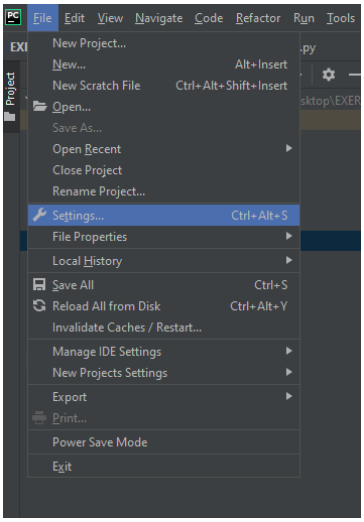
Linha 2: deixamos como comentário para não mostrar todos os emojis que temos do idioma inglês

Linha 3: Usamos o emoji '***:Cancer:***'. É IMPORTANTE VERIFICAR QUE O VALOR VERDADEIRO, ATRIBUÍDO A ***use_aliases=True*** está com a letra **T (maiúsculo)**

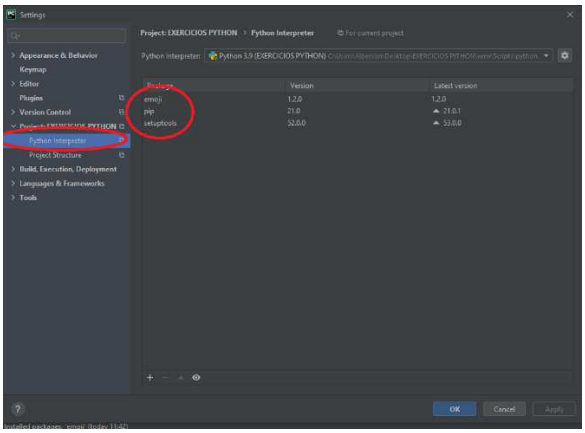
Após a execução que foi impresso o emoji Câncer na mensagem para o usuário.

Para verificar quais as BIBLIOTECAS EXTERNAS (criadas por outros programadores da comunidade) você tem instaladas no seu PYCHARM, siga os seguintes passos:

Passo 1)

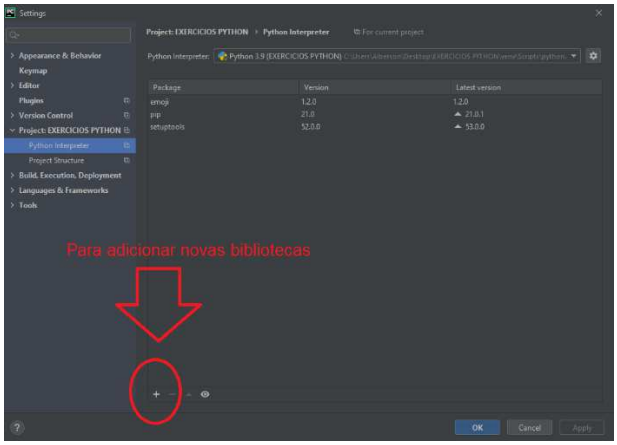


Passo 2)

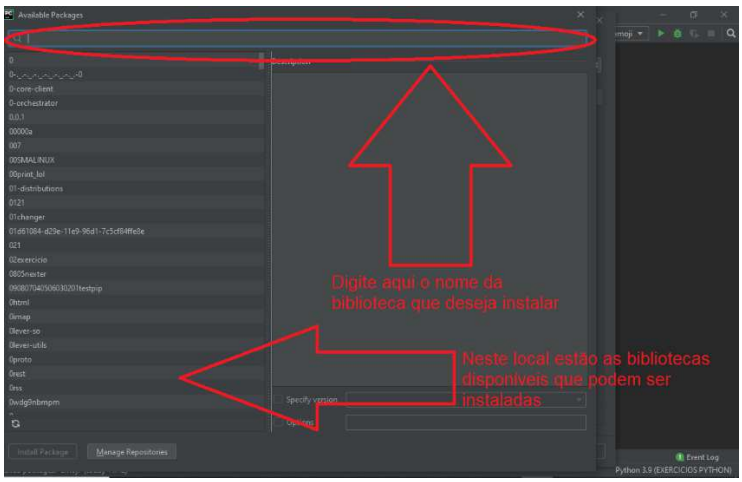


Ao clicar em ***Python Interpreter***, serão mostradas todas as bibliotecas externas contidas no seu computador. Perceba que na figura acima, tenho instalada a biblioteca ***emoji***, que instalei anteriormente.

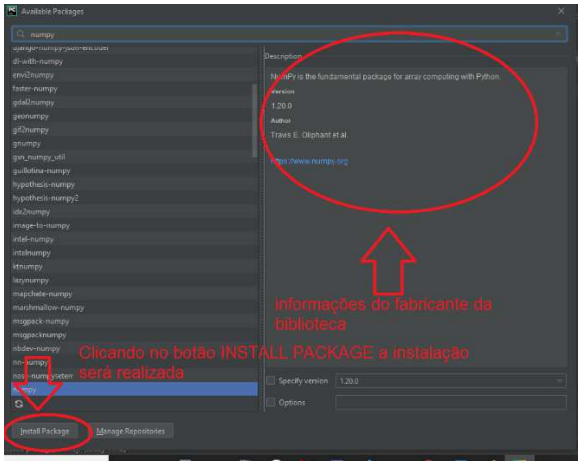
Você também poderá instalar por esta área suas bibliotecas, basta clicar no botão **+** :



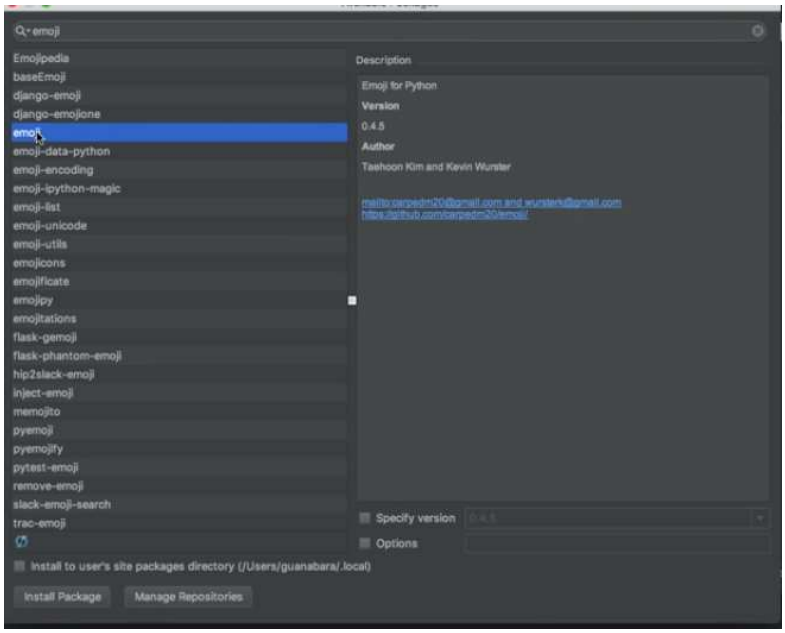
Surgirá a próxima tela para que você procure pela biblioteca que deseja instalar:



Após escolher a biblioteca desejada para ser instalada, basta clicar no seguinte botão:



Veja que poderíamos ter instalado a biblioteca ***EMOJI*** por este local:



CONCLUINDO!!!!

Pelo menu *File* opção *settings...*, no submenu que surgir escolha *Python Interpreter,* você poderá gerenciar todos os pacotes de bibliotecas, instalando e desinstalando bibliotecas.

Qualquer dúvida acesse o seguinte vídeo do youtube para tirar suas dúvidas sobre instalação de novas bibliotecas no seu PYCharm: Como instalar uma biblioteca o PYCHARM em https://www.youtube.com/watch?v=xlVaNMdBP24

16.3 INSTALANDO BIBLIOTECAS NO PYTHON COM COMANDO pip (LINHA DE COMANDO NO PROMPT DO WINDOWS)

Outra forma de instalar bibliotecas pode ser feito pelo **pip**. Para isso basta entrar no **prompt de comandos do Windows** e digitar o seguinte comando, por exemplo:

C:\>pip install <nome_da_biblioteca>

Observação:
<nome_da_biblioteca> refere-se ao nome da biblioteca a ser instalada.

ATENÇÃO: ALGUMAS BIBLIOTECAS NÃO ESTÃO DISPONÍVEIS NO REPOSITÓRIO DO pypi (Python Package Index – índices de pacotes do python). Caso isso aconteça você precisará buscar por referências de instalação destas na internet.

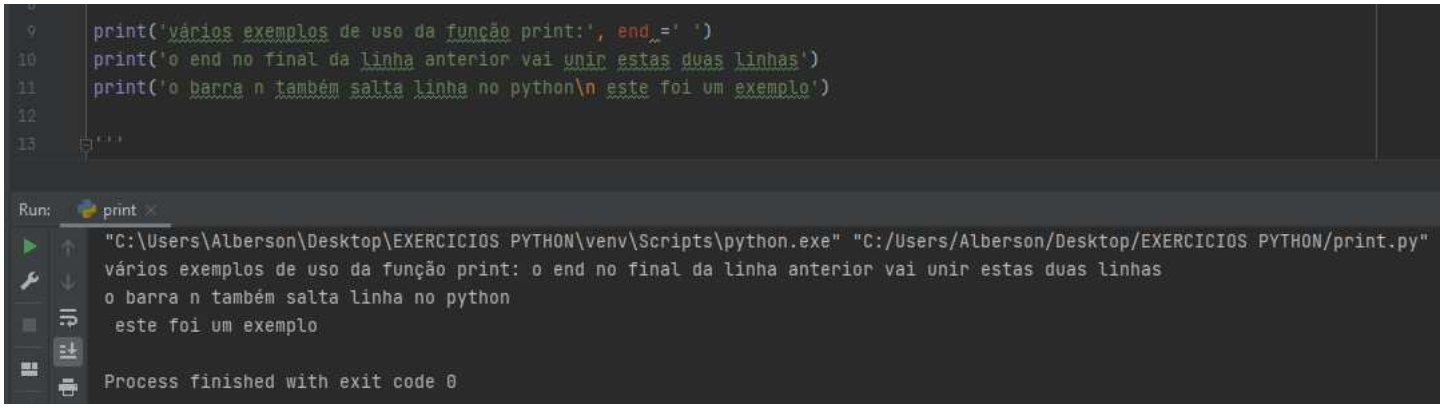
Exemplo:

```
C:\Users\fabio>pip install scipy
Collecting scipy
  Downloading https://files.pythonhosted.org/packages/db/9e/465a416eb04114e3722b17b0f4fa5235bab8a
76961de51db0e5850183fb1/scipy-1.4.1-cp38-cp38-win32.whl (27.9MB)
     |                                | 27.9MB 726kB/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\fabio\appdata\local\programs\python\pyth
on38-32\lib\site-packages (from scipy) (1.18.0)
Installing collected packages: scipy
Successfully installed scipy-1.4.1
```

Podem assistir também algumas dicas neste canal do youtube: https://www.youtube.com/watch?v=3qyDy9S2TzM (01/02/2021) sobre instalações de bibliotecas usando comando *pip*.

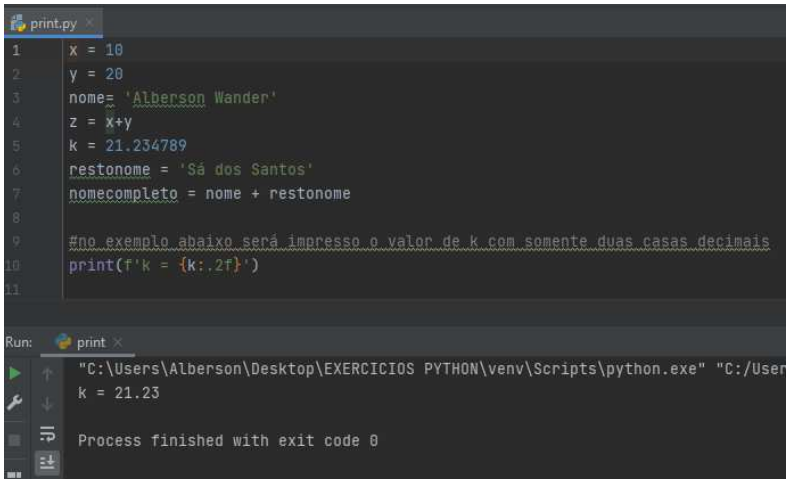
17. FUNÇÃO print()

Esta função do python é utilizada basicamente para imprimir mensagens e/ou valores de variáveis (objetos) na tela para o usuário. Deve ser escrito em letras minúsculas como nos exemplos a seguir.



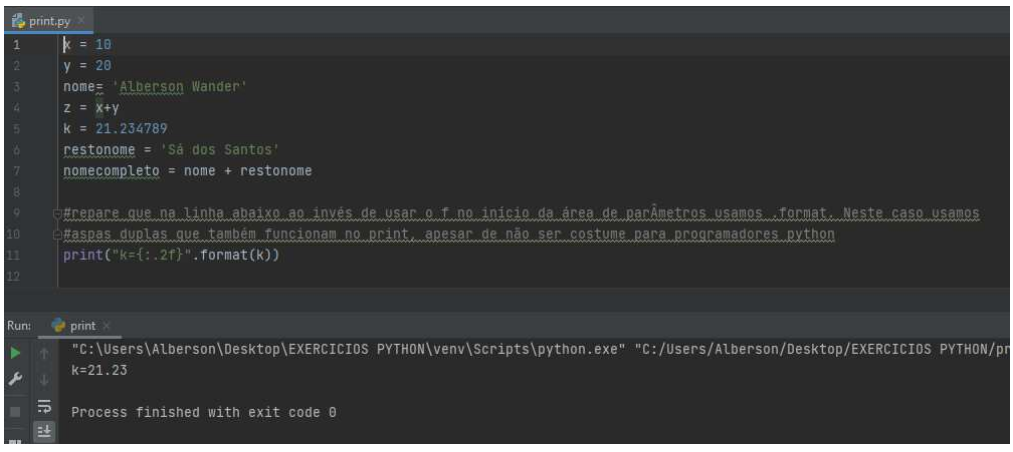
Observem o código acima:

a) Na linha 9 o parâmetro ***end=‘ ’***, após a vírgula provoca a união de resultados dos prints da linha 9 e 10
b) O ***\n*** escrito n alinha 11 imprimiu o restante da mensagem em outra linha



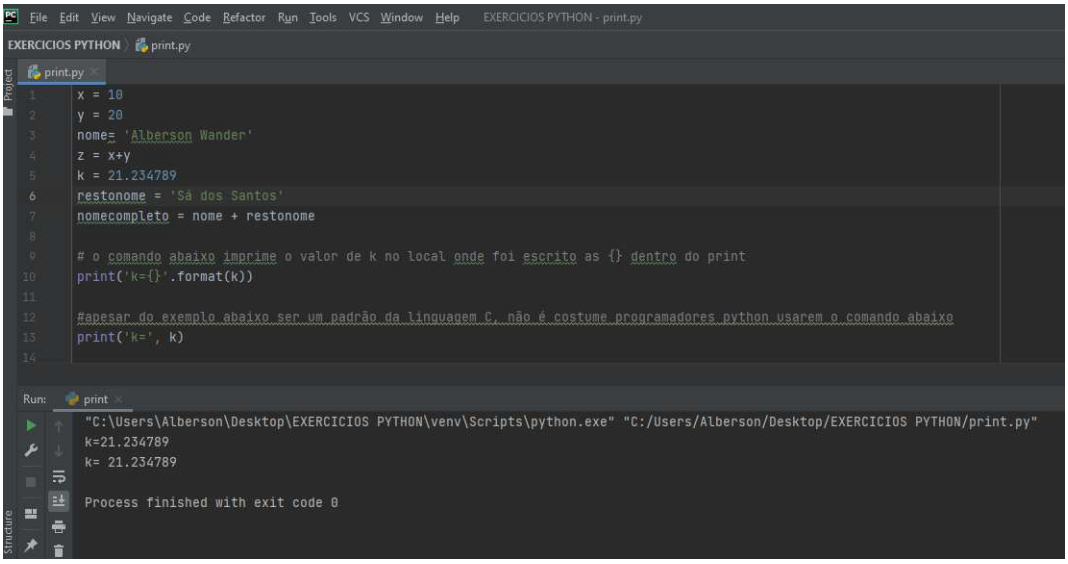
Observe o código acima:

a) A letra ***f*** escrita inicialmente dentro dos parênteses do print indica que será feita uma impressão do valor armazenado em ***k*** e uma formatação de tal valor para exibir somente duas casas decimais.



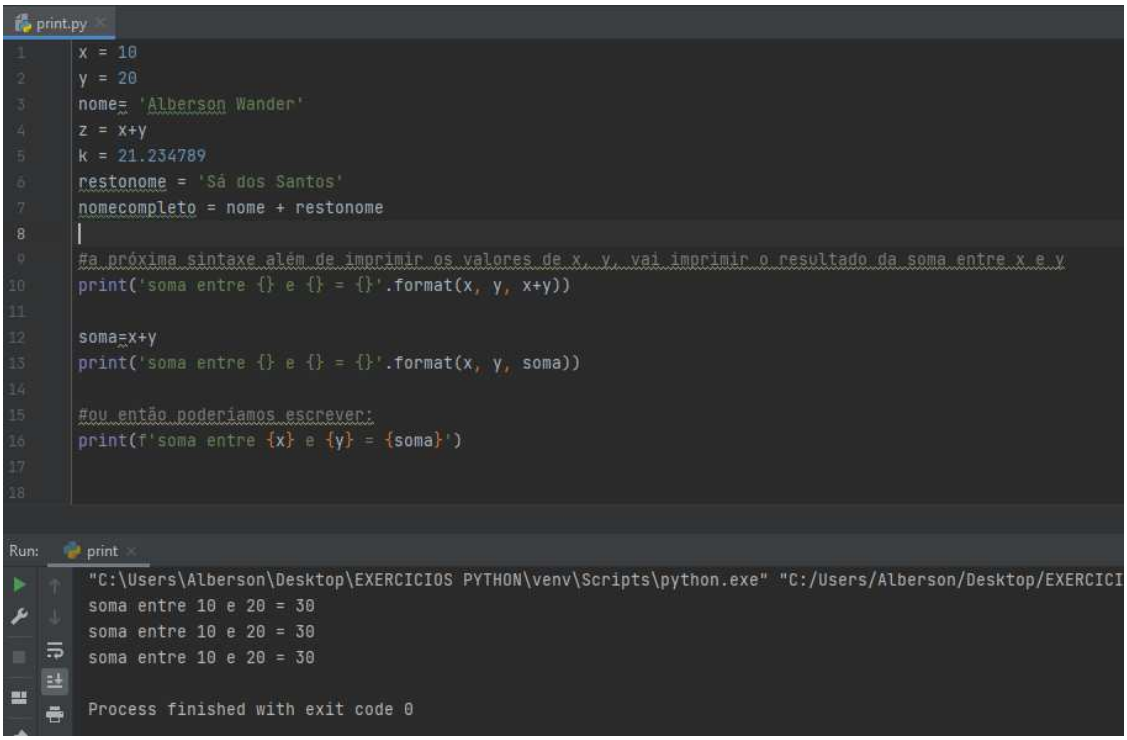
Observe o código acima:

a) Este programa faz a mesma coisa que o anterior, porém o nome da variável a ser impressa é indica fora das ‘’ (aspas), mas o python sabe que as chaves dentro do printf indicam o local de impressão do dado de ***k.***



Observem o código acima:

a) As duas linhas de print fazem a mesma coisa, porém a que possui o parâmetro ***format(k)*** é a mais usada pelos programadores python. Inclusive a segunda linha de impressão da linha 13 tende a não ser mais usada em versões futuras da linguagem.



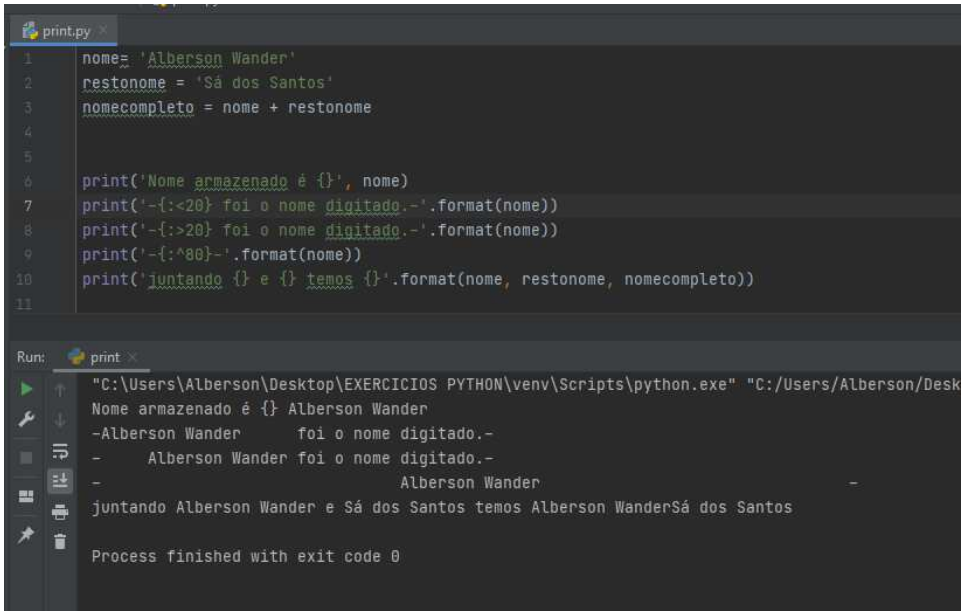
Observem as formas de impressão de soma exibidas acima:

a) Na linha 10 realizamos a soma e já imprimimos o resultado dentro do próprio ***format()***. Isso se faz quando não há necessidade de guardar na memória o resultado desta soma.

b) Na linha 13 o comando print está entendendo que nas 3 ***{}*** escritas receberão os dados de ***x, y, soma***, respectivamente.

c) Na linha 16 o comando print dispensa o parâmetro ***format()*** e já indica a posição de impressão das variáveis diretamente no local da mensagem delimitado com as ***{}***.

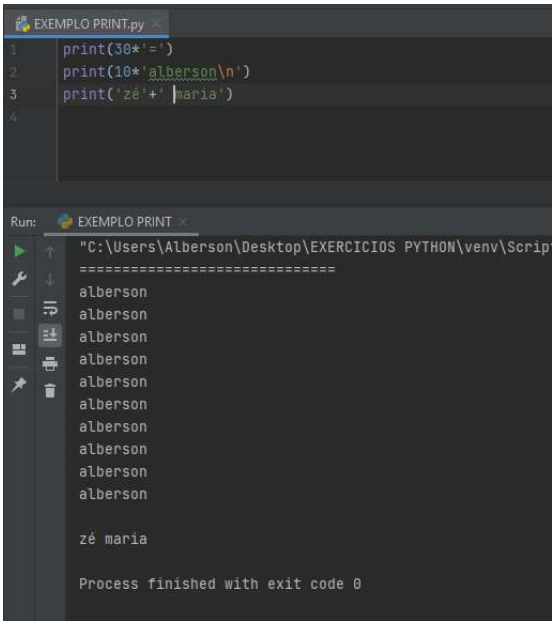
d) Note que os resultados impressos foram exatamente iguais.



Observem o código acima:

a) O comando print da linha 6 imprimiu para o usuário as {}, antes do ***nome*** armazenado, pois não foi usado o ***.format().*** Caso opte por esta sintaxe **não use as {}**.
b) No print da linha 7 o {:<20} indica que o ***nome*** será impresso neste local e alinhado a esquerda, em 20 espaços, observe a impressão.
c) No print da linha 8 o {:>20} indica o que o ***nome*** será impresso neste local e alinhado a direita, em 20 espaços, observe a impressão.
d) O comando print da linha 9 indica que o ***nome*** será impresso centralizado entre 80 espaços definidos.
e) O comando print da linha 10 indica a posição de impressão das variáveis ***nome, restonome, nomecompleto,*** respectivamente.
f) Os print() das linhas 7, 8 e 9 usei o '-'(traço) para mostrar os alinhamentos definidos dentro das chaves.

CURIOSIDADES QUE PODEM SER FEITAS COM print():



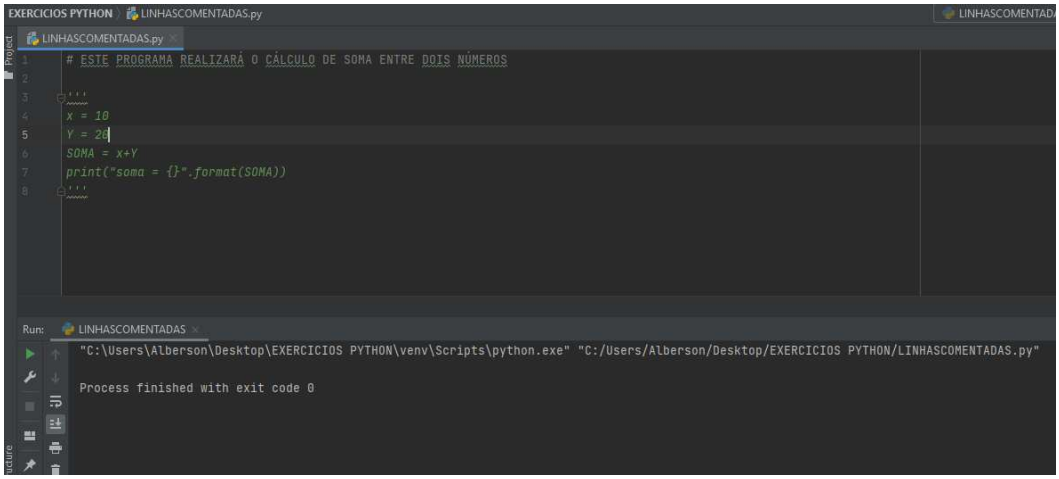
Percebeu os resultados apresentados?

a) Usei o print para imprimir 20 vezes o sinal ***"="***.
b) Imprimi 10 vezes o nome ***"Alberson"*** usando ***\n.***
c) Imprimi duas mensagens alfabéticas juntando-as ***"zé" + "maria",*** por isso foi impresso ***"zé maria"***

18. CRIANDO LINHAS OU BLOCOS DE COMENTÁRIOS DE COMENTÁRIOS

Para comentarmos uma linha qualquer na área de edição do seu programa python basta usarmos o **#** no início desta. Entretanto, há situações que precisamos comentar um bloco de linhas na área de edição do programa. Para esta ituação usaremos no início do bloco ‘‘‘ (três aspas) e no final mais ’’’ (três aspas).

Veja:



Observem que no exemplo acima apesar de escrever linhas de código válidas para python, nada foi executado, pois todas as linhas estão comentadas porque estão entre as ***‘‘‘ ’’’***. A linha 1 deste programa acima também é um comentário, pois inicia com ***#.***

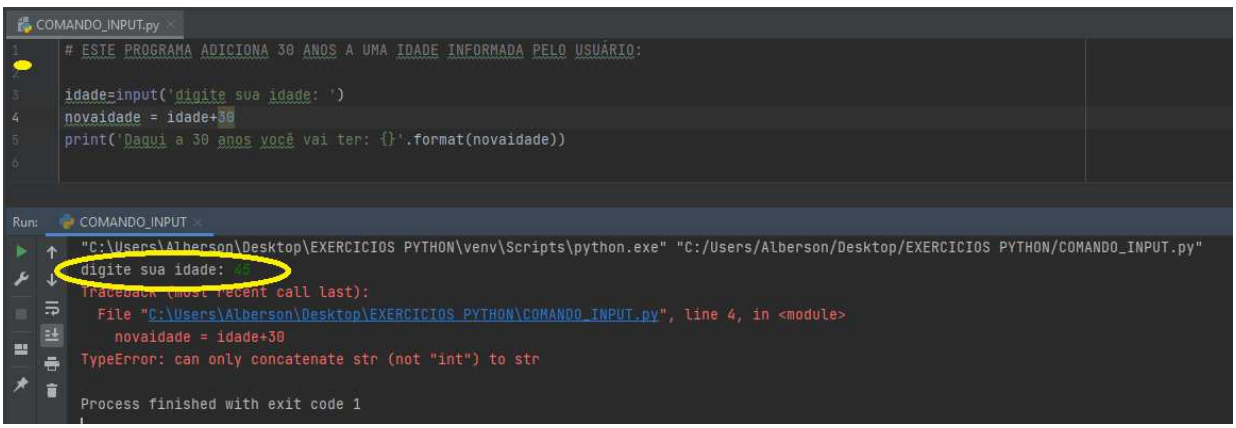
19. FUNÇÃO input()

Esta função é utilizada para solicitar ao usuário de um programa para digitar dados via teclado.

MUITO IMPORTANTE: OS DADOS DIGITADOS PELOS USUÁRIOS SEMPRE SERÃO DO TIPO STRING, MESMO QUE ELE IMAGINE ESTAR DIGITANDO UM NÚMERO, O PYTHON ENTENDERÁ APENAS COMO SENDO UMA SEQUÊNCIA DE CARACTERES.

Diferente de algumas linguagens, nesta função indicaremos como parâmetro uma mensagem ao usuário do que deve ser digitado, para não precisarmos usar na linha anterior a função ***print()***.

Vamos aos exemplos:



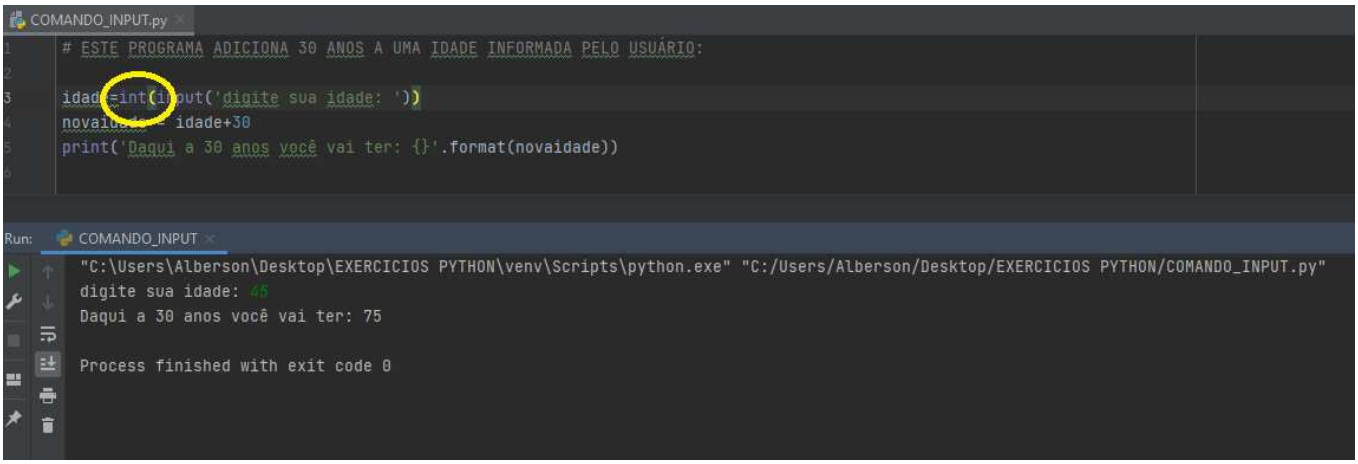
Vamos as observações sobre o teste feito:

a) Inicialmente é solicitado ao a digitação de uma idade**.**

b) **Porém, ocorreu erro na linha 4, que indica que não é possível realizar cálculo com o valor do tipo string. Lembrem-se que a variável *idade* é do tipo string por receber dados de uma função input. Não podemos de forma alguma somar uma string a um valor numérico qualquer.**

c) Para solução deste problema temos que transformar o valor retornado pela função input para o tipo de dado que estamos desejando trabalhar, neste caso para inteiro.

Vamos concertar o erro ocasionado:



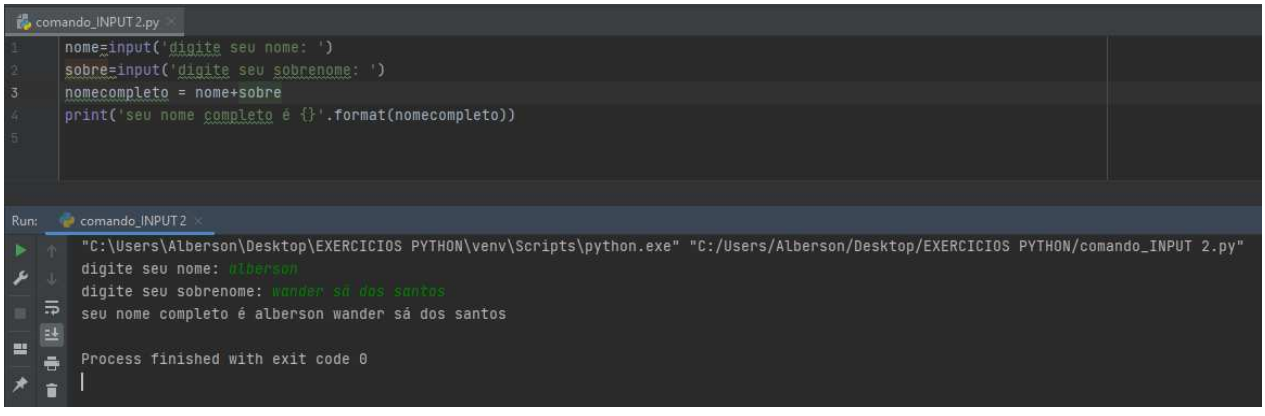
Repare agora:

a) Na linha 3, logo após o comando de atribuição ***(=)*** utilizei a função ***int()*** para transformar o valor digitado pelo usuário para um dado do tipo inteiro.

b) O programa rodou perfeitamente, pois a idade digitada ***45*** foi transformada para inteiro, e na linha 4 do programa somou-se 30 anos. O resultado foi armazenado em uma variável ***novaidade*** que assume neste momento o tipo inteiro também.

c) Desta forma a função ***print()*** mostrou a idade do usuário após 30 anos, neste caso ***75.***

Vamos a outro exemplo interessante:



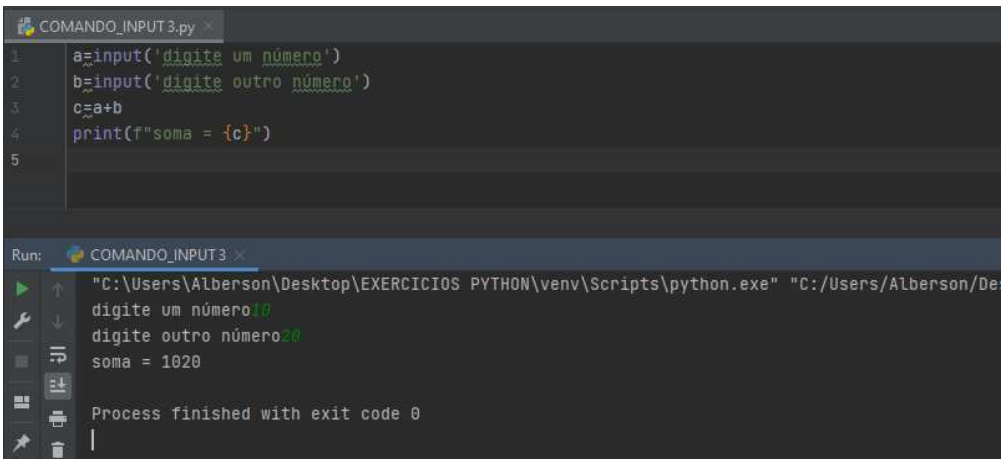
Repare alguns pontos do script :

a) Neste programa dois dados do ***tipo string*** foram guardados nas variáveis ***nome*** e ***sobre.***

b) Na linha 3, repare que o operador **+** tem a funcionalidade de juntar os dados armazenados nas variáveis ***nome*** e ***sobre,*** gerando assim um conteúdo para a variável ***nomecompleto.***

c) No exemplo de execução acima digitei ***"alberson "*** que foi guardado em ***nome*** e ***"wander sá dos santos"*** que foi guardado em ***sobre.*** Quando juntadas pelo operador **+** foi impresso pela linha 4 o conteúdo da variável ***nomecompleto,*** neste caso ***"alberson wander sá dos santos".***

Exemplo de execução de programa que não gera erro de sintaxe, mas pode gerar dúvidas futuras:



Analise os comandos acima:

a) O usuário digitou ***10*** e ***20***. Porém a soma dos números deveria gerar 30, mas o resultado exposto foi ***1020***. Isso aconteceu porque não foram usados os tipos primitivos (ou conversores), para que o python entenda que estaríamos tentando realizar a soma entre valores numéricos e não junção de strings, que foi o que aconteceu. Lembro que em alguns momentos da sua vida de profissional este exemplo poderá ser utilizado para concatenação (junção) de strings.

20- OPERADORES RELACIONAIS

OPERADOR RELACIONAL	REFERENTE A:
==	Igual a
!=	Diferente
>=	Maior ou igual
>	Maior que
<	Menor que
<=	Maior ou igual

20- OPERADORES LÓGICOS

OPERADOR	O QUE SIGNIFICA?
And	.E.
Or	.OU.
not	.NAO.
in	Igual a qualquer um
not in	Diferente de qualquer um
is	É
not is	Não É

21- ESTRUTURAS DE CONDIÇÕES SIMPLES E COMPOSTAS– COMANDO if

Comando de seleção if:

Só para relembrar usamos este comando sempre que desejarmos determinar o fluxo de execução de um programa, diante de resultado de testes condicionais que serão realizados.

Quando um teste condicional é realizado e o resultado for .verdadeiro., executa-se todos os comandos abaixo do comando *if.* Caso seja falso, serão executados os comando escritos no *else* ou *elif.*

Sintaxe 1:

```
if condicao==true:
        bloco_comandos_verdadeiro
```

IMPORTANTE:
O BLOCO DE COMANDOS VERDADEIRO SERÁ EXECUTADO SEMPRE QUE O TESTE DE CONDIÇÃO RESULTAR EM VERDADEIRO, DEVE SER OBRIGATORIAMENTE ESCRITO INDENTANDO O BLOCO DE COMANDOS, OU SEJA, USE A TECLA TAB PARA DESCOLOCAR E ALINHAR TODO O BLOCO PARA DIREITA, CONFOME SINTAXE ACIMA. NESTE CASO NÃO TEMOS CASO CONTRÁRIO, OU SEJA, *else OU elif;*

Sintaxe 2:

```
if condicao==true:
        bloco_comandos_verdadeiro
else:
        bloco_comandos_falso
```

IMPORTANTE:
REPARE ACIMA QUE O BLOCO DE COMANDOS FALSO, TAMBÉM DEVE OBRIGATORIAMENTE SER ESCRITO INDENTADO, OU SEJA, USANDO A TECLA TAB TEMOS QUE DESLOCÁ-LO TODO MAIS PARA DIREITA, CONFOME DEMONSTRA A SINTAXE 2, ACIMA;

Sintaxe 3: ANINHAMENTOS DE TESTES CONDICIONAIS

```
if condicao1==true:
        bloco_comandos_verdadeiro
else:
        if condicao2==true:
                bloco_comandos_verdadeiro_condicao2
        else:
                .......
                If condicaoN == true:
                        bloco_comandos_verdadeiro_ultimacondicao
```

IMPORTANTE:
QUANDO TEMOS ANINHAMENTOS DE if, REPARE QUE SE TORNA NECESSÁRIO USAR O TAB DIVERSAS VEZES, DESLOCANDO OS TESTES CONDICIONAIS E TABÉM SEUS BLOCOS DE COMANDOS INTERNOS(V). OBSERVE QUE OS ALINHAMENTOS TENDEM A CRESCER PARA A DIREITA A MEDIDA QUE TESTAMOS UM NÚMERO MAIOR DE CONDIÇÕES

Sintaxe 4:

```
if condicao1==true:
        bloco_comandos_verdadeiro_da_condicao1
elif condicao2 == true:
        bloco_comandos_verdadeiro_da_condicao2
elif condicao3 == true:
        bloco_comandos_verdadeiro_da_condicao3
.....
elif condicaoN == true:
        bloco_comandos_verdadeiro_da_condicao3
.....
else:
        ultimo_bloco_comandos_caso_todas_acima_falso
```

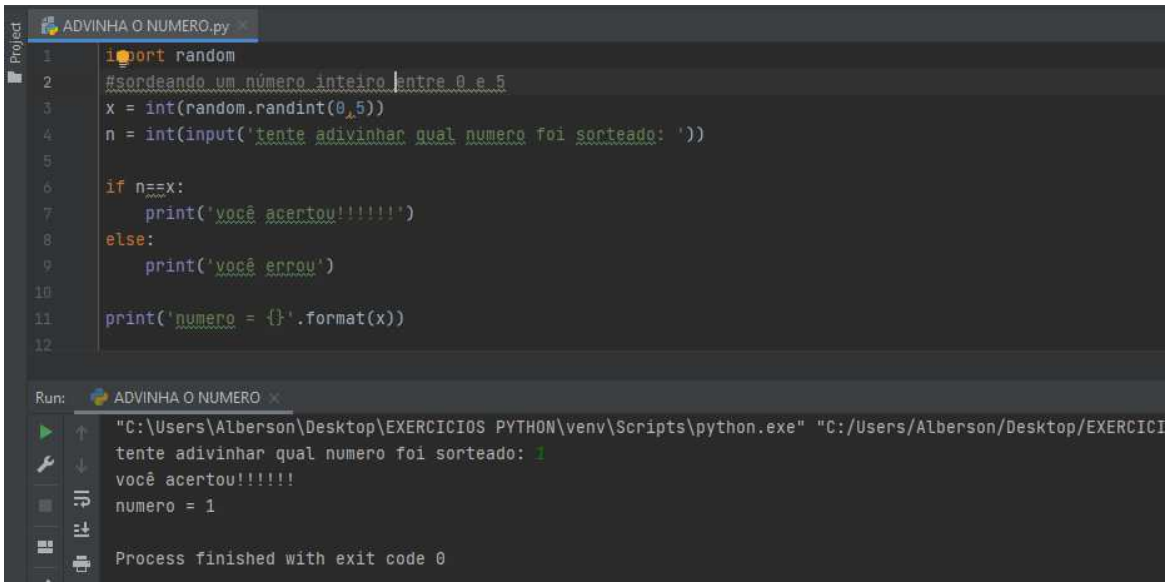
IMPORTANTE:

REPARE ACIMA QUE COM A CLÁUSULA *elif* NÃO NECESSITAMOS INDENTAR ESTA LINHA DE COMANDO, BASTANDO INDENTAR SOMENTE O BLOCO DE COMANDOS INTERNO DE CADA CONDIÇÃO, QUANDO ESTE RESULTAR EM VERDADEIRO. DESTA FORMA PERCEBA QUE AS INDENTAÇÕES NÃO CRESCEM ABUNDANTEMENTE PARA DIREITA.

SE DESEJAR NÃO TESTAR MAIS NENHUMA CONDIÇÃO PODE USAR O COMANDO *else*, DE TAL FORMA QUE UM ÚLTIMO BLOCO DE COMANDOS SEJA EXECUTADO DIANTE DO RESULTADO FALSO DE TODAS AS CONDIÇÕES ANTERIORES TESTADAS.

VEJAMOS ALGUNS EXEMPLOS:

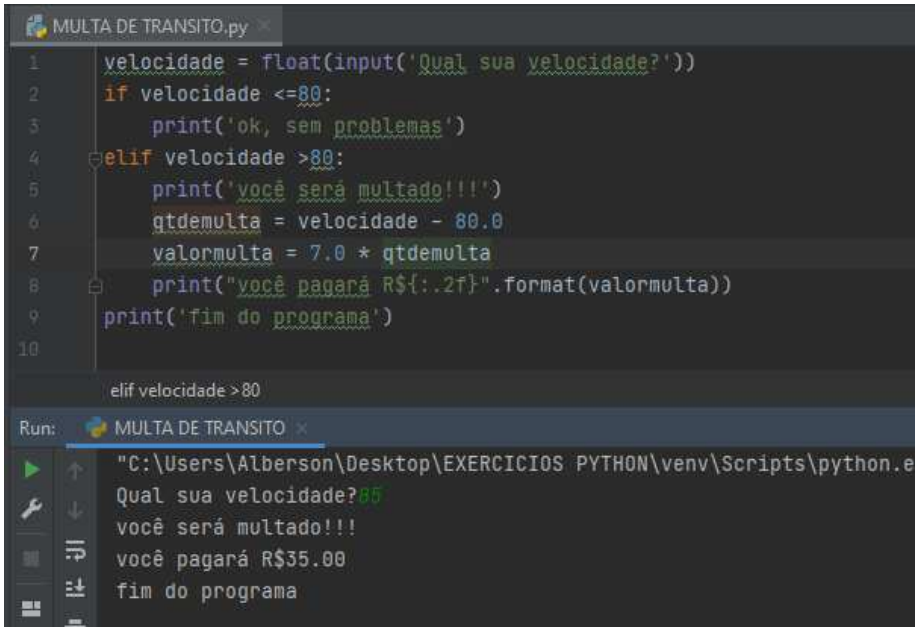
i) Programa para verificar se o usuário acerta qual número foi o sorteado pelo computador:



Vejamos o programa acima:

a) Importei o pacote ***random***.
b) A variável x receberá um número aleatório sorteado pelo computador entre 0 e 5
c) Um número é solicitado ao usuário.
d) A partir da linha 6 do programa mostrado realiza-se o teste condicional COMPOSTO, para verificar se o usuário acertou ou não o número sorteado.

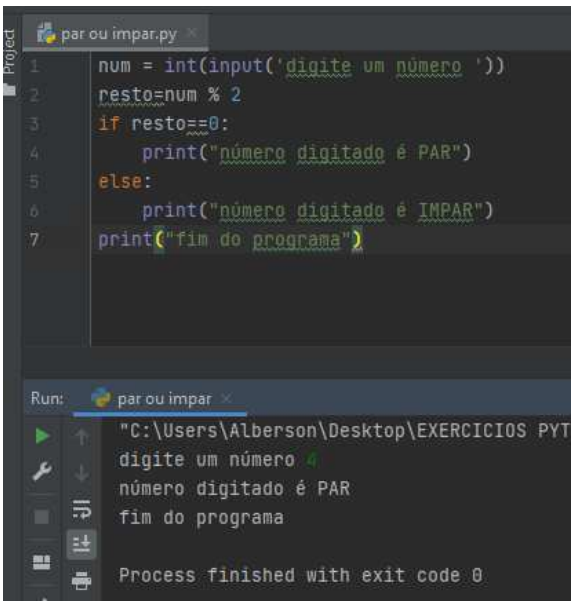
ii) O programa abaixo multa um infrator de trânsito que ultrapassou 80 km por hora. O valor por cada km acima de 80 é R$7,00. Veja a solução abaixo:



Veja:

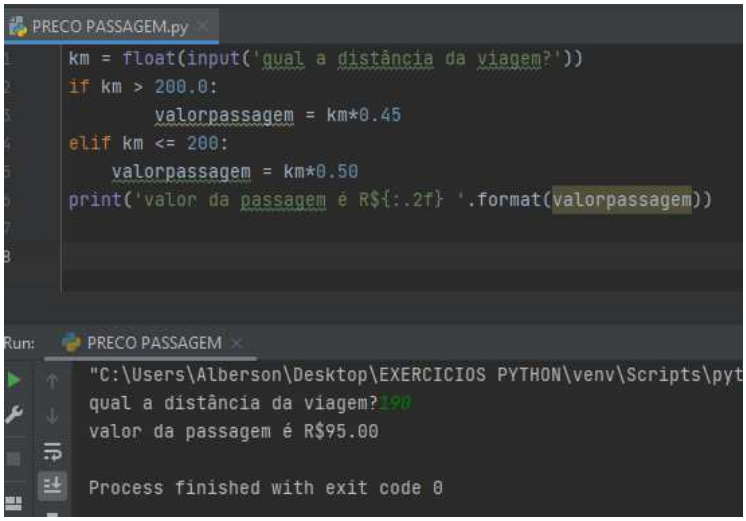
a) A cláusula elif foi usada já com o teste condicional no programa mostrado
b) Observe que o print, escrito na linha 9 está alinhado mais à esquerda, isso significa que *"fim do programa"* irá aparecer sempre, ou seja, se o usuário for ou não multado, pois não está dentro de nenhum dos blocos de comandos do teste

iii) O programa abaixo verifica se um número digitado pelo usuário é par ou ímpar:

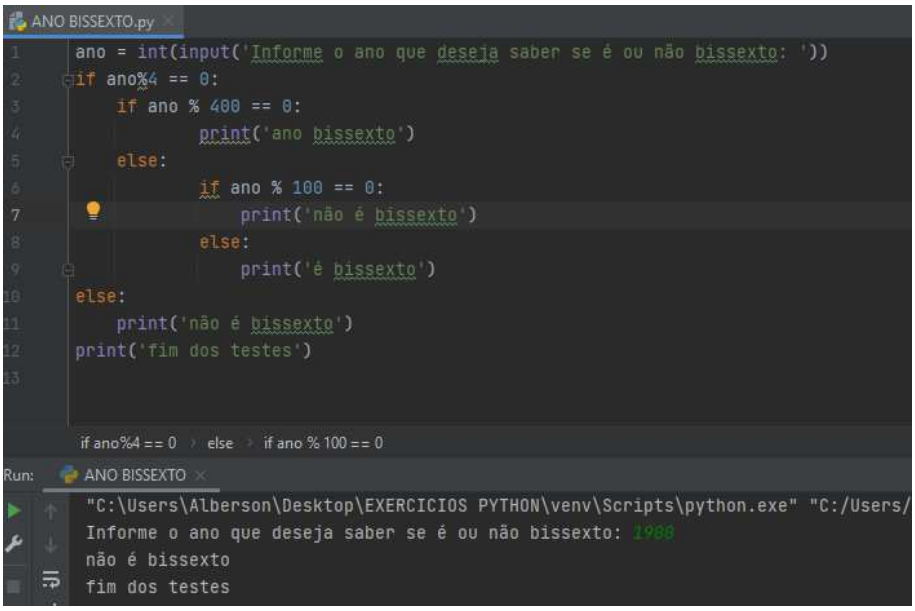


Observe:
Neste programa não foi necessário testar uma 2ª condição, visto que só existem 2 resultados possíveis, ou seja, um número é par ou ímpar. Desta forma testei se era par, senão fosse, logicamente seria ímpar.

iv) O próximo exemplo vai calcular o preço de uma passagem de ônibus mediante a quilometragem da viagem informada. Se km maior que 200km o valor da viagem será quantidade de km multiplicado por R$0.45,00. Caso contrário, o valor será o km multiplicado por R$0,50.

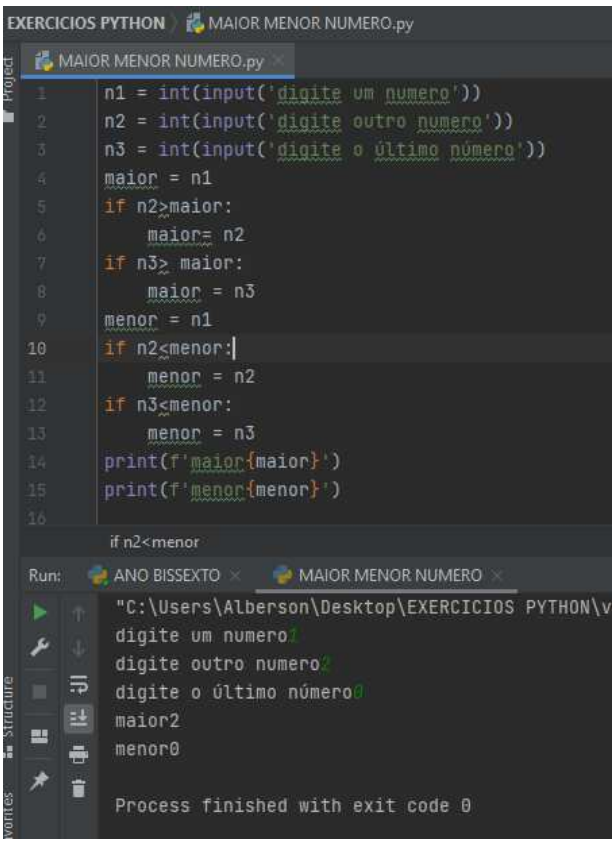


v) O próximo programa é um clássico para quem aprende testes condicionais. Ele mostra ao usuário se um ano é ou não bissexto:



Observação:
Para um ano ser bissexto tem que ser divisível por 4 e por 400 e não pode ser divisível por 100

vi) O programa a seguir verifica qual o maior e menor número entre 3 números digitados pelo usuário



22- USO DOS OPERADORES LÓGICOS

Vamos relembrar tabela verdade usando operadores lógicos para casos de testes de 2 condições. Vejamos:

Tabela verdade operador **and**:

Condicao1	Condicao2	Resultado teste
V	V	V
V	F	F
F	V	F
F	F	F

A tabela acima poderia ser exemplificada da seguinte forma:

```
if condicao1 and condicao2:
        <comandos caso resultado verdadeiro>
else:
        <comandos caso resultado falso>
```

Tabela verdade operador **or**:

Condicao1	Condicao2	Resultado teste
V	V	V
V	F	V
F	V	V
F	F	F

A tabela acima poderia ser exemplificada da seguinte forma:

```
if condicao1 or condicao2:
        <comandos caso resultado verdadeiro>
else:
        <comandos caso resultado falso>
```

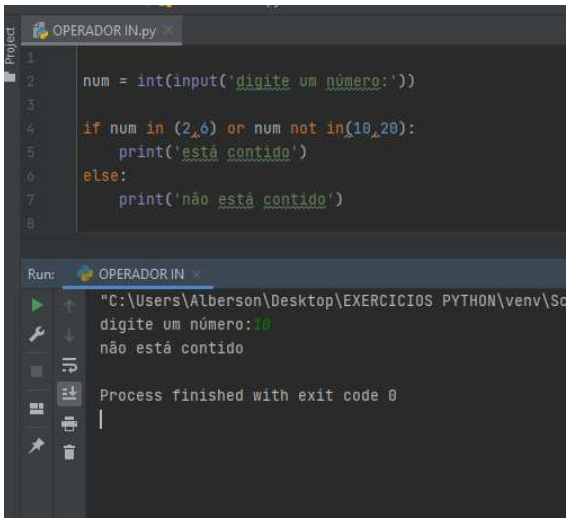
Tabela verdade operador **not**:

Condicao1	not(Condicao1)
V	F
F	V

A tabela acima poderia ser exemplificada da seguinte forma:

```
if not(condicao1):
        <comandos caso resultado verdadeiro>
else:
        <comandos caso resultado falso>
```

O operador in verifica se o operando a sua esquerda está contido numa lista de valores. Da mesma forma o not in verifica o contrário, ou seja, se não está contido na lista a esquerda. Veja exemplo:



Repare:
Este programa verifica se um número digitado **é o** 2 ou 6 OU **NÃO É** 10 ou 20.

Vamos avaliar!!!

Operador or , Sabendo que foi digitado número 10, perceba os resultados dos testes na tabela abaixo:

`num in (2,6)` está testando se foi 2 ou 6?	`num not in(10,20)` está testando se NÃO FOI DIGITADO o 10 ou NÃO FOI DIGITADO o 20?	Resultado teste
F, pois não foi digitado o número 2 nem o 6	F, pois FOI DIGITADO O 10	F por isso foi impressa a mensagem "não está contido"

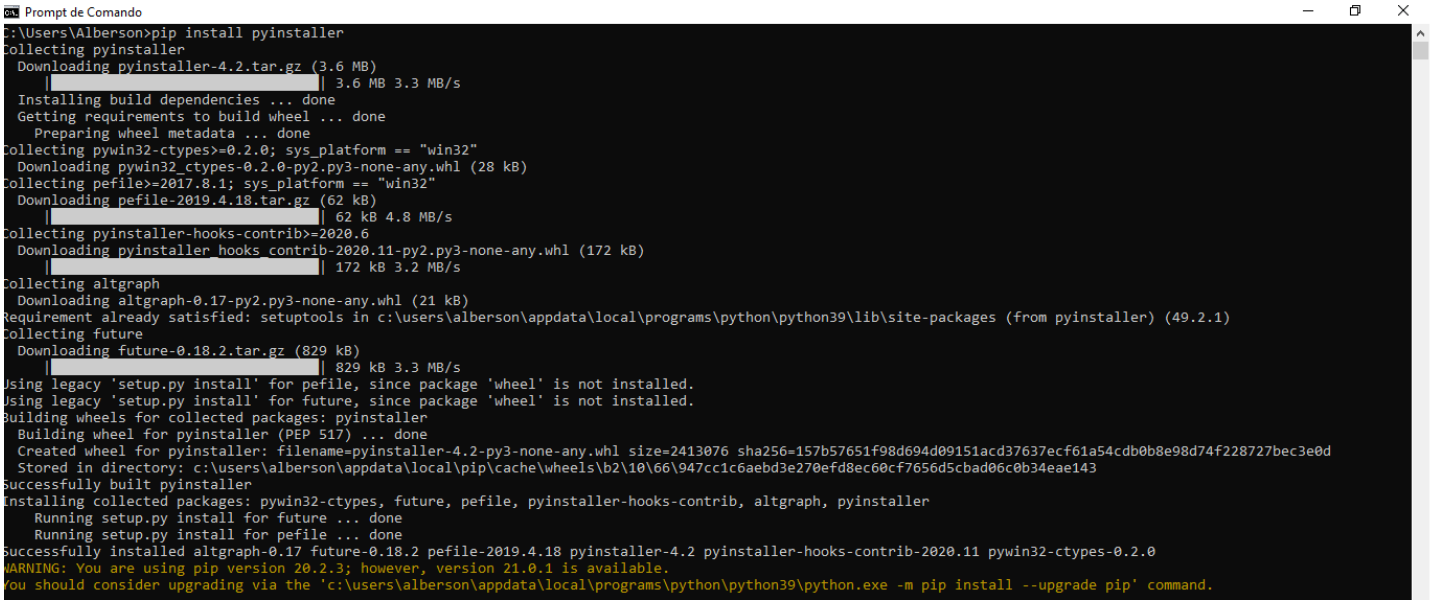
23- CRIANDO ARQUIVOS EXECUTÁVEIS NO PYTHON (pyinstaller)

Sempre que desenvolvemos qualquer programa muitas vezes necessitamos executá-los em equipamentos que NÃO TEMOS O PYTHON instalado. Daí a necessidade de gerar arquivos executáveis dos programas que criamos.
Devemos instalar com o pip um gerador de arquivos executáveis. Para isso entre nas janela de linhas de comandos no Windows e escreva a seguinte linha de comando:

```
C:\>pip install pyinstaller
```

Vale lembrar que para o comando acima funcionar você deve ter o python instalado no Windows seu computador.

Ao pressionar <enter> após a digitação do comando no prompt do Windows, você verá a instalação iniciar, conforme figura a seguir:

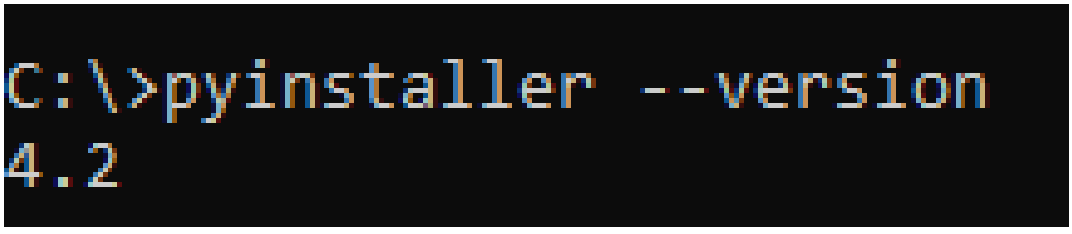


Repare acima, nas últimas linhas em amarelo, que poderá o ocorrer do seu pip estar desatualizado, porém, isso não impediu a instalação do pyinstaller

Para verificar a versão do pyinstaller que foi instalada basta digitar no prompt de comando do Windows o seguinte comando:

```
C:\>pyinstaller --version
```

Veja figura abaixo, quando você pressionar <enter>:



Para gerar o arquivo executável de um programa criado em python (.py) basta usar a seguinte sintaxe no prompt do windows:

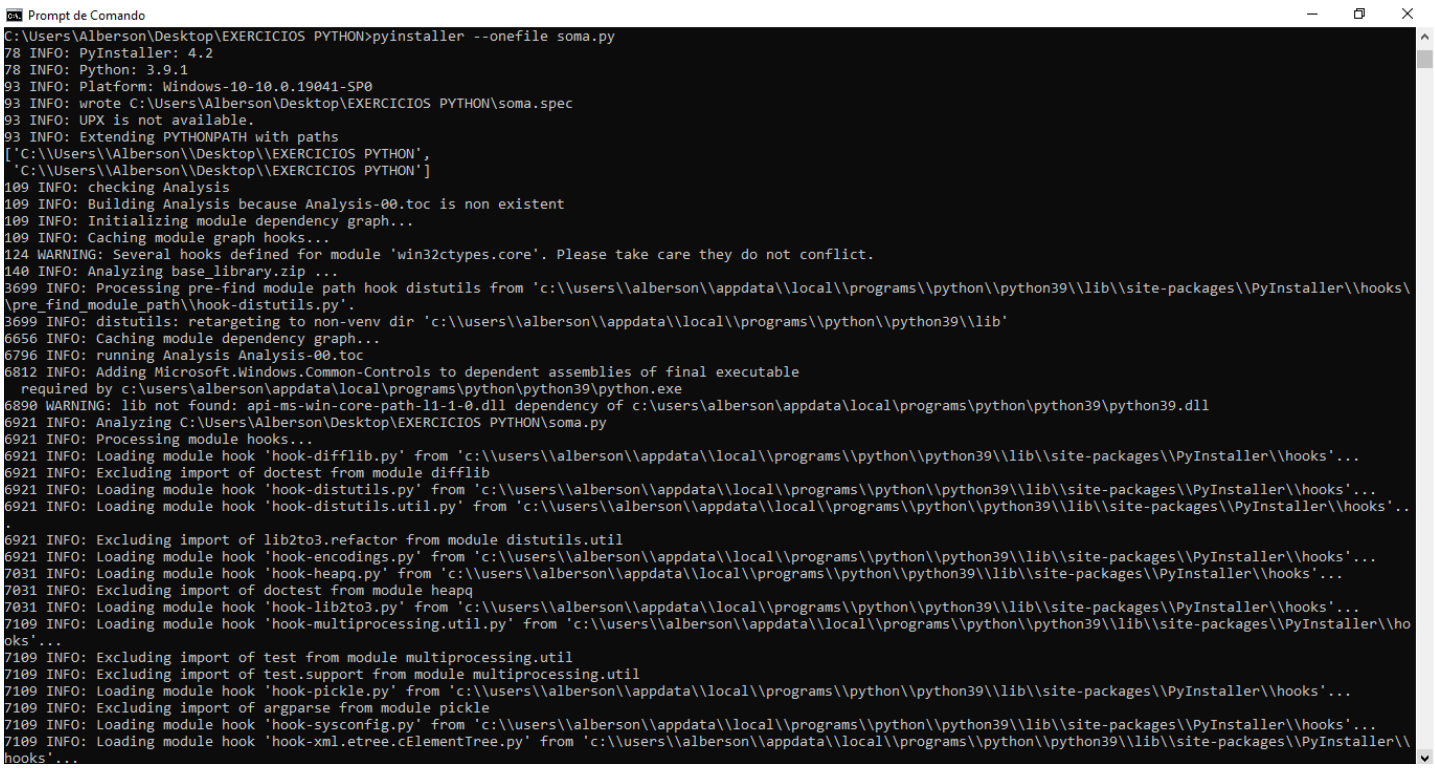
C:\(caminho_do_programa_py)\>pyinstaller –onefile <nome_programa.py>

Como exemplo vou usar um programa chamado soma.py que criei numa pasta no meu computador, veja:

C:\Users\Alberson\Desktop\EXERCICIOS PYTHON>pyinstaller --onefile soma.py

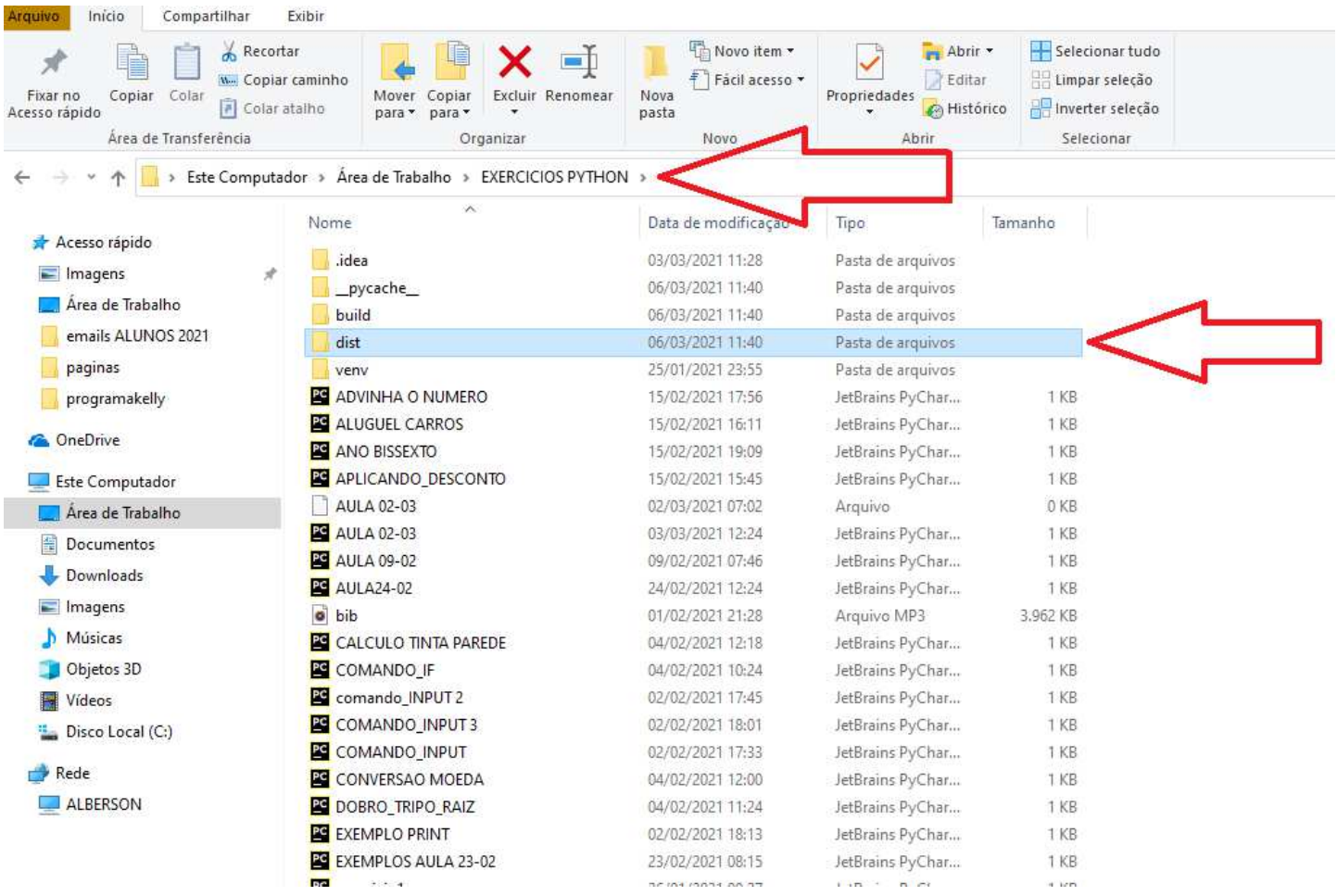
Ao pressionar <enter> você acompanhará a geração do arquivo .EXE. Após a geração será criada uma pasta “DIST” na pasta “EXERCÍCIOS PYTHON” que estou usando como exemplo. Na pasta “DIST” está o arquivo soma.exe, o qual você usará em computadores que não possuem o python instalado. Simples assim !!!

Veja como será exibido o processo de geração do programa .EXE a seguir:

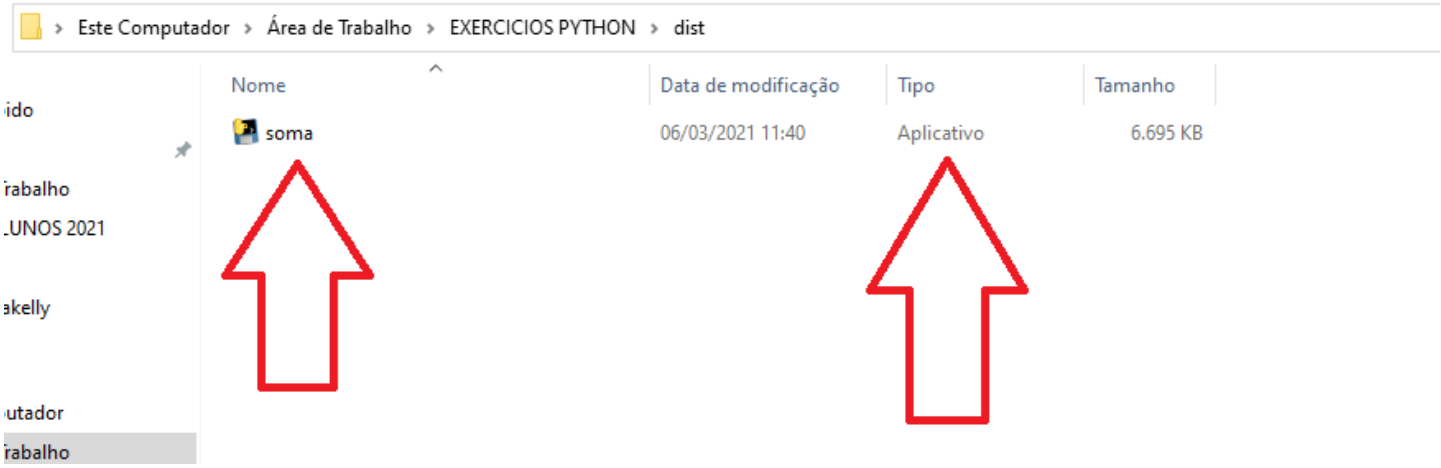


```
7109 INFO: Loading module hook 'hook-xml.py' from 'c:\\users\\alberson\\appdata\\local\\programs\\python\\python39\\lib\\site-packages\\PyInstaller\\hooks'...
7199 INFO: Loading module hook 'hook-_tkinter.py' from 'c:\\users\\alberson\\appdata\\local\\programs\\python\\python39\\lib\\site-packages\\PyInstaller\\hooks'...
7362 INFO: checking Tree
7362 INFO: Building Tree because Tree-00.toc is non existent
7362 INFO: Building Tree Tree-00.toc
7503 INFO: checking Tree
7503 INFO: Building Tree because Tree-01.toc is non existent
7503 INFO: Building Tree Tree-01.toc
7596 INFO: checking Tree
7612 INFO: Building Tree because Tree-02.toc is non existent
7612 INFO: Building Tree Tree-02.toc
7643 INFO: Looking for ctypes DLLs
7690 INFO: Analyzing run-time hooks ...
7690 INFO: Including run-time hook 'c:\\users\\alberson\\appdata\\local\\programs\\python\\python39\\lib\\site-packages\\PyInstaller\\hooks\\rthooks\\pyi_rth_multiprocessing.py'
7706 INFO: Looking for dynamic libraries
8128 INFO: Looking for eggs
8128 INFO: Using Python library c:\users\alberson\appdata\local\programs\python\python39\python39.dll
8128 INFO: Found binding redirects:
[]
8147 INFO: Warnings written to C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\build\soma\warn-soma.txt
8212 INFO: Graph cross-reference written to C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\build\soma\xref-soma.html
8252 INFO: checking PYZ
8252 INFO: Building PYZ because PYZ-00.toc is non existent
8255 INFO: Building PYZ (ZlibArchive) C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\build\soma\PYZ-00.pyz
8912 INFO: Building PYZ (ZlibArchive) C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\build\soma\PYZ-00.pyz completed successfully.
8928 INFO: checking PKG
8928 INFO: Building PKG because PKG-00.toc is non existent
8928 INFO: Building PKG (CArchive) PKG-00.pkg
10778 INFO: Building PKG (CArchive) PKG-00.pkg completed successfully.
10778 INFO: Bootloader c:\users\alberson\appdata\local\programs\python\python39\lib\site-packages\PyInstaller\bootloader\Windows-64bit\run.exe
10778 INFO: checking EXE
10778 INFO: Building EXE because EXE-00.toc is non existent
10778 INFO: Building EXE from EXE-00.toc
10809 INFO: Copying icons from ['c:\\users\\alberson\\appdata\\local\\programs\\python\\python39\\lib\\site-packages\\PyInstaller\\bootloader\\images\\icon-console.ico']
10809 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
10809 INFO: Writing RT_ICON 1 resource with 3752 bytes
10809 INFO: Writing RT_ICON 2 resource with 2216 bytes
10809 INFO: Writing RT_ICON 3 resource with 1384 bytes
10809 INFO: Writing RT_ICON 4 resource with 37019 bytes
10809 INFO: Writing RT_ICON 5 resource with 9640 bytes
10809 INFO: Writing RT_ICON 6 resource with 4264 bytes
10809 INFO: Writing RT_ICON 7 resource with 1128 bytes
10825 INFO: Updating manifest in C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\build\soma\run.exe.fv7y9ke0
10825 INFO: Updating resource type 24 name 1 language 0
10825 INFO: Appending archive to EXE C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\dist\soma.exe
10840 INFO: Building EXE from EXE-00.toc completed successfully.
```

Após o processo de gerar o arquivo executável ser finalizado vamos abrir a pasta “EXERCICIOS PYTHON” e verificaremos que foi criada a pasta “DIST”, onde se encontrará o arquivo “soma.exe”, veja a seguir:



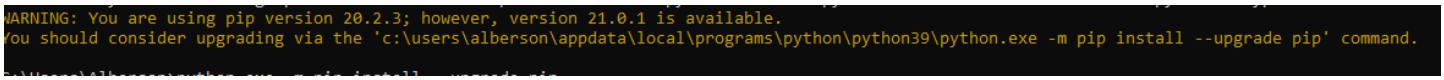
Abrindo a pasta “dist”, verifique os arquivos executáveis que você possui, conforme exemplo a seguir:



Se você clicar no arquivo acima o programa será executado como qualquer programa Windows. SIMPLES ASSIM!!!

24 – ATUALIZAÇÃO DO PIP

Como sabemos, as ferramentas de desenvolvimento do python sofrem atualizações constantes. Não diferente disso, o próprio pip pode solicitar atualização, principalmente após a instalação de bibliotecas. No item anterior desta apostila, "23 - CRIANDO ARQUIVOS EXECUTÁVEIS NO PYTHON (pyinstaller)", verificamos que após instalar o pyinstaller foi exibida a solicitação de atualização do pip, veja:



AVISO: você está usando a versão 20.2.3 do pip; no entanto, a versão 21.0.1 está disponível.
Você deve considerar a atualização por meio do comando
'c: \users\alberson\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip'

Vale dizer que a mensagem acima referencia a minha área de usuário e uma pasta "Alberson", a qual não existirá no seu computador. Sendo assim, para evitar qualquer dúvida e erros, digite a seguinte linha de comando no prompt do Windows:

```
C:\>python.exe -m pip install --upgrade pip
```

Quando pressionar <enter> será executada a atualização conforme exemplo a seguir:

```
C:\Users\Alberson>python.exe -m pip install --upgrade pip
Collecting pip
  Downloading pip-21.0.1-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 2.2 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
Successfully installed pip-21.0.1
```

Desta forma você estará com seu pip atualizado, conforme o EXEMPLO acima.

25- ESTRUTURAS DE REPETIÇÕES

Os comandos de estruturas de repetições do python são:

- for
- while

Veremos a seguir as sintaxes usadas para o uso dos dois comandos:

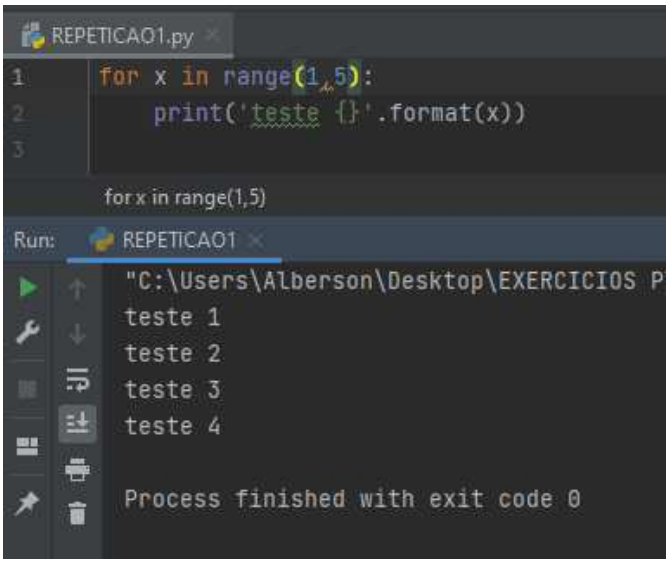
25.1 – COMANDO for

Sintaxe 1 :

for <variável_de_controle> in range(<valorinicial>, <valorfinal>, <passo>):

    <comandos que serão repetidos>

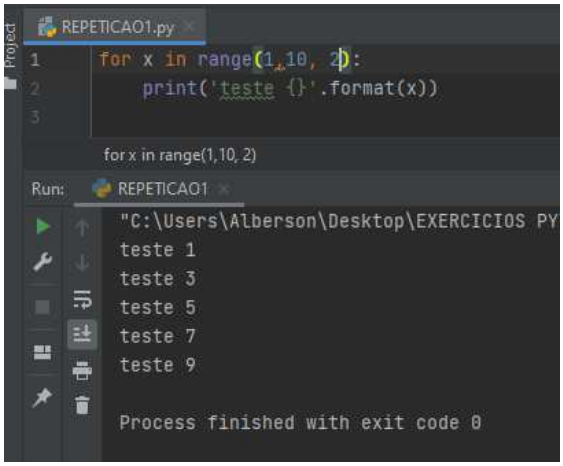
Exemplo 1:



Repare no exemplo acima:

a) Em python, o valor final indicado nos parênteses do range indica que a variável de controle NÃO RECEBERÁ ESTE VALOR, assim sendo o laço será repetido de 1 a 4.
b) Neste exemplo NÃO USAMOS o passo, assumindo desta forma que o passo será automático de 1

Exemplo 2:



Repare neste exemplo:

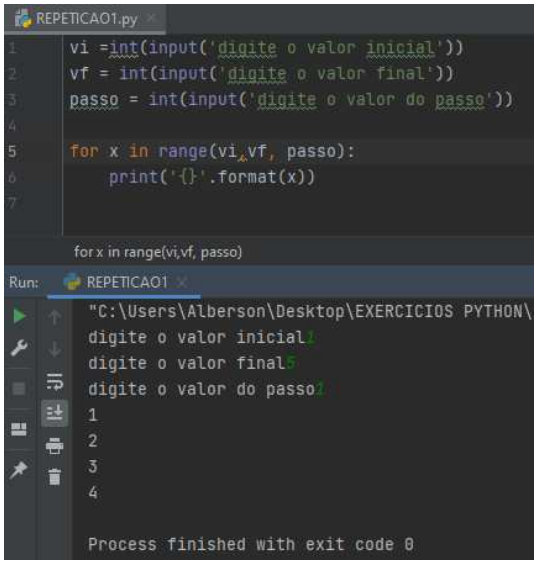
a) Neste exemplo usamos o parâmetro do passo no range(), o que provocou a contagem de 1 até 10 de 2 em 2.
b) Lembrem-se que neste caso o laço vai de fato de 1 até 9, pois no 10 ele deve ser finalizado, não entrando no laço de repetição.

Exemplo 3:

Repare neste exemplo:

a) Neste exemplo foram usadas 3 variáveis para determinar valores de início, fim e o passo do laço de repetição.
b) Observe também, que a contagem do laço se deu de forma regressiva, ou seja, de 10 até 2, pois o vf sendo 1 o laço é finalizado.
c) O “passo” do exemplo acima é -1 porque a contagem é regressiva, ou seja , de 10 até 2;

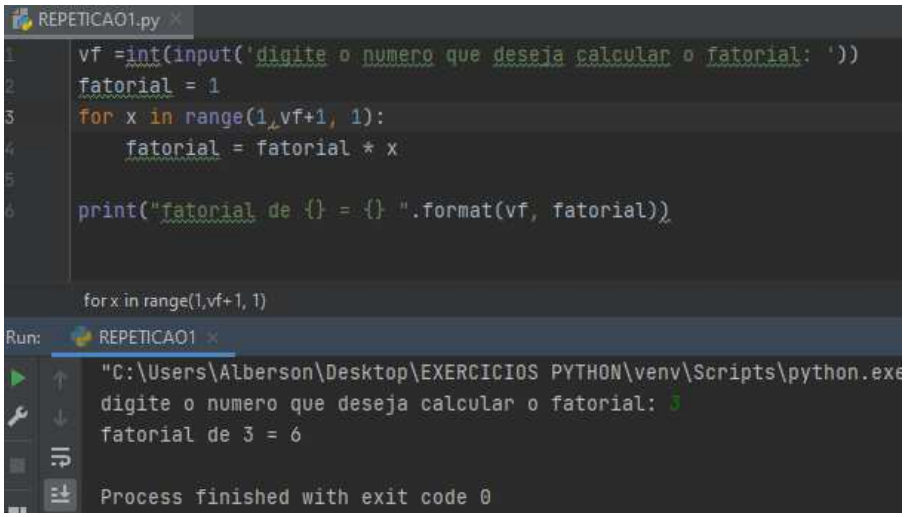
Exemplo 4:



Repare neste exemplo:

a) O usuário informou o valor inicial, final e o passo para que fosse realizado a repetição.
b) Neste caso, o usuário informou que o laço seria repetido de 1 até 5 vezes, assim sendo foi executado 4 vezes.

Exemplo 5: Fatorial de um número informado pelo usuário

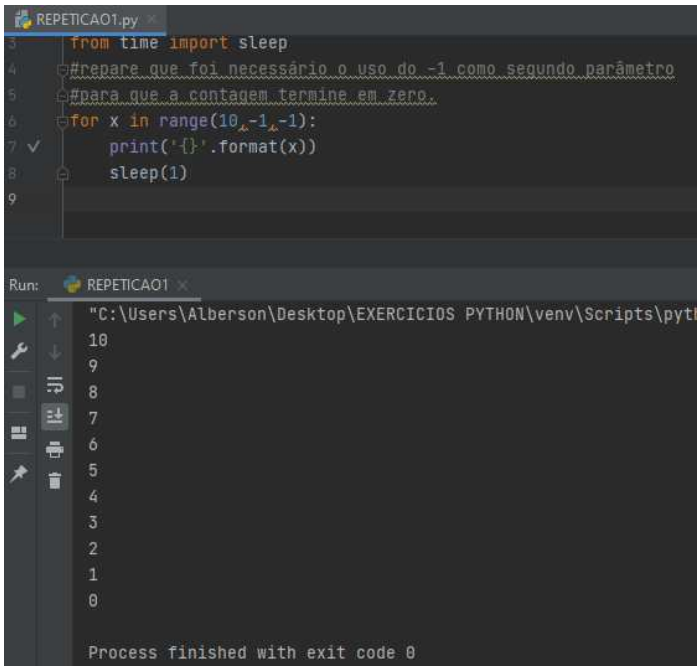


Repare neste exemplo:

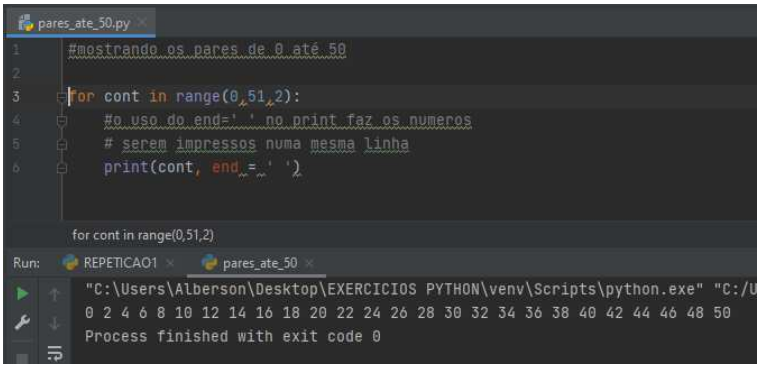
a) Só para lembrar, fatorial de um número é o produto entre todos os números que o antecede, acumulando tais resultados. Portanto, fatorial do 3! = 3*2*1 ou 3! = 1*2*3
b) Para este exemplo o programa deve repetir o laço de acordo com número de vezes informado pelo usuário
c) Para cálculo do fatorial do número 3, o laço deve ser executado de 1 a 3 vezes e não 2. Desta forma foi necessário escrever *vf+1* dentro o parâmetro range, pois se isto não fosse feito, o laço seria executado de 1 até 2.

Exemplos diversos:

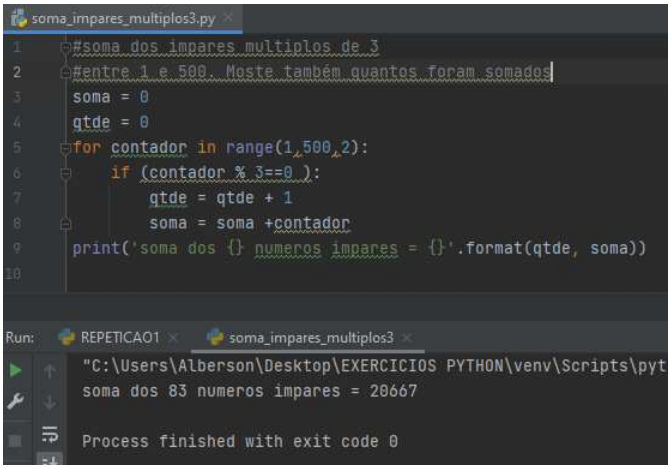
a) Fazer um programa em python para realizar uma contagem regressiva de 10 até zero de um em um segundo. É necessário uso da biblioteca time para uso da funcionalidade sleep, veja:



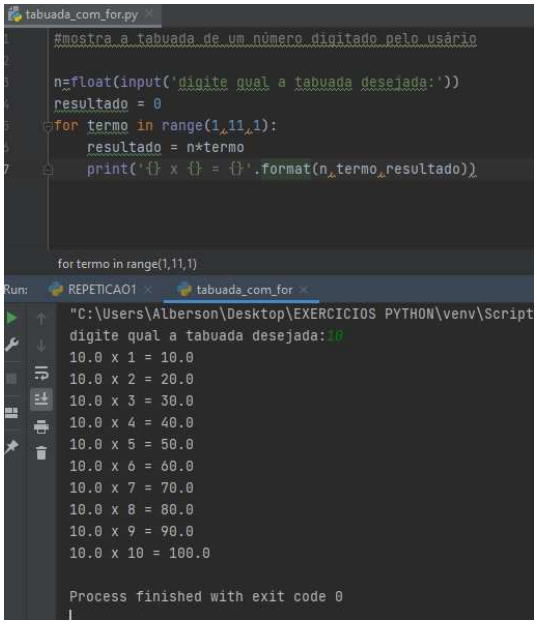
b) Mostrando os pares de 0 até 50:



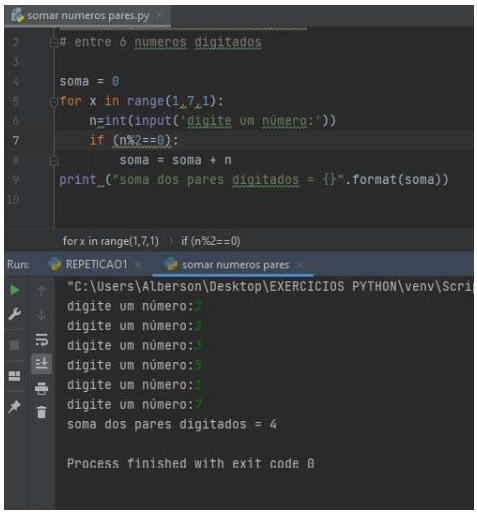
c) Percorrer os ímpares entre 1 e 500 e mostre quantos são os múltiplos de 3 entre estes ímpares e some-os:



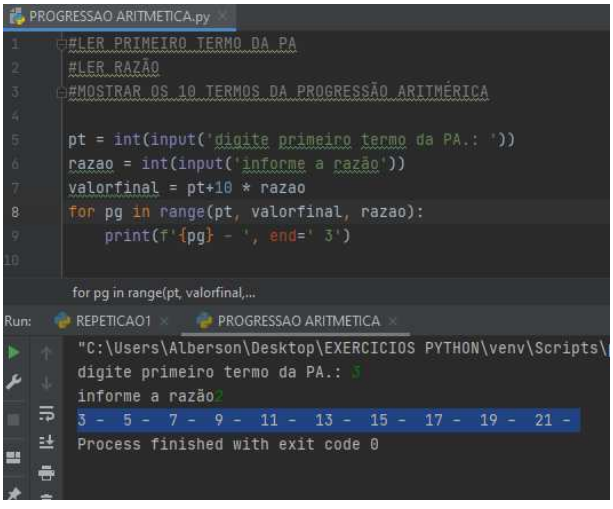
d) Mostrar a tabuada de um número digitado pelo usuário:



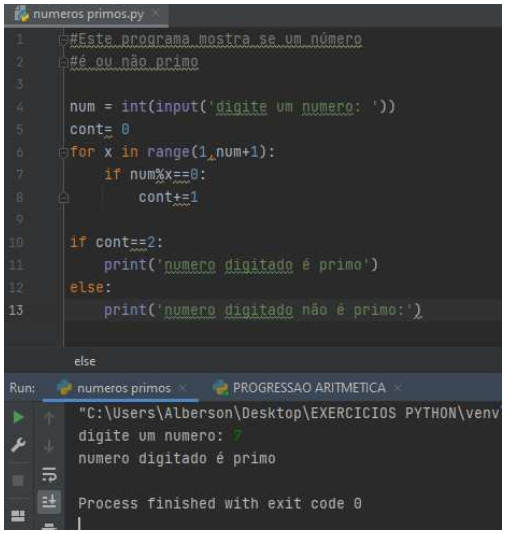
e) Somar todos os números pares entre 6 números digitados pelo usuário:



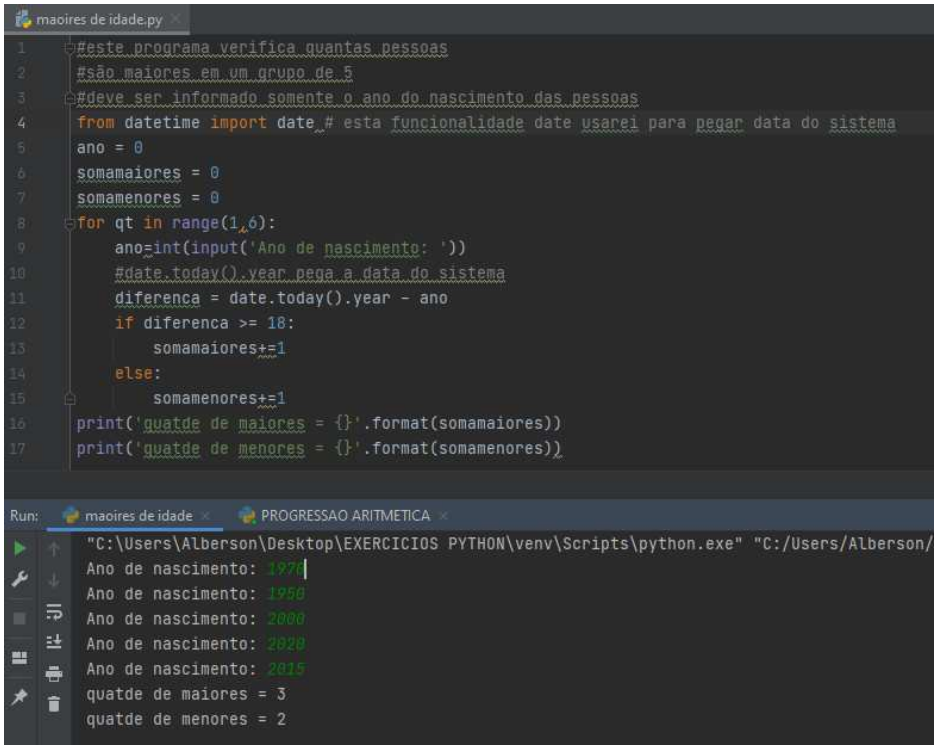
f) Mostrar os 10 primeiros termos de uma P.A. (PROGRESSÃO ARITMÉTICA) que inicie de um número informado pelo usuário e use uma razão, também informada pelo usuário:



g) Informar ao usuário se um número digitado é ou não primo:



h) O programa abaixo mostra quantas pessoas são maiores e quantas são menores num grupo de 5 pessoas:



i) O último exemplo abaixo mostrará a média de idade entre 5 pessoas, qual o nome do homem mais velho e quantas mulheres tem menos de 20 anos:

analisadorDADOS.py

```
# este programa analisa e mostra: nome do mais velho, média de idades
# e quantas mulheres tem menos de 20 anos
idademaisvelho = 0
qtdemulheres = 0
soma = 0
mediaidade = 0
for x in range(1,6,1):
    print('{}ª pessoa: '.format(x))
    print(30*'=')
    nome=input("nome:")
    sexo = input("sexo:")
    idade = int(input('idade: '))
    soma += idade
    if idade>idademaisvelho:
        idademaisvelho = idade
        nomemaisvelho = nome
    if sexo == 'f' and idade<20:
        qtdemulheres+=1
mediaidade = soma /5

print(f'Média das idades = {mediaidade}')
print(f'Quantidade de mulheres menores de 20 anos: {qtdemulheres}')
print(f'Homem mais velho = {nomemaisvelho}')
```

Rodando o programa temos o seguinte resultado, como exemplo:

analisadorDADOS

```
1ª pessoa:
==============================
nome:alberson
sexo:m
idade: 50
2ª pessoa:
==============================
nome:albert
sexo:m
idade: 49
3ª pessoa:
==============================
nome:irene
sexo:f
idade: 50
4ª pessoa:
==============================
nome:nair
sexo:f
idade: 17
5ª pessoa:
==============================
nome:janaina
sexo:f
idade: 13
Média das idades = 35.8
Quantidade de mulheres menores de 20 anos: 2
Homem mais velho = alberson
```

25.2 – COMANDO while

Como é de conhecimento de todos, este comando deve ser usado principalmente quando o programador NÃO SABE quantas vezes um trecho de programa deve ser repetido. Como exemplo e lembrança desta técnica, podemos citar trechos de programas para exigir a digitação de dados como “deseja continuar? s/n”, o qual dependerá da resposta “s” ou “n” do usuário. Caso o usuário digite algo diferente o programa repete esta pergunta.

Sintaxe:

<valor inicial da variável que controla laço>

while condição(ções):

<bloco de comandos caso teste condicional seja verdadeiro>

<alterar valor das variável<veis> que controla(lam) laço>

IMPORTANTE: REPARE QUE PARA ESTE COMANDO NÃO EXISTE FIM DO ENQUANTO, OU SEJA, FECHAMENTO DA ESTRUTURA. A SIMPLES IDENTAÇÃO CORRETA INDICA QUAIS OS COMANDOS SERÃO REPETIDOS.

Exemplo 1:

validacaoDADOS.py

```
resp = input('deseja iniciar programa? s/n')

while resp != "s" and resp != "n":
    resp = input('resposta inválida!!!! continuar? s/n')

print('resposta ok. Vamos continuar o programa')
print('fim do programa')
```

Run: validacaoDADOS

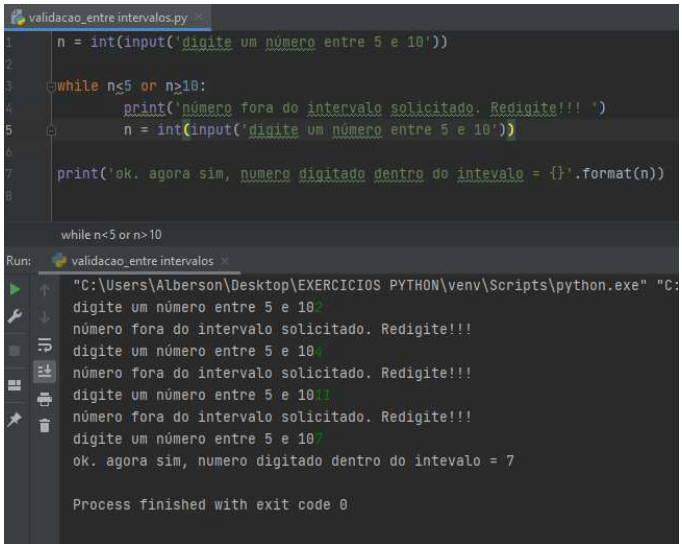
```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Script
deseja iniciar programa? s/nsim
resposta inválida!!!! continuar? s/nnao
resposta inválida!!!! continuar? s/nquero
resposta inválida!!!! continuar? s/nS
resposta inválida!!!! continuar? s/nN
resposta inválida!!!! continuar? s/ns
resposta ok. Vamos continuar o programa
fim do programa

Process finished with exit code 0
```

Repare neste exemplo:

a) Enquanto o usuário digita dados diferentes de “s” e “n” o programa exige do usuário a digitação correta.
b) Quando o usuário digita o dado corretamente, ou seja, “s” ou “n” o laço de repetição é abandonado e continua-se a execução do programa.

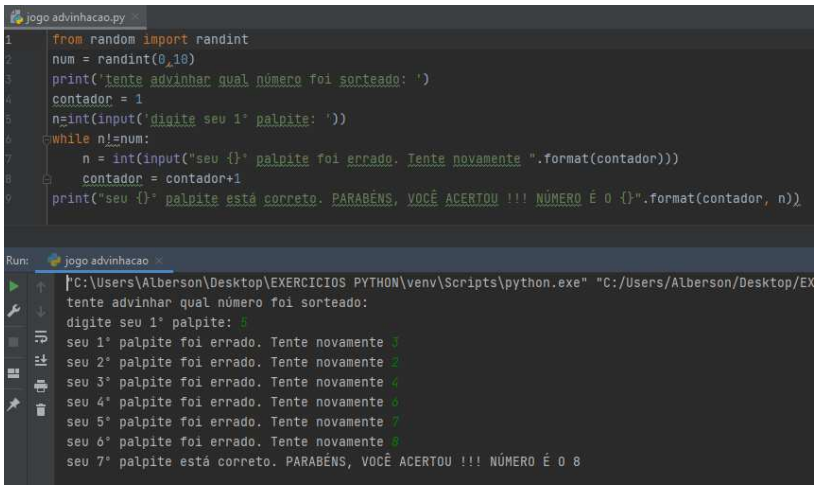
Exemplo 2:



Repare neste exemplo:

a) No caso é exigido do usuário a digitação de um número entre 5 e 10. Se for digitado número fora deste intervalo o programa não permite sair do laço de repetição.
b) Quando digitado número do intervalo solicitado, o laço de repetição é abandonado e o próximo comando FORA DO LAÇO DE REPETIÇÃO será executado, mostrando o número que foi digitado.

Exemplo 3:



Repare neste exemplo:

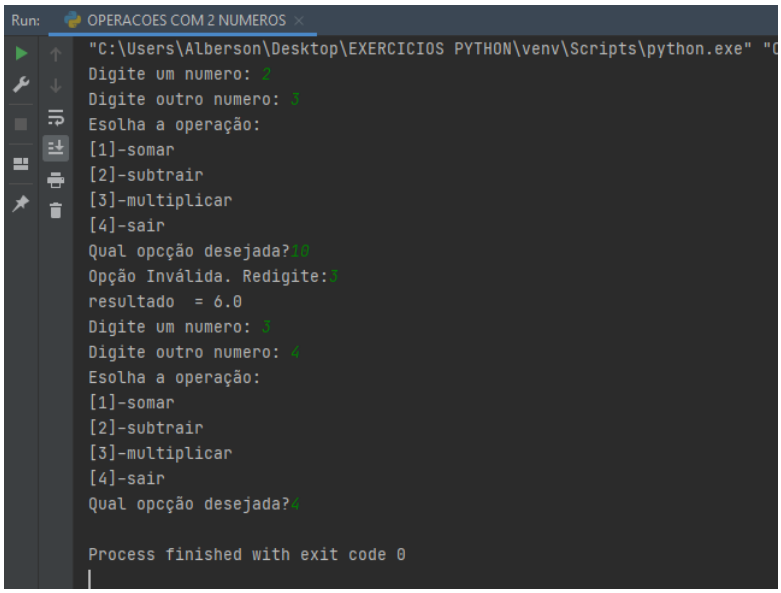
a) Neste programa o usuário tenta adivinhar um número sorteado pelo computador entre 0 e 10.
b) Quando ele acerta, o programa informa quantas tentativas foram necessárias para o acerto. Neste caso usamos também o comando while.

Exemplo 4 :

```
n1 = float(input('Digite um numero: '))
n2 = float(input('Digite outro numero: ' ))
print("Esolha a operação: ")
print('[1]-somar')
print('[2]-subtrair')
print('[3]-multiplicar')
print('[4]-sair')
opcao = int(input('Qual opcção desejada?'))
while opcao <1 or opcao >4:
    opcao = int(input('Opção Inválida. Redigite:'))

while opcao != 4:
    if opcao == 1:
        r = n1+n2;
    elif opcao ==2:
        r = n1-n2;
    elif opcao == 3:
        r= n1*n2
    print('resultado  = {}'.format(r))
    n1 = float(input('Digite um numero: '))
    n2 = float(input('Digite outro numero: ' ))
    print("Esolha a operação: ")
    print('[1]-somar')
    print('[2]-subtrair')
    print('[3]-multiplicar')
    print('[4]-sair')
    opcao = int(input('Qual opcção desejada?'))
    while opcao <1 or opcao >4:
        opcao = int(input('Opção Inválida. Redigite:'))
```

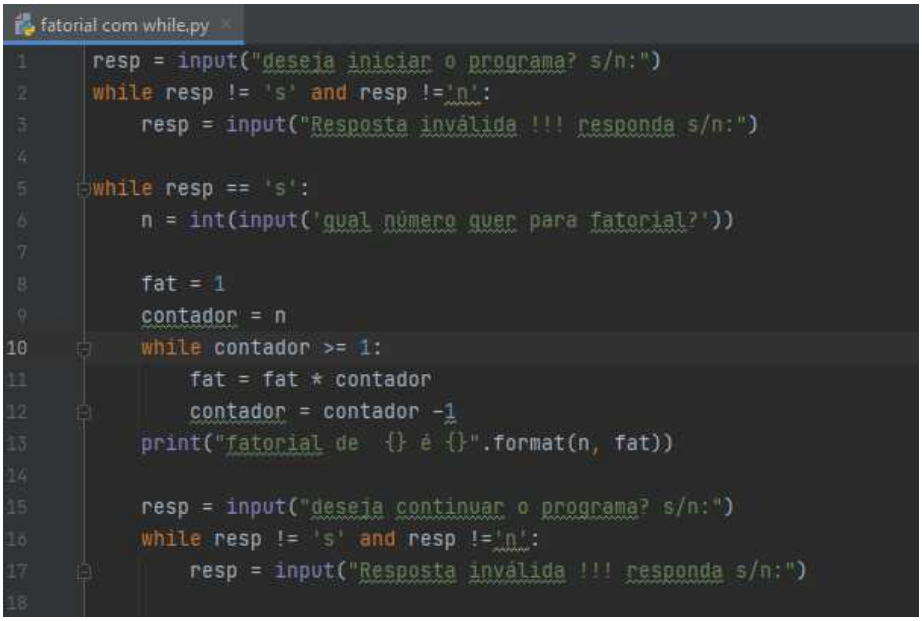
Veja o resultado do teste deste último programa:



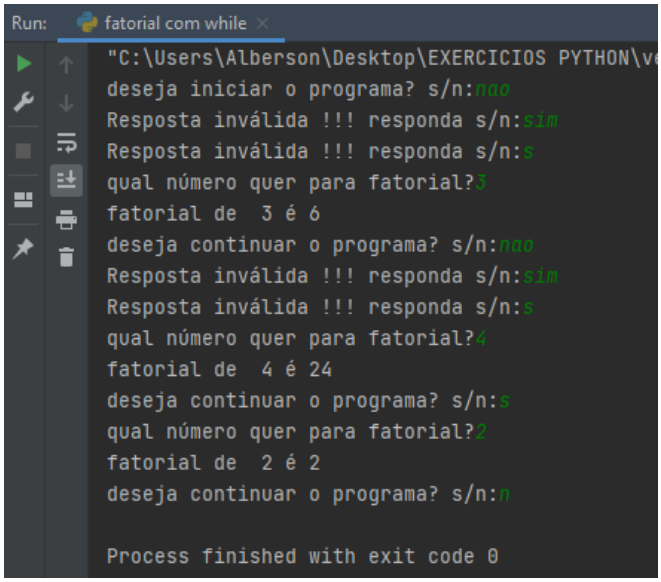
Repare no código do programa:

a) Existem dois laços de repetições do comando while. O primeiro laço exige a digitação de uma das opções disponíveis. Já o segundo é usado para continuar solicitando números e novas escolhas de operações.

Exemplo 5: Neste exemplo o programa calcula fatoriais de que números informados. Este programa só é finalizado quando o usuário responder que deseja finalizar o programa. Analise:

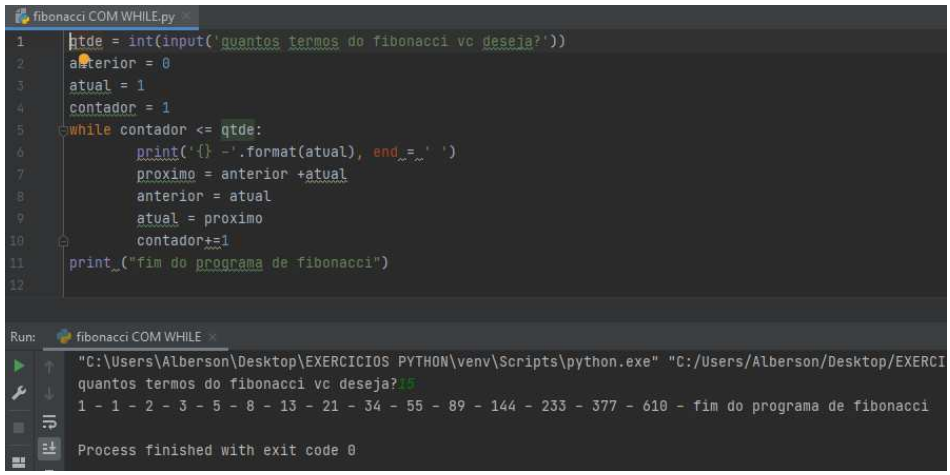


Veja o teste realizado para o programa do exemplo em questão:

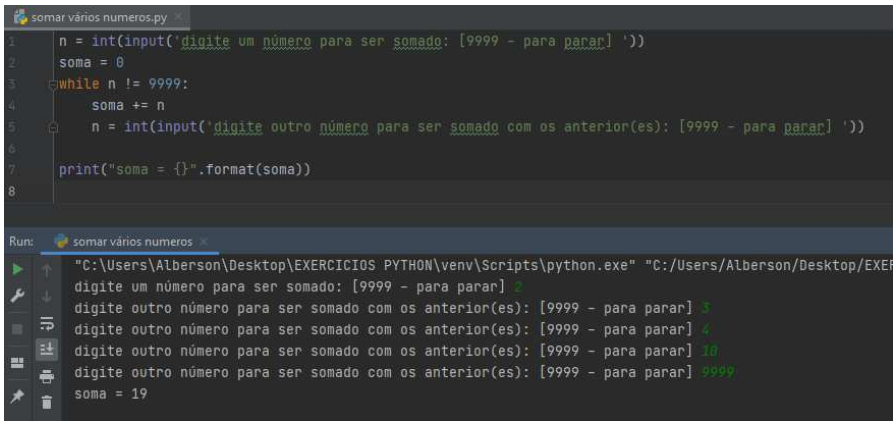


Vamos ver outros exemplos de uso do comando while a seguir:

a) O exemplo abaixo imprime a sequência de Fibonacci até a quantidade de termos informada pelo usuário:



b) O próximo exemplo faz soma de vários números digitados. Quando o usuário digitar 9999 o programa finaliza e exibe o resultado da soma dos ***n*** números digitados:

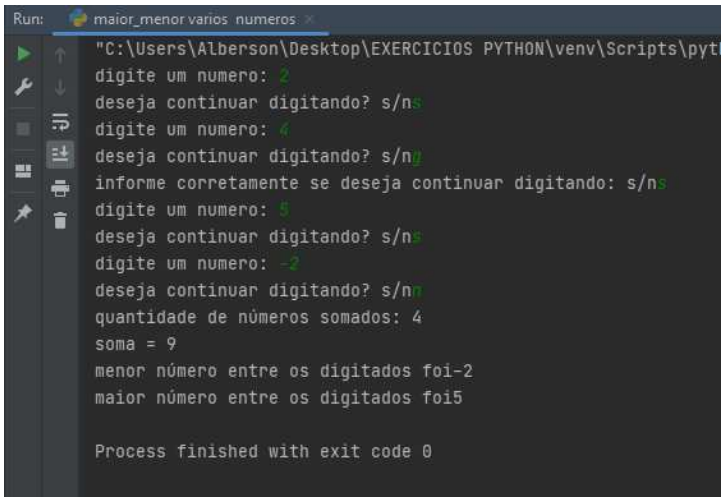


c) O programa abaixo soma diversos números digitados pelo usuário. Este programa exibe a soma calculada e também qual foi o menor e maior números digitados:

```
soma = 0
qtdenumeros = 1
resp = 's'
while resp == 's':
    n = int(input('digite um numero: '))
    if qtdenumeros == 1:
        maior = n
        menor = n
    if n > maior:
        maior = n
    if n  < menor:
        menor = n
    soma+=n
    resp = input('deseja continuar digitando? s/n')
    while resp !='s' and resp != 'n':
        resp = input('informe corretamente se deseja continuar digitando: s/n')
    if resp == 's':
        qtdenumeros += 1
print ("quantidade de números somados: {} ".format(qtdenumeros))
print (f'soma = {soma}')
```

```
print (f'menor número entre os digitados foi{menor}')
print (f'maior número entre os digitados foi{maior}')
```

Veja o resultado do teste abaixo :

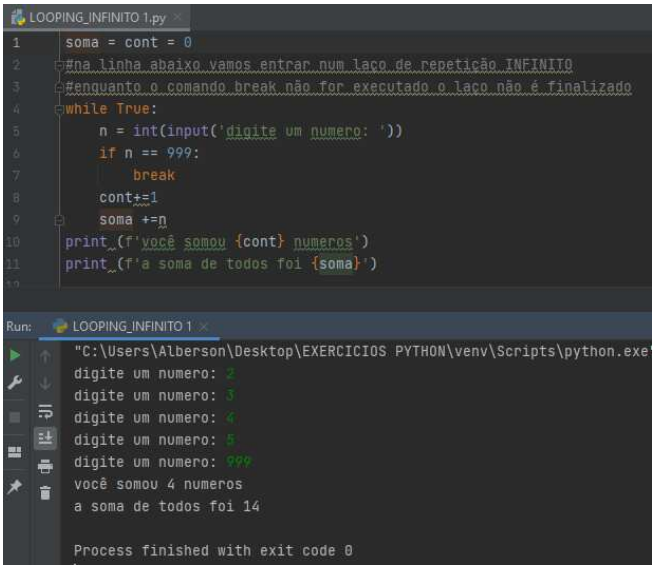


25-3 – LOOPING INFINITO COM COMANDO while / break

É possível utilizarmos looping de repetição INFINITO com o python. Para isso, usaremos o "***True***" (com "T", primeira letra maiúscula) no local do teste condicional no comando while. Assim sempre o teste será VERDADEIRO, o que provocará sempre a entrada no laço de repetição. Este laço só será interrompido quando for encontrado o comando ***break***. Ao encontrar o ***break***, o laço é abandonado **na linha onde o mesmo foi escrito**, **independente do que seria executado nas linhas posteriores**.

Vejamos exemplos:

Exemplo 1: Somando vários números digitados até que seja digitado 999 para finalizar o programa.



Vamos analisar o código do exemplo 1:

a) Repare que na linha 4 o foi escrita a seguinte linha de comando ***while True:*** . Isto quer dizer que está sendo criada uma estrutura de repetição INFINITA, ou seja, será executada para sempre.
b) No teste condicional escrito na linha 6, repare que se o usuário digitar ***999*** o comando ***break*** interromperá o laço na linha 7, ou seja, todos os comandos escritos dentro do while, a partir da linha 8, não serão mais executados.

Exemplo 2: Jogando PAR OU ÍMPAR com o computador. O programa deve terminar quando o computador vencer o usuário. Quando terminar mostre quantas partidas o usuário venceu.

LOOPING INFINITO 2.py

```
1  import random
2  ptsusuario = ptscomputador = 0
3  while True:
4      n = int(input('qual número entre 0 e 10? '))
5      parouimpar = input('par ou impar? responda: p/i  ')
6      while parouimpar != 'p' and parouimpar != 'i':
7          parouimpar = input('Resposta inválida? responda: p/i  ')
8      computador = random.randint(0,10)
9      soma = n+computador
10     if soma%2 == 0 and parouimpar == 'p':
11         ptsusuario+=1
12     elif soma%2 == 1 and parouimpar == 'i':
13         ptsusuario += 1
14     else:
15         ptscomputador+=ptscomputador
16         break
17 print(f"você venceu {ptsusuario} partidas")
```

Veja o teste abaixo com o programa em execução:

Run: LOOPING INFINITO 2

```
"C:\Users\Alberson\Desktop\EXERCICIOS
qual número entre 0 e 10? 2
par ou impar? responda: p/i  p
qual número entre 0 e 10? 4
par ou impar? responda: p/i  p
qual número entre 0 e 10? 3
par ou impar? responda: p/i  i
qual número entre 0 e 10? 9
par ou impar? responda: p/i  i
qual número entre 0 e 10? 5
par ou impar? responda: p/i  p
qual número entre 0 e 10? 4
par ou impar? responda: p/i  i
qual número entre 0 e 10? 4
par ou impar? responda: p/i  i
você venceu 6 partidas

Process finished with exit code 0
```

Repare no programa desenvolvido para este caso:

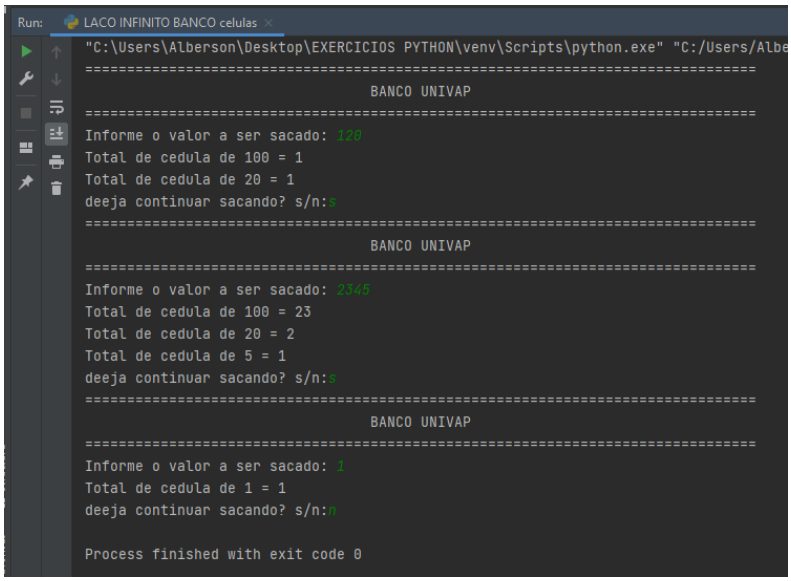
a) Quando o computador vence, ou seja, quando o ***else*** escrito na linha 14 é executado, soma-se ponto da vitória para computador e finaliza-se o laço de repetição com o comando ***break***
b) O programa informa, em seguida, a quantidade de jogos vencidos pelo usuário.

Vamos a outros exemplos:

a) O programa abaixo simula um caixa eletrônico. Diante de um valor monetário informado pelo usuário, ele informa a quantidade de cédulas de cada valor serão entregues ao cliente. (R$100/R$50/R$20/R$10/R$5/R$2/R$1)

```
while True:
    print(80 * '=')
    print('{:^80}'.format('BANCO UNIVAP'))
    print(80 * '=')
    valor = int(input('Informe o valor a ser sacado: '))
    valortotal = valor
    cedula = 100
    qtdecedula = 0
    while True:
        if valortotal >= cedula:
            valortotal -= cedula
            qtdecedula+=1
        else:
            if qtdecedula >0:
                print(f'Total de cedula de {cedula} = {qtdecedula}')
            if cedula == 100:
                cedula = 50
            elif cedula == 50:
                cedula = 20
            elif cedula == 20:
                cedula = 10
            elif cedula == 10:
                cedula = 5
            elif cedula == 5:
                cedula = 2
            elif cedula == 2:
                cedula = 1
            qtdecedula = 0
            #print (f"valor total = {valortotal}")
            if valortotal == 0 :
                break
    resp = input("deeja continuar sacando? s/n:")
    while resp != 's' and resp != 'n':
        resp = input("deeja continuar sacando? responda -> s/n:")
    if resp == 'n':
        break
```

Veja o teste do programa das cédulas bancárias:



Observações importantes sobre este assunto da apostila!!!

```
while True:
```

Quando usar o True em substituição de testes condicionais, escreva-o com a primeira letra em maiúsculo.

```
    break
```

O laço de repetição ***while True:*** é abandonado sempre na linha de comando ***break***, existindo ou não outros comandos dentro do laço, após a linha onde este foi escrito.