

E-book: Primeiros passos em Python.

ÍDICE

=====

Capítulo 1: Como baixar e instalar o Python com a IDE Pycharm Community, em seu PC.

- 1.1 – Como baixar e instalar o Python no Windows.
- 1.2 – Como baixar e instalar o Python em sistemas operacionais GNU/Linux.
- 1.3 – Como baixar e instalar o Python no Mac OS.
- 1.4 – Como baixar a IDE PyCharm(Windows,GNU/Linux e Mac OS)

Capítulo 2: Aprendendo a usar as formatações de print no Python.

- 2.1 – O que é uma linguagem de programação (uma breve explicação sobre computação).
- 2.2 – Introdução a IDE Pycharm.
- 2.3 – Formatações mais usadas em Python e seu primeiro código em Python.
- Exercícios do capítulos 2.

Capítulo 3: Como declarar variáveis int, float e str.

- 3.1 – Entendendo e declarando variáveis com o int.
- 3.2 – Entendendo e declarando variáveis com o float.
- 3.3 – Entendendo e declarando variáveis com o str.
- 3.4 – Operadores aritméticos.
- 3.5 – Operadores relacionais(de comparação).
- Exercícios do capítulo 3.

Capítulo 4: Introdução as condicionais (if, else e elif)

- 4.1 – O que são estruturas condicionais.
- 4.2 – Como usar a estrutura condicional if.

4.3 – Como usar a estrutura condicional **else**.

4.4 – Como usar a estrutura condicional **elif**.

Exercícios do capítulo 4.

Capítulo 5: Usando estruturas condicionais em seus códigos.

5.1 – **Aninhamento**, entendendo como o programa lê as condicionais dentro de condicionais.

5.2 – Exemplos de códigos com **aninhamento**.

Exercícios do capítulo 5.

Capítulo 6: Introdução as estruturas de repetição (for** e **while**)**

6.1 – A estrutura de repetição **for** e como usá-la.

6.2 – Usando a estrutura de repetição **while**.

6.3 – Diferença entre **for** e **while** e quando usá-las.

Capítulo 7: Usando estruturas de repetição em seus códigos.

7.1 – Exemplos de códigos utilizando estruturas de repetição.

Exercícios dos capítulos 6 e 7.

Capítulo 8: Aplicando seus conhecimentos em códigos práticos.

Exercícios para praticar a programação de códigos para usar no dia a dia.

Capítulo 9: Resposta dos exercícios propostos.

Respostas de todos os exercícios propostos.

Capítulo 10: Bônus, seus próximos passos em Python.

Já aprendi tudo? Se não, quais meus próximos passos?

Introdução:

Bem vindo(a) ao E-book: Primeiros passos em Python, meu objetivo nesse E-book é mostrar de forma fácil e bem ilustrada como programar em Python. Ao longo dos capítulos você irá ter contado com as estruturas básicas da linguagem e entender como e porquê uma linguagem de programação funciona. No E-book você terá o material necessário para iniciar muito bem na programação em Python, construindo uma base bem sólida não só em Python mas desenvolvendo a forma de pensar como um programador, o que irá te ajudar na maioria das linguagens de programação que existem, a famosa “ **lógica de programação** ”. Existe muito para aprender em Python, nosso objetivo é oferecer o início, a base para programar nessa linguagem, sendo assim o E-book não deve ser o último “curso” que você deve fazer, sempre procure mais aprendizado e sempre pergunte os “**porquês**”. Vamos lá.

*ISB

=====

=====

=====

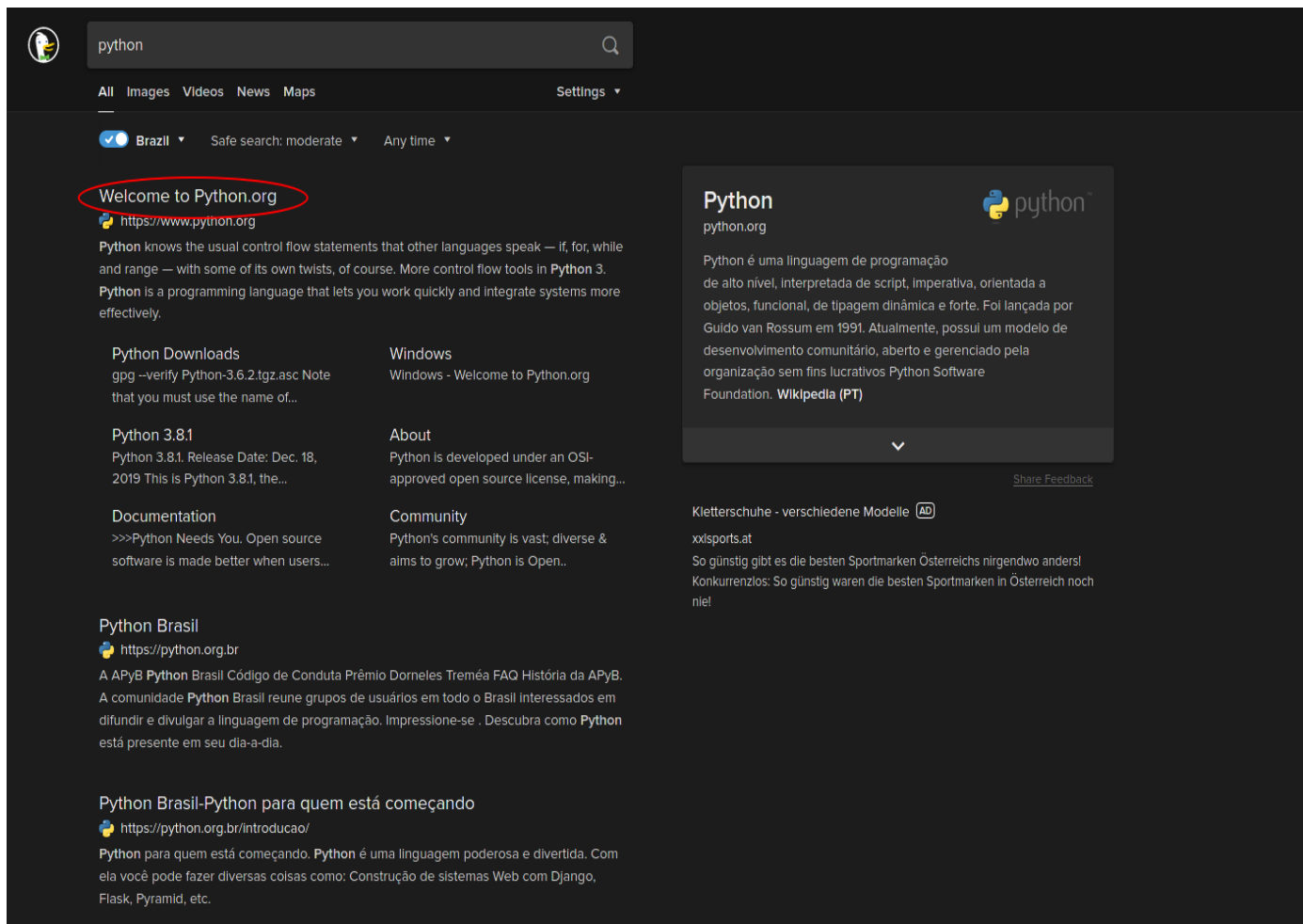
Capítulo 1: Como baixar e instalar o Python com a IDE PyCharm Community, em seu PC.

Instalar a IDE PyCharm será muito útil para seu aprendizado em Python visto que por ser uma IDE (Integrated Development Environment ou ambiente de Desenvolvimento Integrado) possui inúmeras ferramentas que auxiliarão seus estudos na linguagem Python, como interface gráfica interativa, análise do código para identificar bugs e autocompletagem de partes do código. Não se preocupe, essas ferramentas ficarão mais claras ao decorrer do curso. Mas primeiro veremos como instalar o Python.

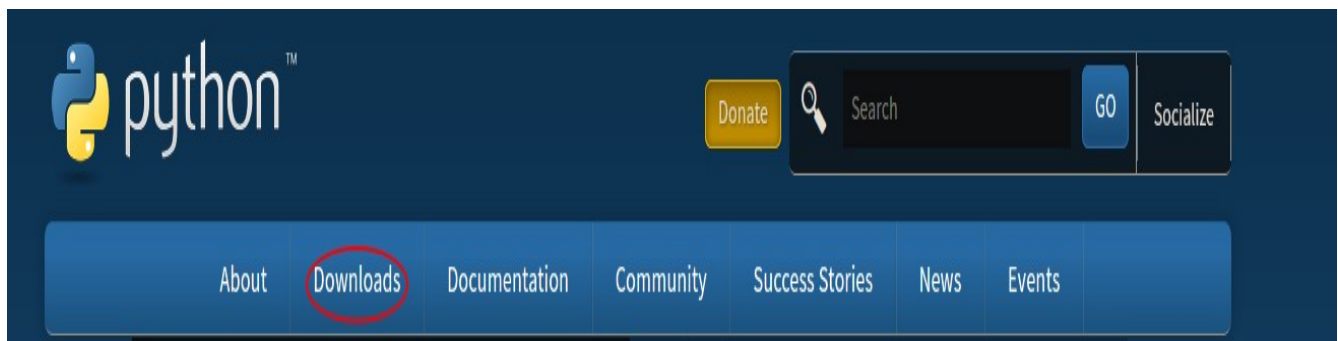
1.1 – Como baixar e instalar o Python no **Windows**.

O Windows é o sistema operacional mais usado no mundo, por isso instalar o PyCharm nele não será um grande problema. Vamos aos passos:

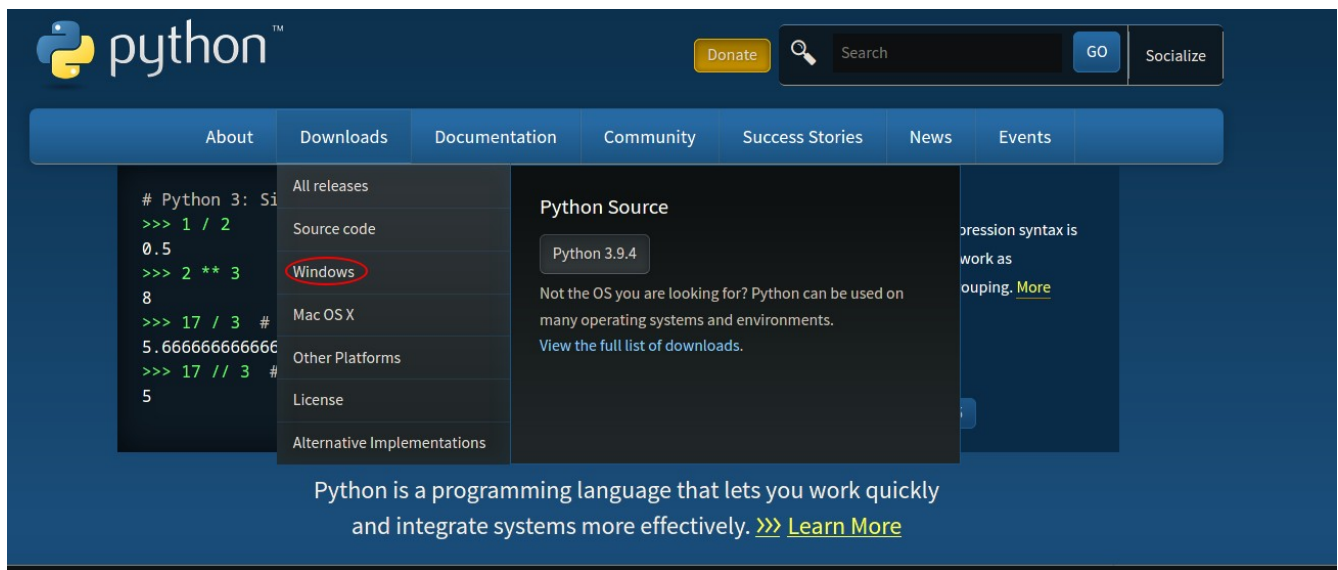
- 1 – Digite em seu buscador “Python” sem aspas. Aparecerá isso:



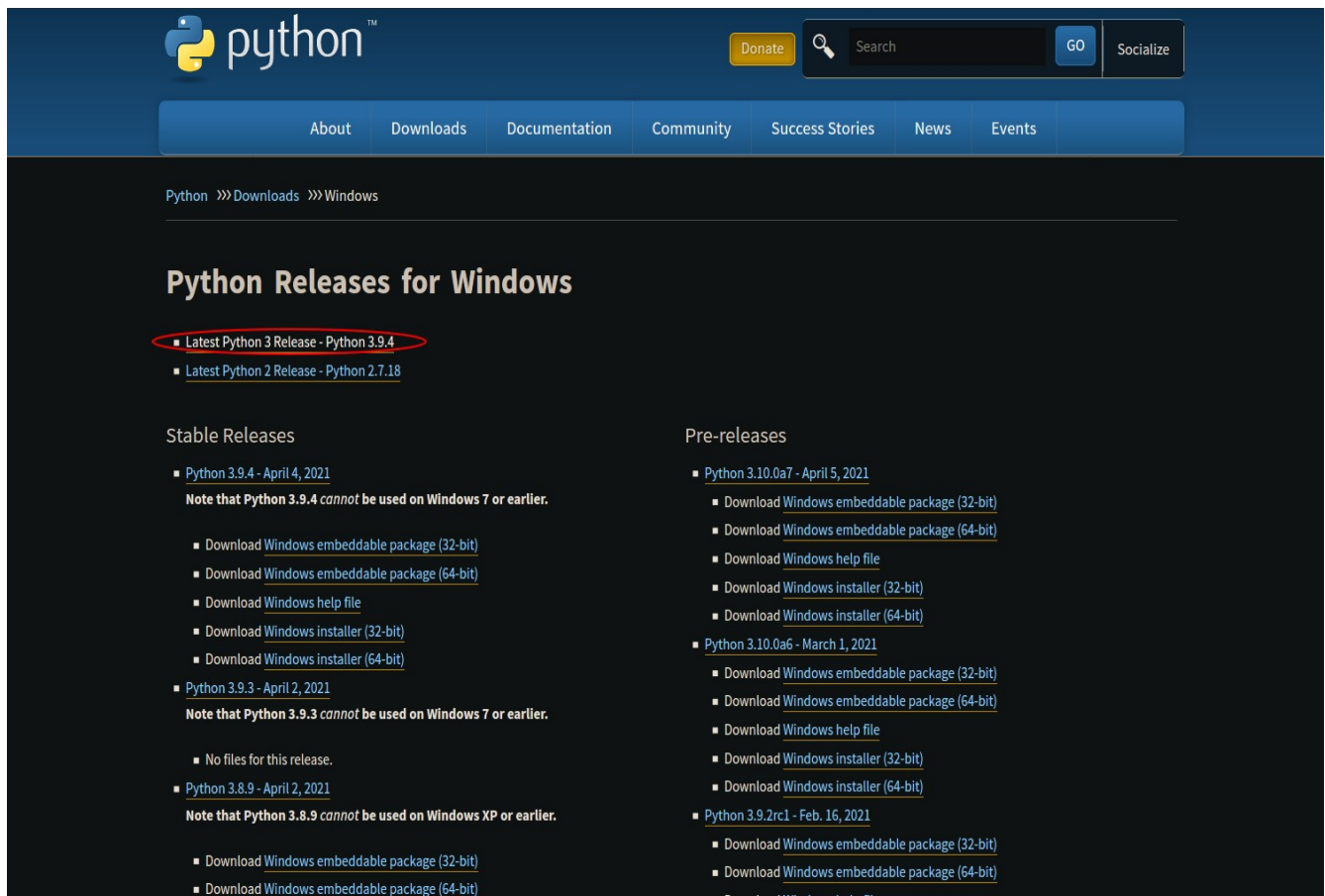
2 – Clique em “Welcome to Python.org”.



3 – Clique em “Downloads”.



4 – Clique na versão mais recente, nesse caso é a 3.9.4.



5 – Aqui escolha qual é a arquitetura do seu processador: x86 = 32 bits ou 64 bits, e baixe o instalador.

| Files | | | | | |
|---|------------------|--|----------------------------------|-----------|---------------------|
| Version | Operating System | Description | MD5 Sum | File Size | GPG |
| Gzipped source tarball | Source release | | cc8507b3799ed4d8baa7534cd8d5b35f | 25411523 | SIG |
| XZ compressed source tarball | Source release | | 2a3dba5fc75b695c45cf1806156e1a97 | 18900304 | SIG |
| macOS 64-bit Intel installer | Mac OS X | for macOS 10.9 and later | 2b974bfd787f941fb8f80b5b8084e569 | 29866341 | SIG |
| macOS 64-bit universal2 installer | Mac OS X | for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental) | 9aa68872b9582c6c71151d5dd4f5ebca | 37648771 | SIG |
| Windows embeddable package (32-bit) | Windows | | b4bd8ec0891891158000c684422014d | 7580762 | SIG |
| Windows embeddable package (64-bit) | Windows | | 5c34eb7e79cfe8a92bf56b5168a459f4 | 8419530 | SIG |
| Windows help file | Windows | | aaacf224768b5e4aa7583c12af68fb0 | 8859759 | SIG |
| Windows installer (32-bit) | Windows | | b790fdaff648f757bf0f233e4d05c053 | 27222976 | SIG |
| Windows installer (64-bit) | Windows | Recommended | ebc65aaa142b1d6de450ce241c50e61c | 28323440 | SIG |

Agora é só abrir o instalador e seguir o passo a passo para instalar o Python. Se tiver dúvidas pode ser útil procurar algum tutorial no YouTube.

1.2 – Como baixar e instalar o Python em sistemas operacionais GNU/Linux.

No caso do Linux é bem simples, na maioria das distribuições já vem o Python instalado e atualizado, como por exemplo no Ubuntu, Linux Mint e Arch Linux. Para saber a versão do Python que você está usando abra o terminal Linux pelo atalho do teclado “ (ctrl + alt + t) em distros que utilizem o DE Gnome, KDE por exemplo ” ou procurando no launcher por “terminal”. Quando abrir o terminal Linux digite “python” tudo minúsculo e sem as aspas, deve aparecer algo assim:

```
[redacted@arch ~]$ python
Python 3.9.3 (default, Apr  8 2021, 23:35:02)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

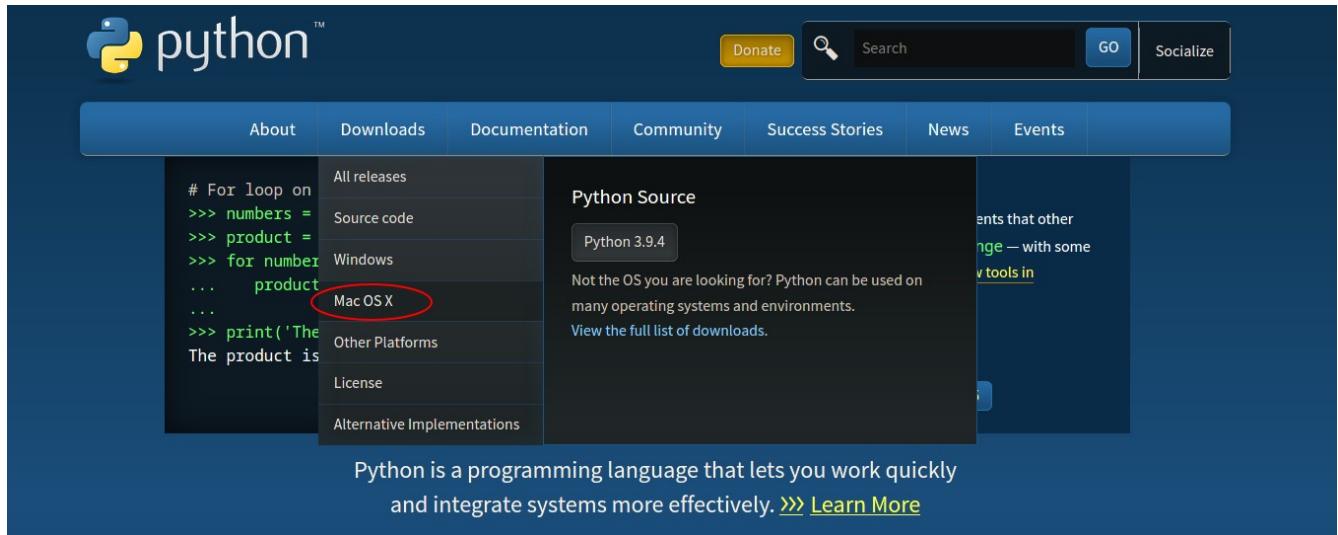
No meu caso a versão do Python é a 3.9.3.

Se a versão da seu PC estiver desatualizada procure como atualizar o Python na documentação da sua distro.

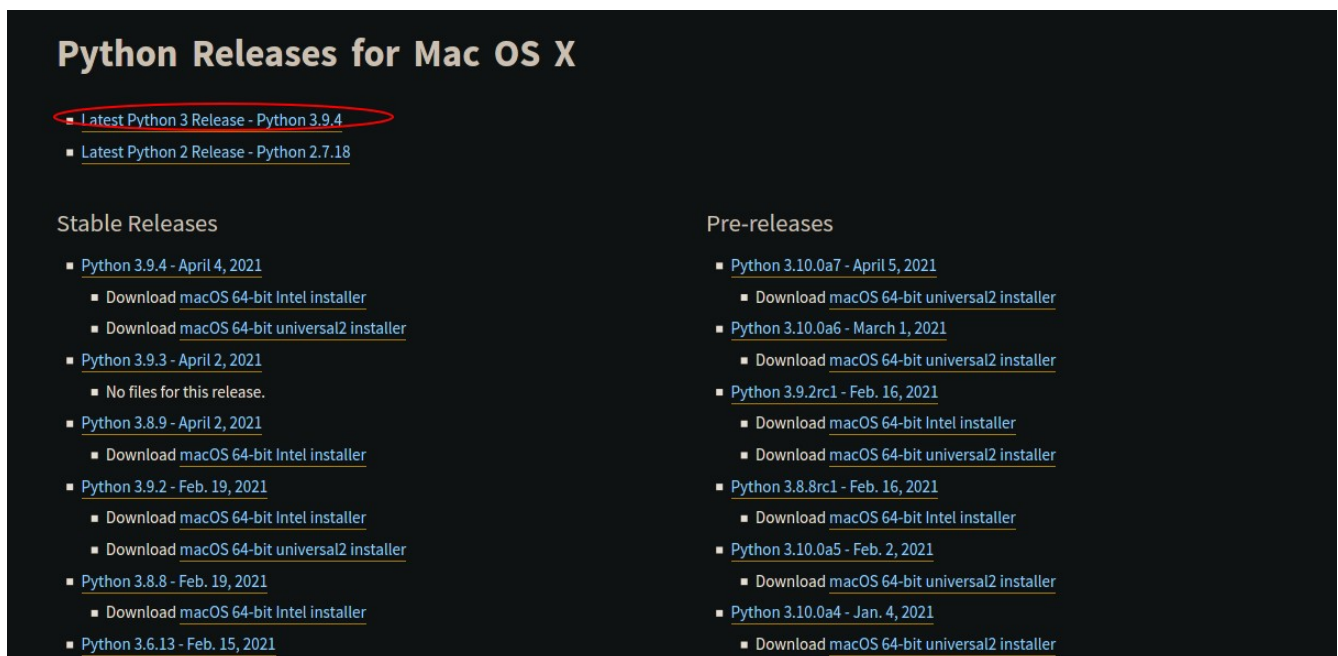
1.3 – Como baixar e instalar o Python no Mac OS.

Como no Windows é relativamente fácil instalar o Python no Mac OS, siga os passos 1 , 2 e 3 de como instalar no Windows e os seguintes passos:

4 - Clique em Mac OS X.



5 – Clique agora na versão mais atualizada.



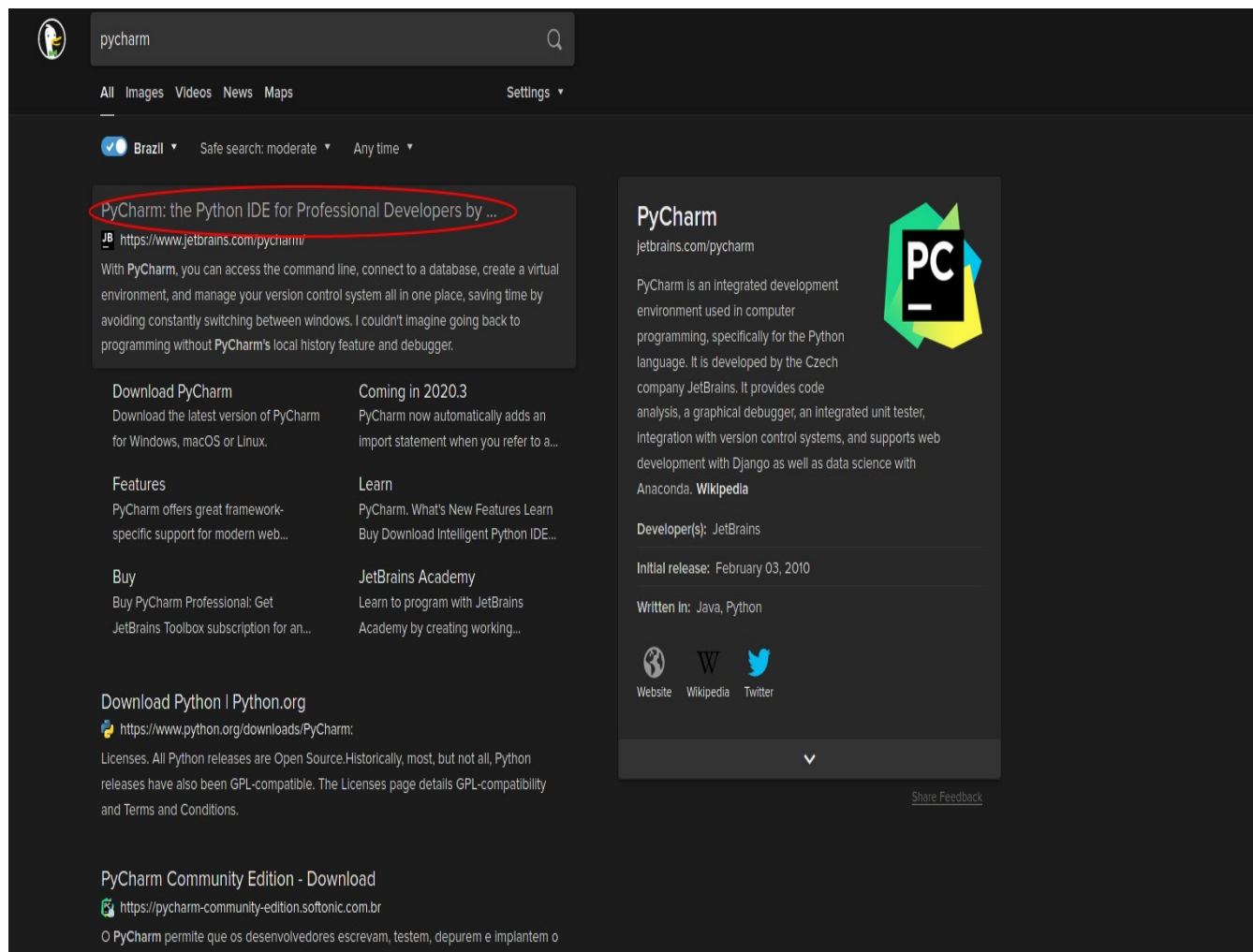
Agora é só abrir o instalador e seguir o passo a passo para a instalação.

1.4 – Como baixar a IDE PyCharm.

Como dito na introdução, uma IDE serve para facilitar a vida dos desenvolvedores disponibilizando ferramentas para a programação em geral. Ela deixará os seus estudos em Python muito mais amigáveis e otimizará seu tempo aprendendo.

Vamos aprender a instalar no **Windows**:

1 – Digite “pycharm” no seu buscador e clique no link marcado :



The screenshot shows a search engine results page for the query "pycharm". The search bar at the top contains the text "pycharm". Below the search bar, there are tabs for "All", "Images", "Videos", "News", and "Maps", with "All" selected. The location is set to "Brazil" and the search is set to "Safe search: moderate" and "Any time".

The first search result is titled "PyCharm: the Python IDE for Professional Developers by ..." and is circled in red. The URL is <https://www.jetbrains.com/pycharm/>. The description states: "With PyCharm, you can access the command line, connect to a database, create a virtual environment, and manage your version control system all in one place, saving time by avoiding constantly switching between windows. I couldn't imagine going back to programming without PyCharm's local history feature and debugger."

Below the first result, there are several other links and information:

- Download PyCharm**: Download the latest version of PyCharm for Windows, macOS or Linux.
- Coming in 2020.3**: PyCharm now automatically adds an import statement when you refer to a...
- Features**: PyCharm offers great framework-specific support for modern web...
- Learn**: PyCharm. What's New Features Learn Buy Download Intelligent Python IDE...
- Buy**: Buy PyCharm Professional: Get JetBrains Toolbox subscription for an...
- JetBrains Academy**: Learn to program with JetBrains Academy by creating working...

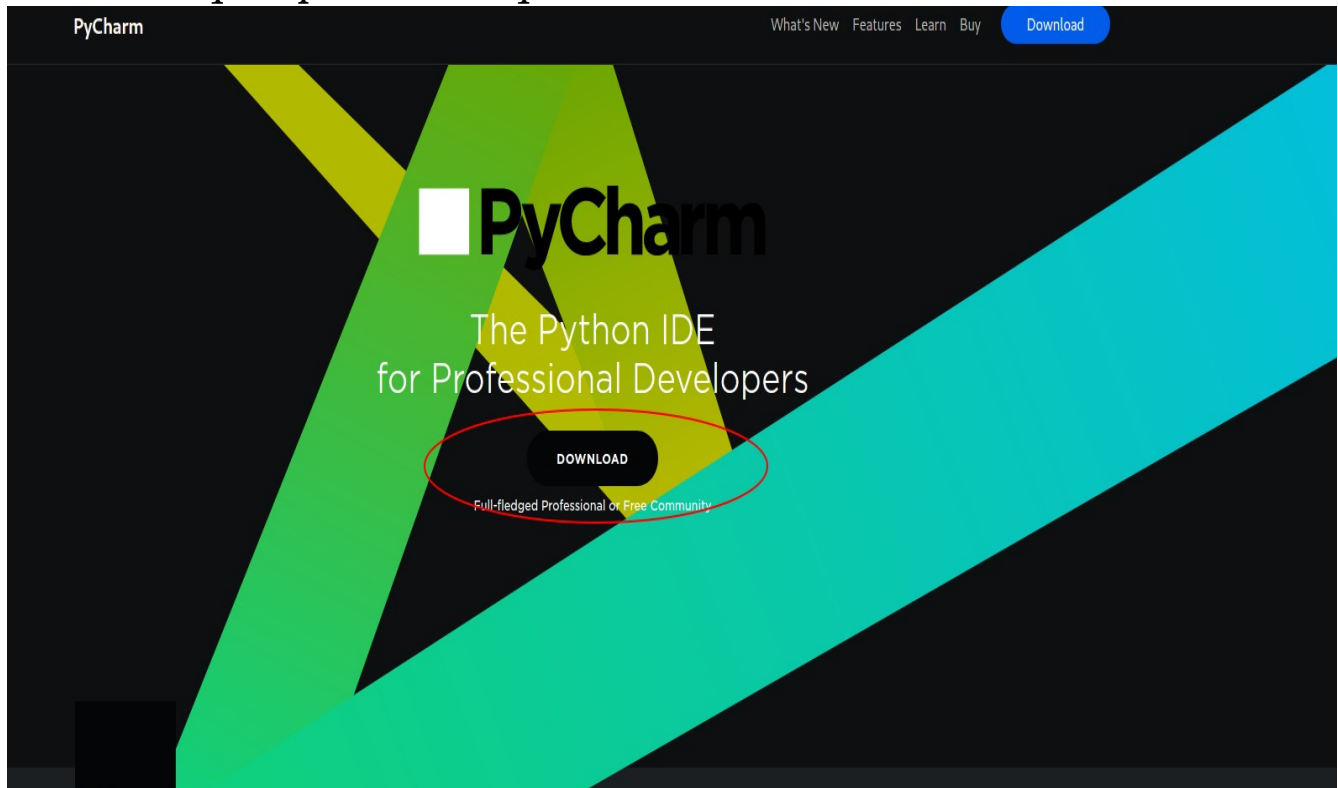
On the right side of the page, there is a detailed card for PyCharm:

- PyCharm**: jetbrains.com/pycharm
- PyCharm** is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django as well as data science with Anaconda. [Wikipedia](#)
- Developer(s)**: JetBrains
- Initial release**: February 03, 2010
- Written in**: Java, Python
- Links to [Website](#), [Wikipedia](#), and [Twitter](#)

At the bottom of the page, there are more links:

- Download Python | Python.org**: <https://www.python.org/downloads/PyCharm/>. Licenses. All Python releases are Open Source. Historically, most, but not all, Python releases have also been GPL-compatible. The Licenses page details GPL-compatibility and Terms and Conditions.
- PyCharm Community Edition - Download**: <https://pycharm-community-edition.softonic.com.br>
- O PyCharm permite que os desenvolvedores escrevam, testem, depurem e implantem o...

2 – Na tela que aparecerá clique em Download :



3 – Clique para baixar a versão Community pois ela é grátis (não esqueça de marcar a opção **Windows** :

Download PyCharm

Windows macOS Linux

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

Free trial

Community

For pure Python development

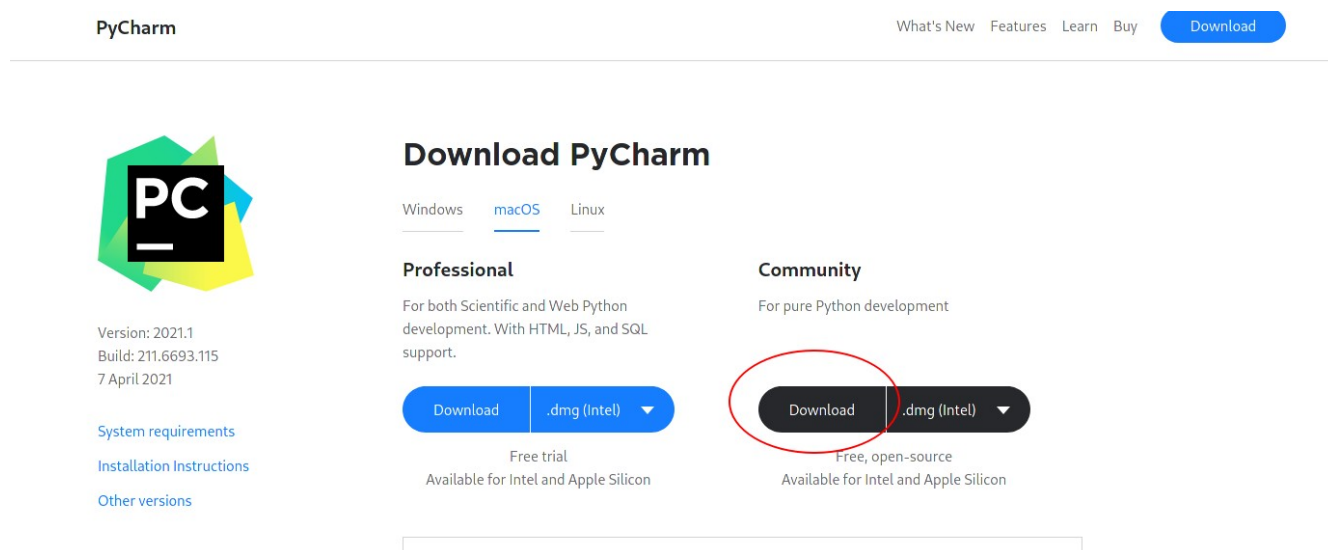
Download

Free, open-source

Depois de baixar abra o instalar e siga o passo a passo.

No Mac OS siga os passos 1 e 2 utilizado para **Windows** e os seguintes passos :

3 – Clique em Download da versão Community :



Feito os passos, você pode abrir o arquivo dmg e instalar o mesmo seguindo o passo a passo.

Por fim para sistemas **GNU/Linux** você pode baixar direto dos repositórios de sua distro, no Caso do Ubuntu utilize a Ubuntu Software e procure por PyCharm Community e instale o programa, para Distros baseadas em Arch Linux baste utilizar o comando “sudo pacman -S pycharm-community-edition” sem aspas e colocar sua senha de usuário.

Caso não consiga por esses meios ou use outra distro, procure como instalar na documentação da sua distro Linux.

Finalmente depois de instalar o Python e a IDE PyCharm temos todas as ferramentas necessárias para começar a aprender programação em Python. Então vamos começar.

Capítulo 2: Aprendendo a usar as formatações de **print** no Python.

2.1 – O que é uma linguagem de programação (uma breve explicação sobre computação).

Quando nos deparamos pela primeira vez com várias linhas de código, seja de Java, Python ou qualquer outra linguagem de programação é fácil nos sentirmos ignorantes perante tantos “**ifs**” e “**elses**”. E é mais fácil ainda nos confundirmos sobre o que realmente todas aquelas expressões e palavras em inglês significam de verdade.

Muitas vezes iniciantes tendem a achar que o computador realmente entende todas aquelas expressões na sua forma abstrata (palavras como “**in**”, “**if**” por exemplo) quando na verdade as linguagens de programação são meios de nos comunicarmos com o computador, já que o mesmo só entende código de máquina, o famoso **código binário**. Vamos então ter uma breve noção de alguns termos que serão usados a seguir.

Código binário :

Quando falamos de código binário muitas vezes nos lembramos dos “zeros” e “uns” que aparecem em linha na tela de um hacker em algum filme ou série.



1 = “ ligado, há tensão”

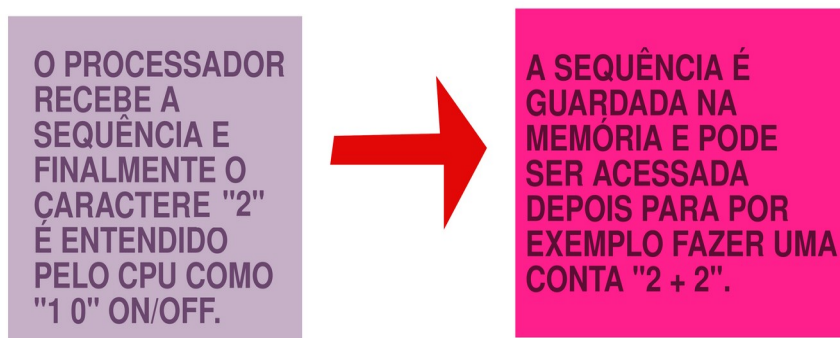
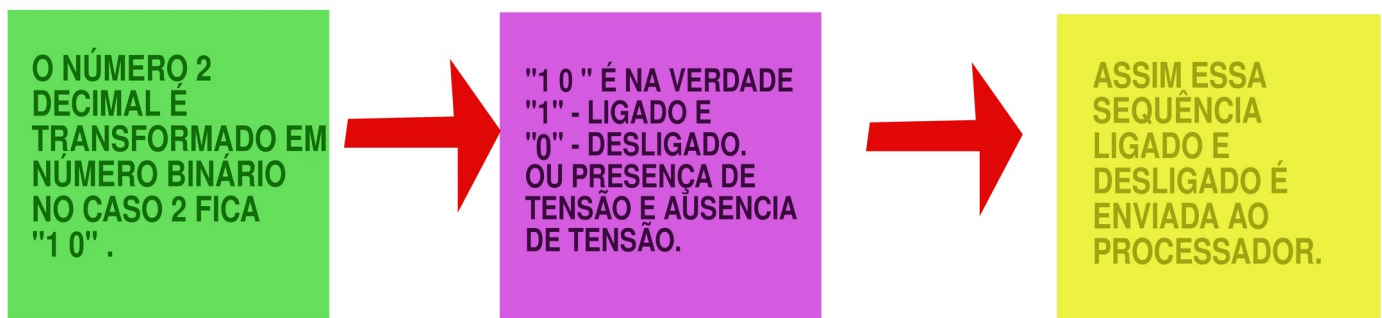
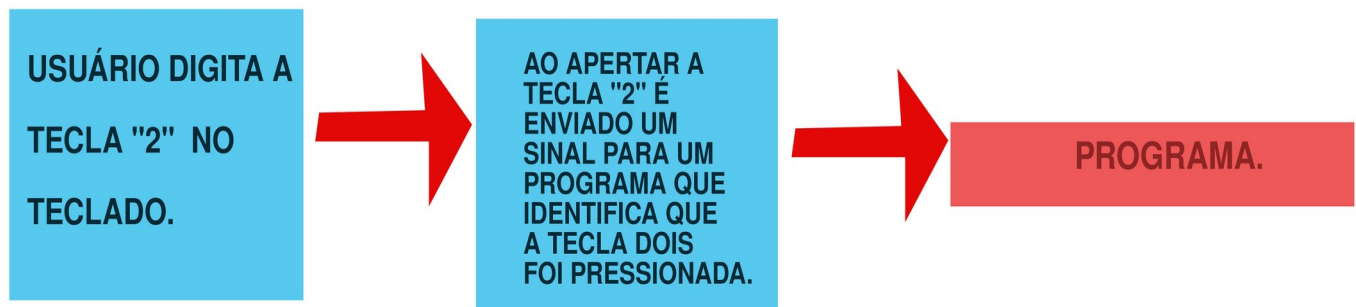
0 = “desligado, não há tensão”

Quem trabalha com “0” e “1”, ligado e desligado é o processador(CPU) onde bilhões de transistors em conjunto processam dados a partir dos estados “**ligado**” e “**desligado**”.

Esse código binário nada mais é que uma forma de transformar comandos escritos em linguagem de alto nível(Python, C, C++) em tensões e correntes que o processador consegue “entender” e realizar as tarefas pedidas.

Exemplo, se eu digito no teclado o número “2” , um programa do computador transforma esse “2” em algo que o processador entenda, o “2” é transformado em código binário para ser analisado no processador e se necessário fazer uma conta, $2 + 2$ por exemplo. Segue um esquema para facilitar o entendimento.

Lembrando que o intuito desse E-book é oferecer os primeiros passos para você aprender Python, criar uma base sólida. Portanto essa pequena introdução não abrange totalmente o tema “como um processador entende comandos digitados no teclado”. Sendo assim indico para quem quer se aprofundar nesse extenso tema, que pesquise na internet, ou busque livros sobre o tema. Palavras chaves para suas pesquisas :
(Como funciona um processador, código binário, compilação e interpretação de código).



Assim comandos utilizados nas linguagens de programação são compilados ou interpretados ou até mesmo ambos e traduzidos em linguagem de máquina(código binário), e executados no processador fazendo o que você programou. Se você ficou confuso não se preocupe, tudo vai ficar mais claro com o decorrer dos capítulos. Não se limite apenas ao nosso E-book, novamente indico fortemente que pesquise mais sobre os assuntos aqui citados.

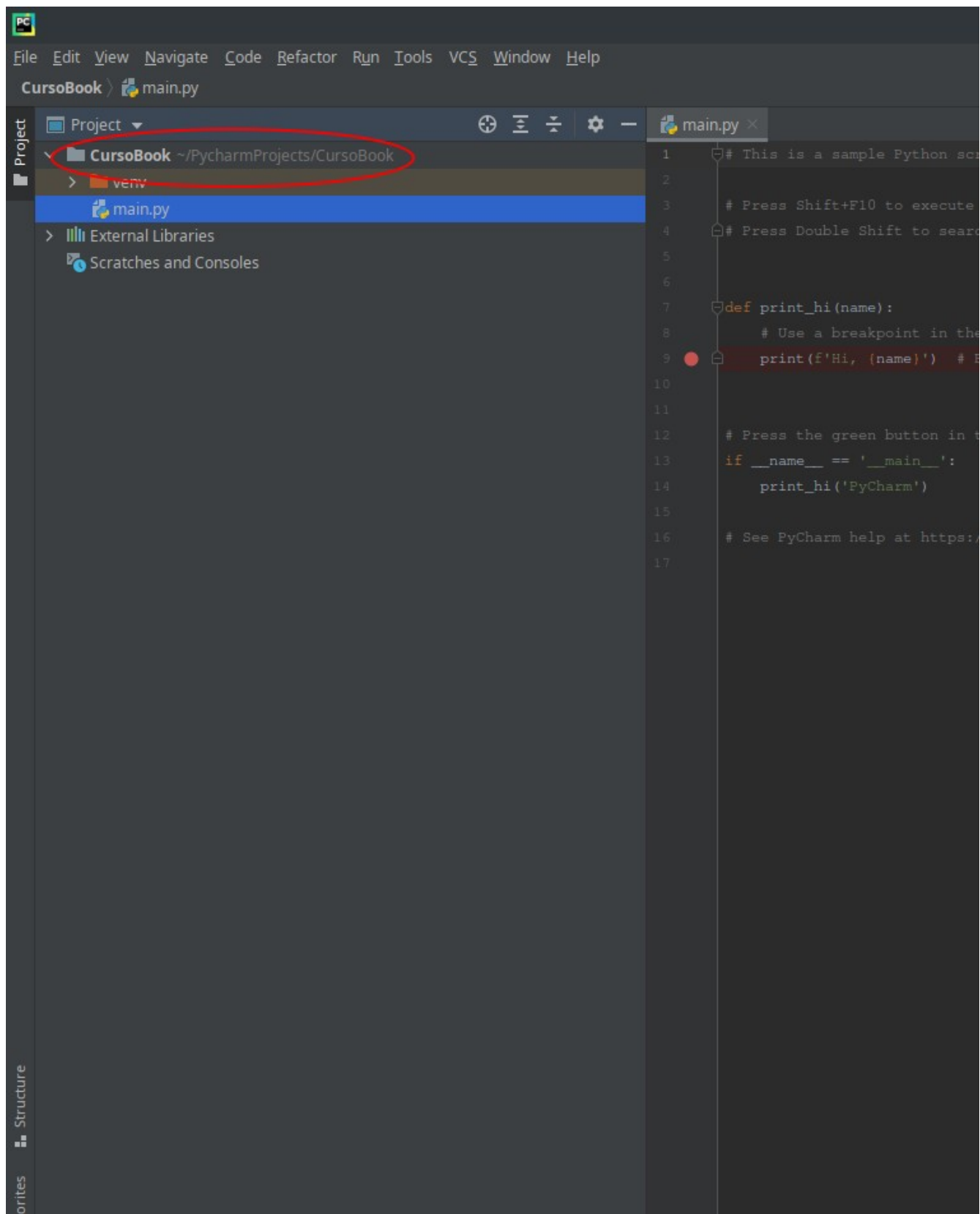
Bom, dado essa breve pincelada sobre linguagens de programação vamos programar nosso primeiro código. Finalmente!

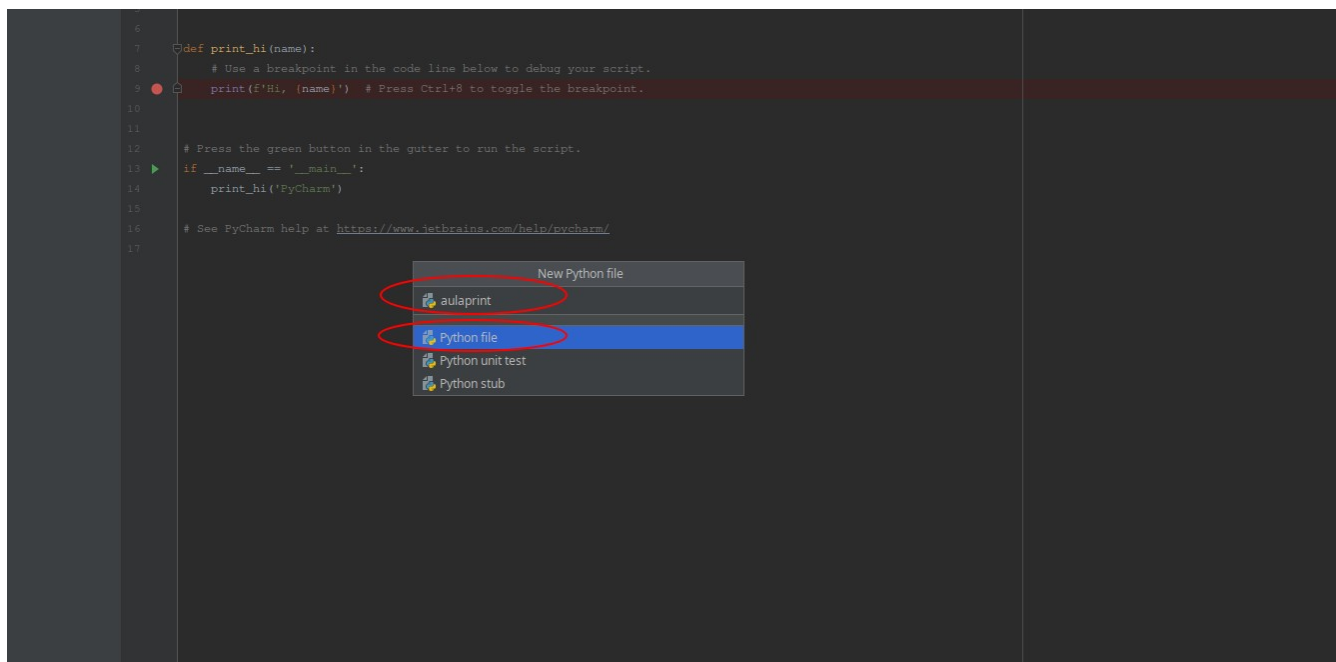
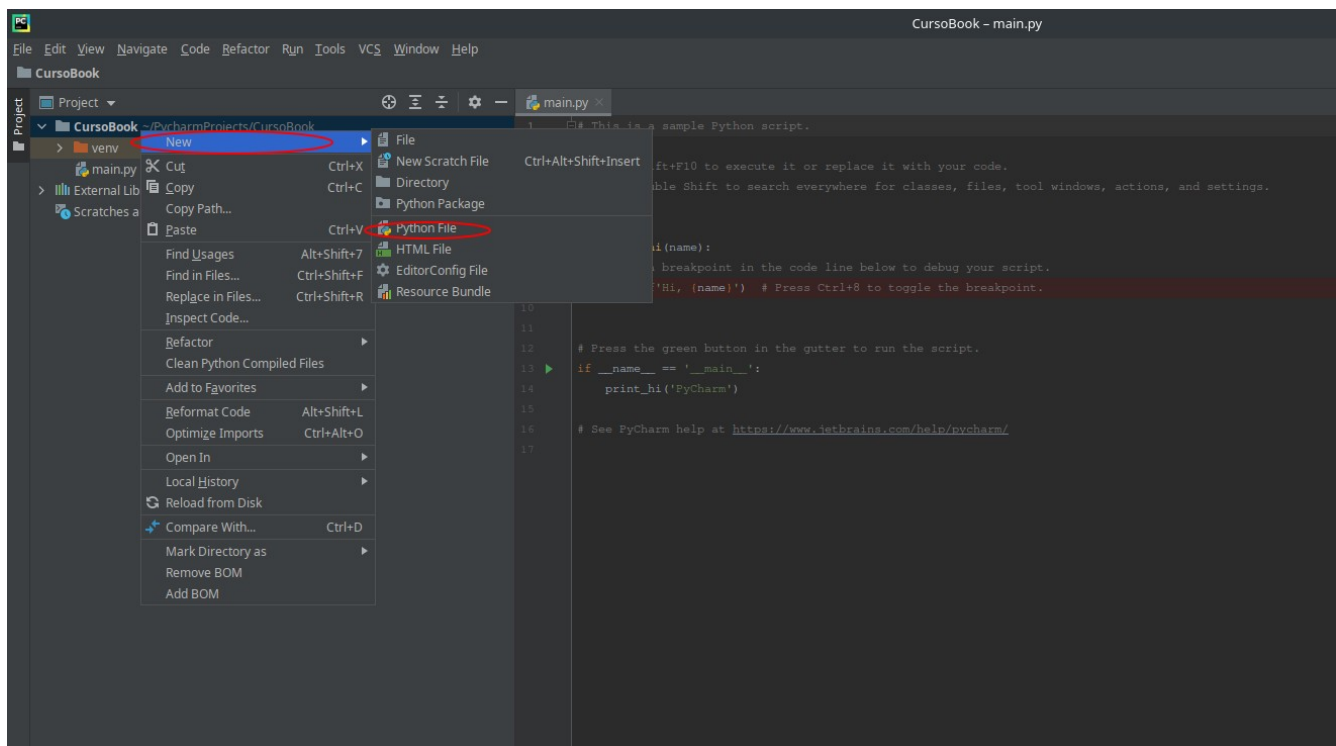
2.2 – Introdução a IDE Pycharm.

Primeiramente abra seu PyCharm e faça as configurações iniciais se ainda não fez, como escolher o tema do Python(indico o tema escuro, seus olhos agradecem). São configurações opcionais então escolha as que você ache melhor.

Crie um novo projeto e escolha um diretório(caminho do projeto no seu computador) que seja fácil encontrá-lo no seu PC, pois ele será útil para os exercícios propostos.

Após criar o novo projeto clique com o botão direito do mouse no nome do seu projeto, depois selecione “**New**”, “**Python File**” e por último de um nome a seu arquivo.py, é aqui onde seu código será salvo. Para cada novo código que você for fazer repita esse processo.





No meu caso coloquei o nome do arquivo.py de “aulaprint”, não esqueça de selecionar a opção “**Python File**”.

2.3 – Formatações mais usadas em Python e seu primeiro código em Python.

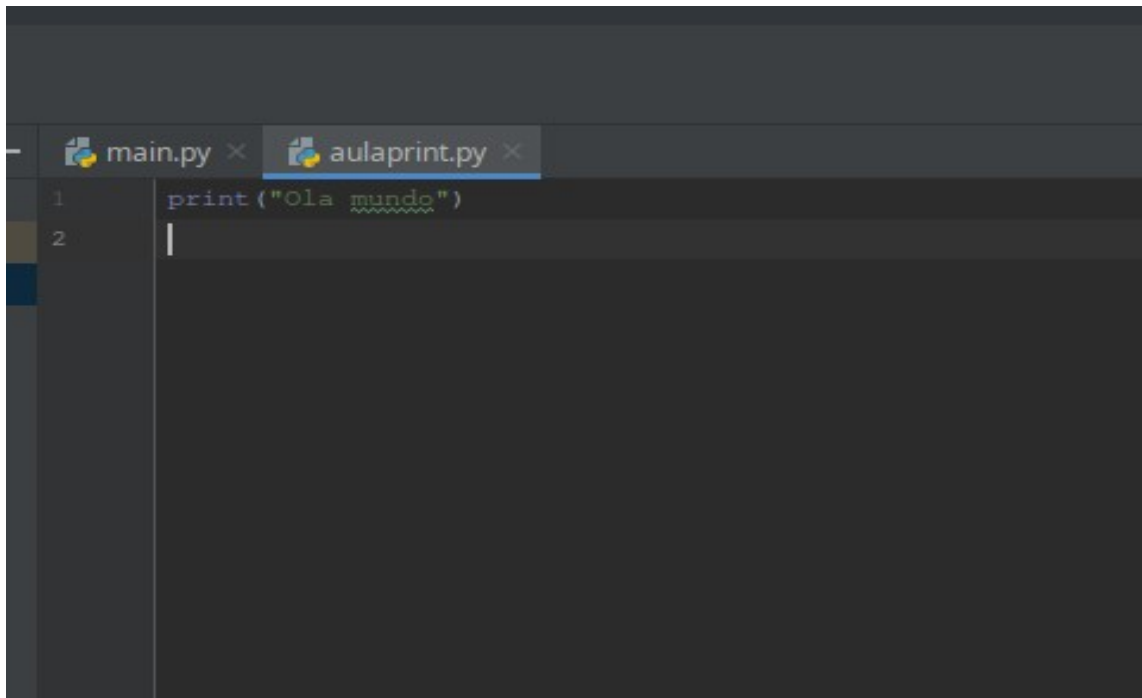
O primeiro código que a gente geralmente aprende em qualquer linguagem de programação é o `print`, ele serve para você “mandar” o computador a imprimir(mostrar na tela) uma palavra.

Em bash shell, uma linguagem muito usada em sistemas operacionais GNU/Linux, utilizamos o “`echo`” para mostrar uma palavra na tela. Em Python usamos o comando “`print`”.

Para imprimir uma frase ou qualquer coisa em Python usa-se a seguinte sintaxe:

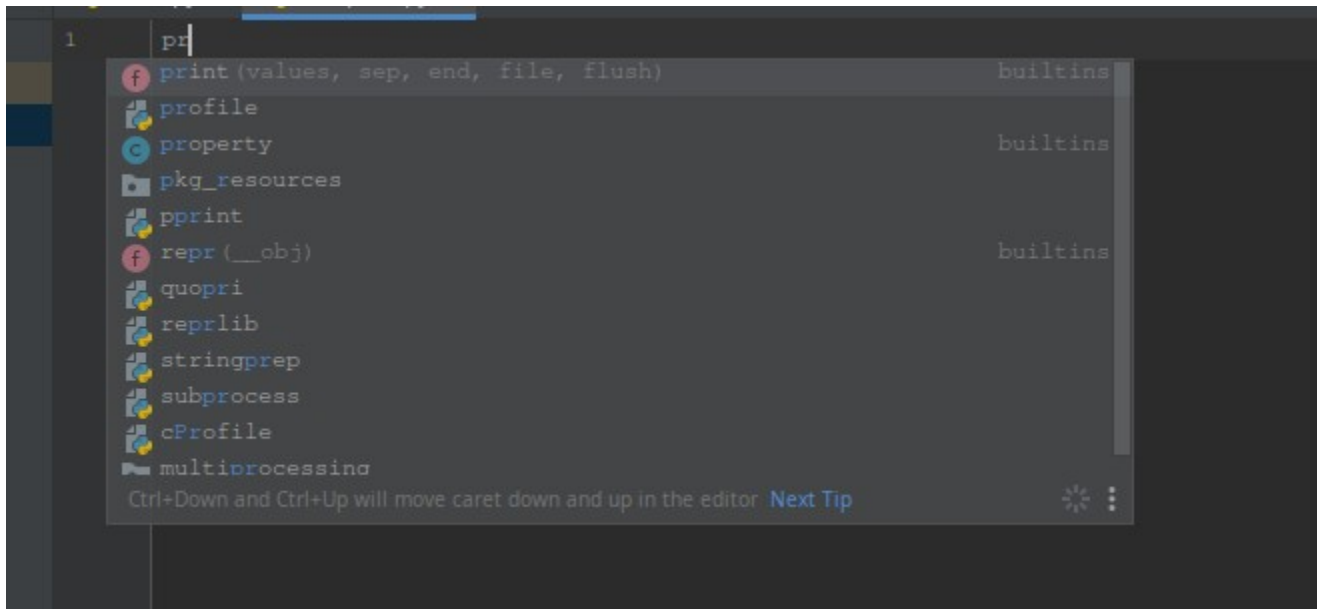
```
print ( “frase ou palavra que você deseja mostrar na tela” )
```

Essa é uma das maneiras de “imprimir” algo na tela, não esqueça as aspas (”) entre as frases e os parêntese depois das aspas. Sem eles o código não funciona. Você pode usar aspas simples (') ou aspas duplas (”) entre as frases. Abaixo um exemplo utilizando essa formatação de print no Python.



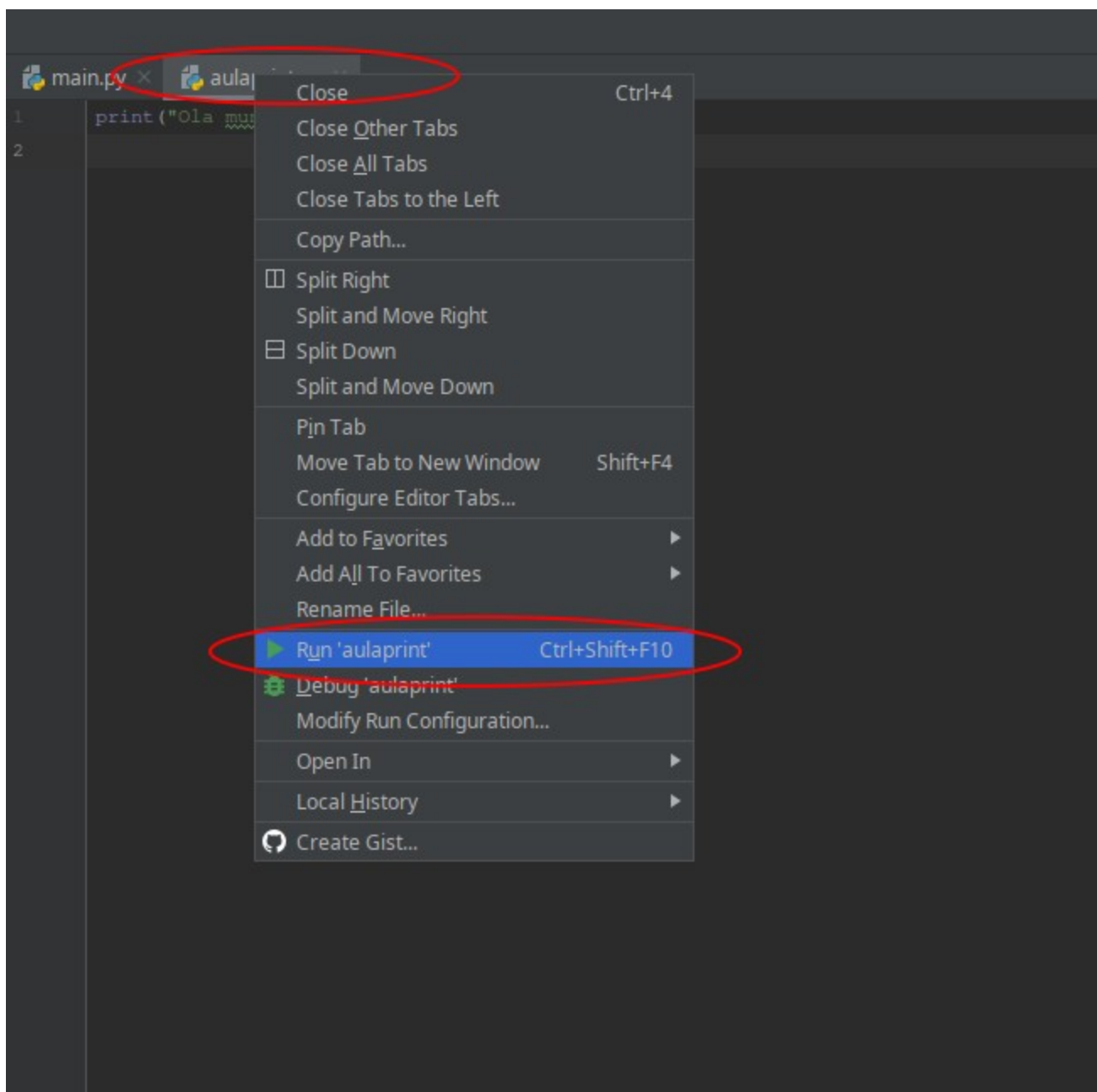
```
main.py x aulaprint.py x
1 print("Ola mundo")
2 |
```

Lembrando de umas das ferramentas incríveis que o Pycharm nos proporciona que é a autocompletagem de código.

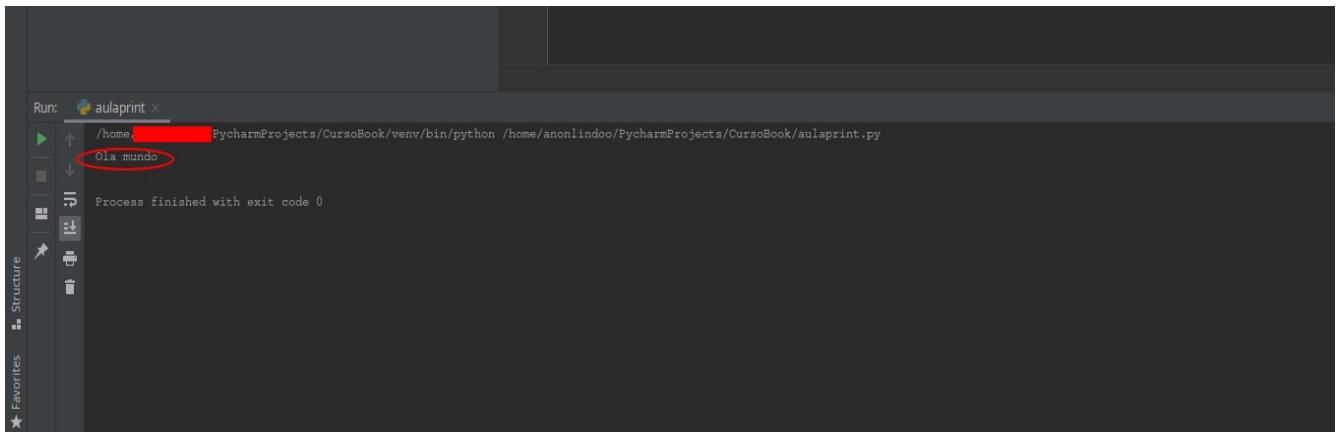


Só clicar em “enter” ou “tab” que o código é completado. Eu indico que no começo para fins de fixação de aprendizado você digite cara parêntese e cada aspas.

Para executar o código, clique com o botão direito do mouse na barra onde está escrito o nome do seu arquivo e depois em run “nome do arquivo”.



A seguinte saída deve aparecer na tela :



```
Run: aulaprint x
/home/.../PycharmProjects/CursoBook/venv/bin/python /home/.../PycharmProjects/CursoBook/aulaprint.py
Ola mundo
Process finished with exit code 0
```

Meus parabéns, você programou seu primeiro código. Vamos agora para algumas outras formatações que serão úteis mais adiante.

Se você **declarar uma variável**(veremos isso no próximo capítulo) e depois quiser utilizar o valor ou palavra declarada no **print** você pode usar as seguintes sintaxes.

numero = 3 “Você pode ler da seguinte forma: A variável **numero** recebe o valor **3**.”

Assim toda vez que você utilizar a variável “**numero**” significa que você estará utilizando o número **3**. Como eu disse no próximo capítulo veremos mais sobre declaração de variáveis e seus tipos, foque apenas em entender como essa formatação de **print** é utilizada. Então se quisermos utilizar o número **3** em um print ou qualquer número ou palavra declarada em uma variável teremos :

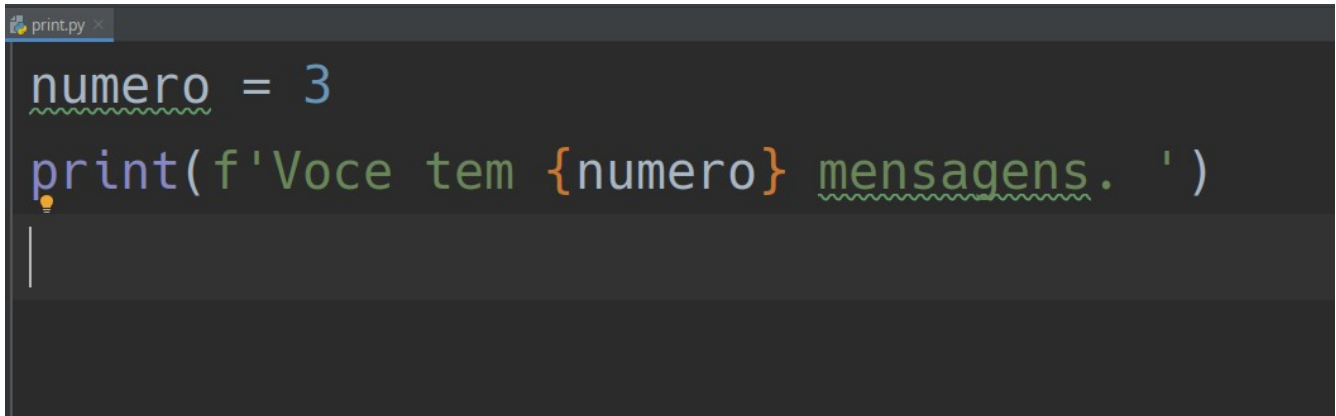
numero = 3 → “Variável declarada.”

print (f“ Você tem {numero} mensagens.”) → “formatação de um print utilizando uma variável dentro do print.”

Ou pode ser usada com aspas simples(**mais indicado**).

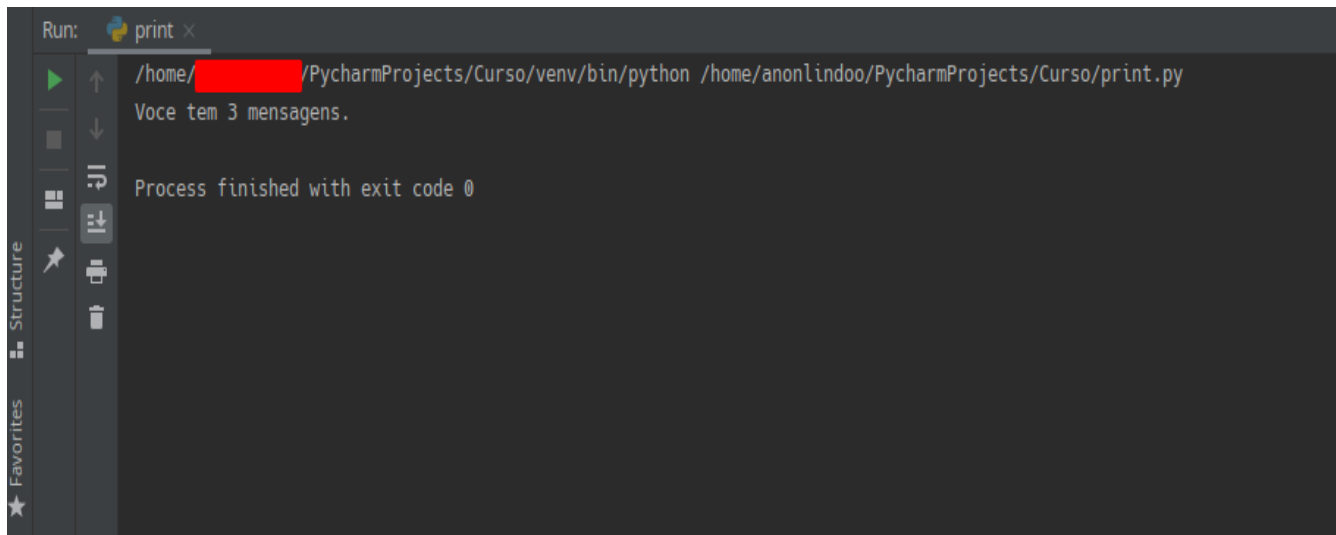
print (f ‘Você tem {numero} mensagens.’)

No PyCharm teremos algo como:



```
numero = 3
print(f'Voce tem {numero} mensagens. ')
```

Entrada do código(as linhas que você digita no programa).



```
Run: print x
/home/[REDACTED]/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/print.py
Voce tem 3 mensagens.

Process finished with exit code 0
```

A saída do programa(mostra o resultado do seu código na tela, no caso mostra o que você pediu para escrever no print).

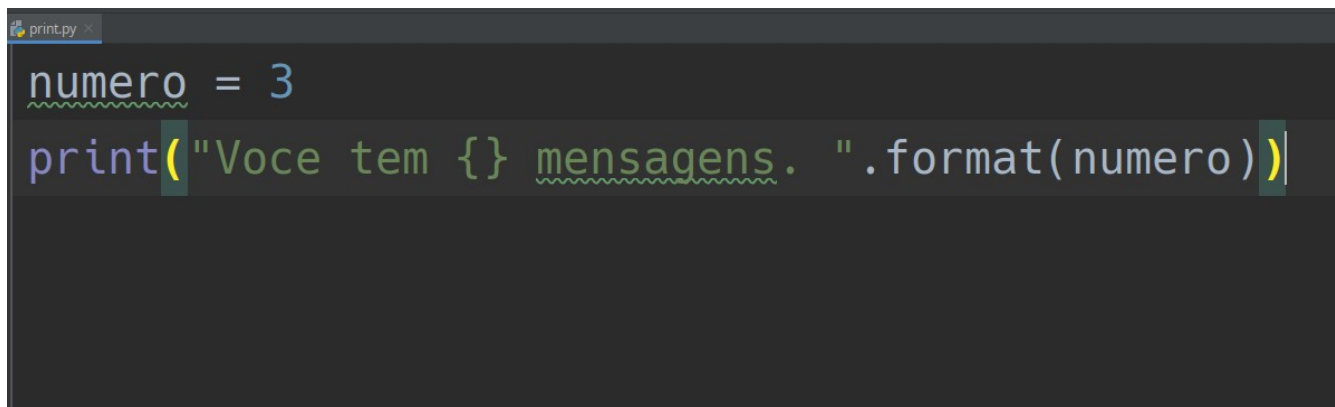
IMPORTANTE!!!

Todas essas formatações mostradas devem ser replicadas por você em seu computador utilizando o PyCharm. É muito importante que você coloque em prática o que leu nesse capítulo. Assim o aprendizado será melhor fixado pelo leitor ou leitora.

O mesmo código pode ser escrito utilizando a formatação mostrada a seguir. Ela utiliza mais caracteres e é útil para determinadas situações que serão exploradas mais para frente no livro, abaixo como sua sintaxe funciona:

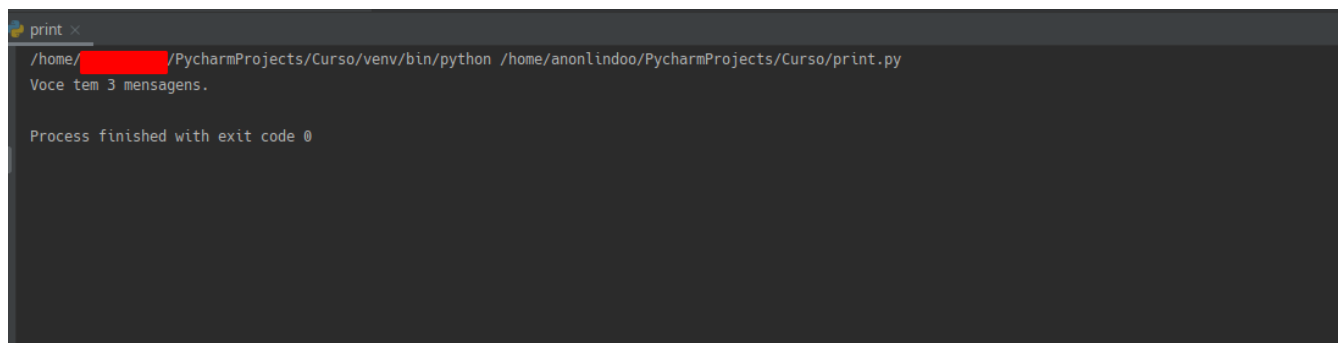
```
numero = 3 → “Variável declarada.”  
print ( “ Você tem {} mensagens.”.format(numero) )
```

No PyCharm teremos algo como:



```
print.py ×  
numero = 3  
print("Voce tem {} mensagens. ".format(numero))
```

Entrada do programa(o código digitado pelo programador/você).



```
print ×  
/home/[redacted]/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/print.py  
Voce tem 3 mensagens.  
  
Process finished with exit code 0
```

Como você pode perceber é a mesma saída da outra formatação.

Saída do programa(seu código mostrado na tela pelo PyCharm).

Se houvesse duas variáveis teríamos algo do tipo:

```
print ( " Você tem {} {} mensagens." .format(numero1 , numero2) )
```

Se fosse três variáveis:

```
print ( " Você tem {} {} {} mensagens." .format(numero1 , numero2 ,  
numero3) )
```

E assim por diante, o mesmo vale quando usamos o

```
print(f'frase{variável1}{variável2}{variável3}')
```

Exercícios do capítulos 2:

1 – Crie um código que mostre na tela a seguinte frase: “Programar requer dedicação”.

2 – Declare uma variável “num = 5” e utilizi-a dentro de uma frase. Usando a formatação com f “palavra{variável}” e utilizando a formatação com .format ambas mostradas no capítulo 2.

3 – Declare duas variáveis “num1 = 3” e “num2 = 4”. Crie uma frase que diga a soma dos dois números, exemplo: “3 + 4 tem resultado igual a 7”. Para esse exercício utilize também as duas formas de formatação mostradas no capítulo 2.

Vale ressaltar que cada variável é declarada apenas uma vez, e pode ser utilizada em quantos prints você quiser.

***As respostas dos exercícios estão no final do livro. Mas não olhe até você ter feito as suas respostas, para aprender a programar tem que fazer, você não aprenderá nada copiando códigos. Se não conseguir não tem problema, olhe a resolução e tente fazer o código novamente.**

Capítulo 3: Como declarar variáveis `int`, `float` e `str`.

Nós tivemos uma palhinha do que é declaração de variáveis no capítulo 2. Vamos agora nos aprofundar mais no assunto.

Mas o que é definitivamente declarar uma variável em Python? Declarar uma variável consiste em “dizer” para o Python que uma palavra significa um número ou uma palavra(string). Exemplo:

`numero = 2` → Estou dizendo para o Python que, sempre que tiver a palavra “numero” quer dizer que estou querendo usar o número 2.

Mas por que não utilizo o “2” de uma vez? Não é mais simples?

Na verdade como você verá mais adiante, quando você declara uma variável, você pode se quiser mudar o valor dela mais adiante no código, ou o mais importante, pedir que o usuário do programa digite o número que ele queria para a variável, no nosso caso a variável “`numero`”.

3.1 – Entendendo e declarando variáveis com o `int`.

Se for necessário criar por exemplo um programa onde o usuário digite o dia do seu aniversário, e esse dia seja printado na tela.

Como fazemos para que ele possa digitar o dia do seu aniversário?

Podemos fazer isso declarando uma variável com `int`. Lembrando que o `int` é para declarar variáveis onde serão inseridos números inteiros, se for necessário inserir um número com vírgula ou uma palavra veremos ainda nesse capítulo.

Como é a sintaxe para que o usuário digite o número que queira?

`dia = int(input())` → Sintaxe para ler um número **inteiro** em Python.

`dia` = Nome da variável que o programador escolhe no código(Você pode escolher qualquer palavra, mas evite acentos e letras maiúsculas. Procure nomes que sejam fáceis de identificar o que é aquela variável para eventuais consultas no código).

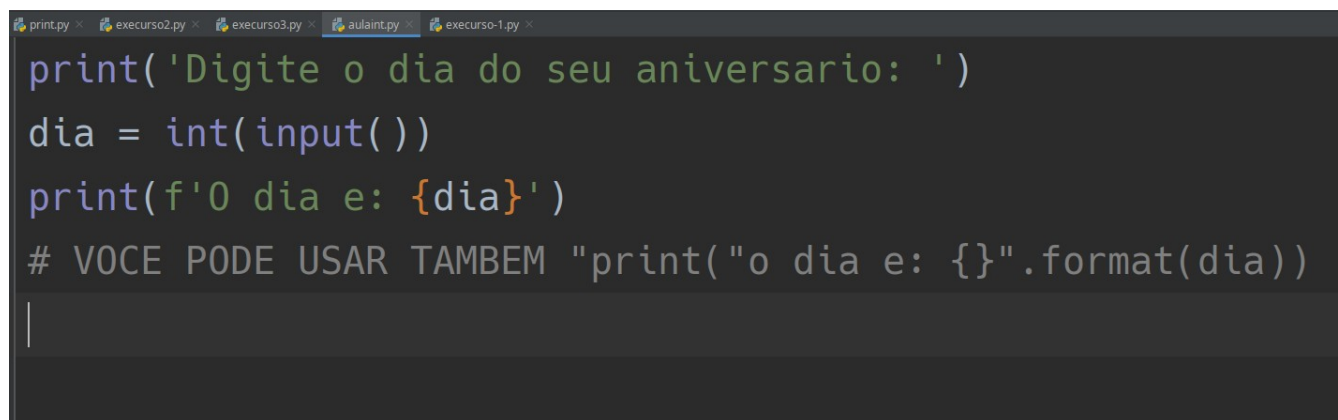
`int` = Aqui você diz que será digitado uma variável do tipo “int”, ou seja uma variável que recebe apenas valores inteiros.

`input` = Esse comando faz com que no visualizador do PyCharm o usuário possa digitar um número. Input vem de entrada, no nosso caso a entrada será um número inteiro digitado pelo usuário.

Vale lembrar que todos os parênteses tem que ser digitados da forma mostrada.

Vale lembrar também que se necessário declarar uma variável sem o auxílio do usuário você pode apenas digitar “variável = número”.

Então um programa como o enunciado seria do tipo:

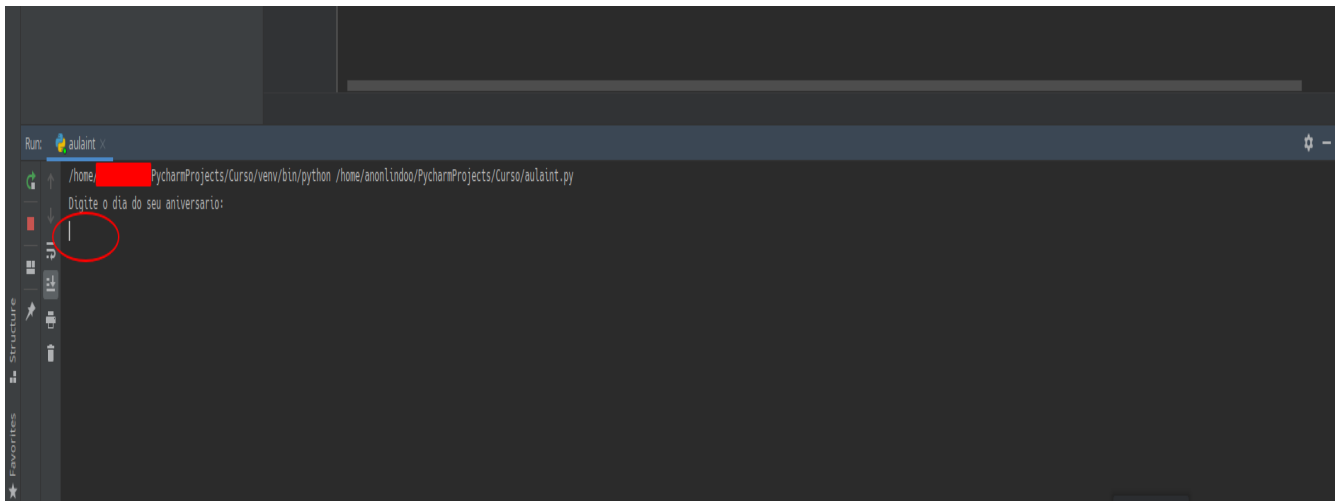
A screenshot of a PyCharm code editor with a dark theme. The editor shows a Python script with the following code:

```
print('Digite o dia do seu aniversario: ')\ndia = int(input())\nprint(f'O dia e: {dia}')\n# VOCE PODE USAR TAMBEM "print("o dia e: {}".format(dia))\n|
```

 The code is color-coded: strings are green, keywords like 'print' and 'int' are blue, and variables like 'dia' are orange. The cursor is at the end of the last line.

Digite esse código em seu PyCharm para aprender ainda mais como ele funciona.

A entrada no PyCharm estará assim:



Veja que onde temos um “|” é onde o usuário digita o dia do seu aniversário.

No programa foi utilizado no começo um “`print('Digite o dia do seu aniversário: ')`”.

Para que o usuário saiba o que deve digitar, porém uma forma de digitar menos e deixar o código mais bonito, podemos escrever o que o usuário deve digitar já na estrutura `dia = int(input())`. Ficaria assim:

```
dia = int(input('Digite o dia do seu aniversário' ))
```

A saída das duas formas será a mesma.



```
Run: aulaint x
/home/[REDACTED]/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/aulaint.py
Digite o dia do seu aniversario:
20
0 dia e: 20
Process finished with exit code 0
```

O número “20” escrito em verde foi digitado pelo usuário. Como dito a saída utilizando `print('Digite o dia do seu aniversario: ')` e utilizando `dia = int(input('Digite o dia do seu aniversario'))` **será a mesma.**

Mais uma vez indico que você refaça esse programa em seu PyCharm.

3.2 – Entendendo e declarando variáveis com o `float`.

Aprendemos a utilizar o `int` para declarar números inteiros. E se quisermos declarar números decimais(números com vírgulas) como por exemplo 2,3, como fazemos? Segue a sintaxe que será usada.

`variavel = float(input())` → Sintaxe para ler um número **decimal ou inteiro** em Python.

Como pode ver a única mudança em relação a sintaxe anterior foi que no lugar de `int`, utilizamos o `float`. Para indicar que estaremos trabalhando com números “flutuantes”, ou números decimais. Quando usamos `float` podemos declarar números decimais e inteiros, mas quando usamos `int` podemos declarar apenas números inteiros.

No sub-capítulo 3.4 veremos como fazer contas em Python utilizando os operados aritméticos, então é importante deixar claro que em Python utilizamos ponto final(.) como separador decimal e não vírgula como usualmente é utilizado no Brasil por exemplo. Assim 2,3 é representado como 2.3.

Refaça o programa anterior que pedia a data de aniversário de uma pessoa, para um programa que peça o peso da pessoa e “printe” o mesmo na tela do programa.

3.3 – Entendendo e declarando variáveis com o **str**.

Sabemos como declarar números inteiros e decimais. Mas e se quisermos declarar palavras? Utilizamos o str e a mesma sintaxe de sempre, só que agora com o str.

variavel = str(input()) → Sintaxe para ler uma **palavra/frase** em Python.

É importante entender o que é “palavra/frase” aqui mencionado. Uma palavra nada mais é que um amontoado de caracteres, uma string(daí o str utilizado na sintaxe, abreviação de string). Quando falamos palavra ou frase você pode declarar números também, já que são caracteres. Então uma string poderia ser do tipo “ Eu gosto de 2 cores, preto e azul”. Como dito números são caracteres, então o programa entenderá que “2” é uma “palavra”.

Declarar variáveis em str é muito útil para coletar informações como nome, sexo cor preferida etc. De algum usuário, como em uma lista de identificação por exemplo.

Novamente peço que refaça o programa anterior, mas agora peça para ler o sexo da pessoa e mostre na tela o que ela digitou.

3.4 – Operadores aritméticos.

Operadores aritméticos como o próprio nome nos diz, são utilizados para cálculos em Python. Primeiro uma lista dos principais operadores aritméticos em Python.

- + **Adição** (2+3)
- **Subtração** (2-3)
- * **Multiplificação** (2*3)
- / **Divisão** (3/2) = 1.5
- // **Divisão** (3/2) → nesse caso o (//) arredonda para baixo ((3//2) = 1).
- % **Resto da divisão** (3%2) → resto é 1.
- ** **Exponenciação** (3**2) = (3²)

Em Python existe ordem de execução de uma expressão matemática, então precisamos respeitar essa ordem em nosso códigos.

Por exemplo nessa expressão qual será o valor obtido?

$$3 + 6 * 6 + 3 / 3$$

Se você pensou 40, está correto(a). Veja que se colocarmos parênteses fica mais fácil de perceber o porquê do resultado.

$3 + (6 * 6) + (3/3)$ → Observe que o resultado seria diferente se utilizarmos os parêntese da forma $((3+(6*6)+3)/3)$.

Então tome cuidado com os parênteses, em suas expressões matemáticas em Python.

3.5 – Operadores relacionais(de comparação).

- == igual**, por exemplo: $2==1$ isso é falso mas $1==1$ ou $2==2$.
- != diferente**, por exemplo $7!=1$ é verdadeiro mas $2!=2$ é falso.
- > maior**, $5 > 1$ ou $8 > 1$ mas $3 > 1$ resulta em falso.
- < menor**, $1 < 9$ ou $5 < 10$ mas $7 < 6$ não é verdade.
- >= maior ou igual**, $4 >= 4$ é verdadeiro, $5 >= 4$ também é verdadeiro.
- <= menor ou igual**, $3 <= 3$, $1 <= 7$ mas $8 <= 3$ é falso.

Tais operadores servem como mostrados, para comparar números variáveis ou strings.

numero1 = 5

numero2 = 4

numero1 == numero2 → Essa comparação resultará em falso(ou False).
Pois $5 \neq 4$ (5 é diferente de 4) ou $5 > 4$ (5 é maior que 4).

Para palavras podemos ter:

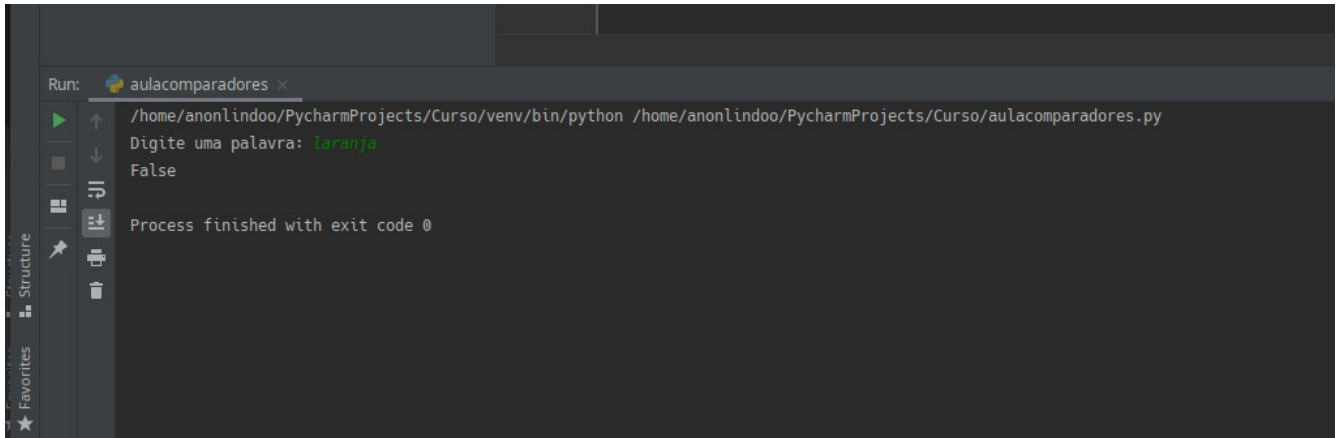
palavra1 = 'M'

palavra1 == M → Essa comparação resulta em verdadeiro(ou True).

Veja um programa que exemplifica o que foi dado até aqui sobre operadores relacionais:

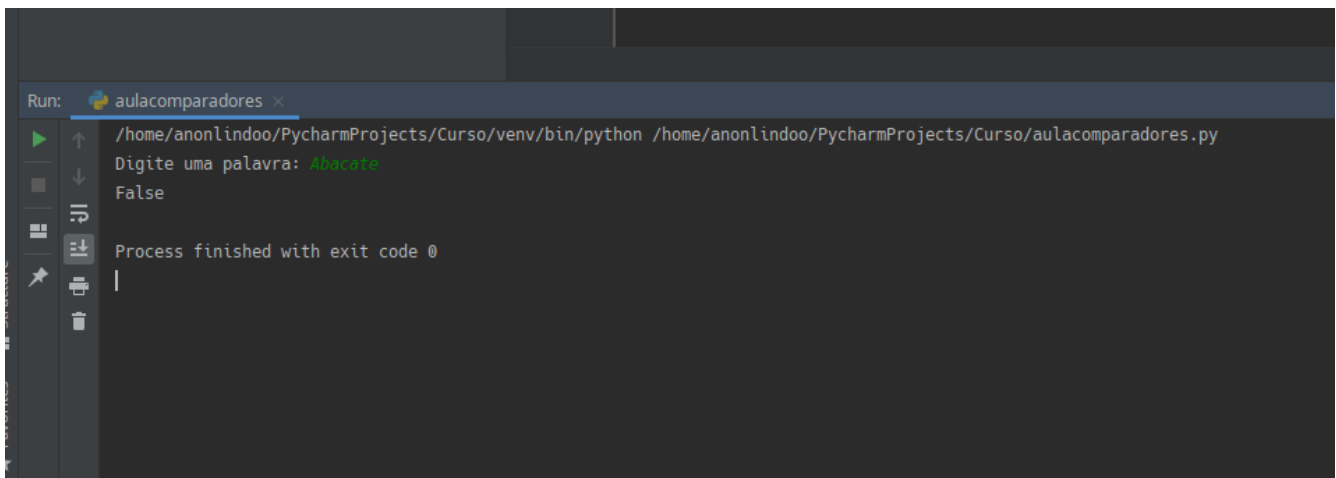
```
aulacomparadores.py x
palavra = 'abacate'
palavrauser = str(input('Digite uma palavra: '))
print(palavra == palavrauser)
```

Se eu digitar outra palavra que não seja “abacate”, aparece “false” no programa, indicando que palavra == palavrauser não são iguais.



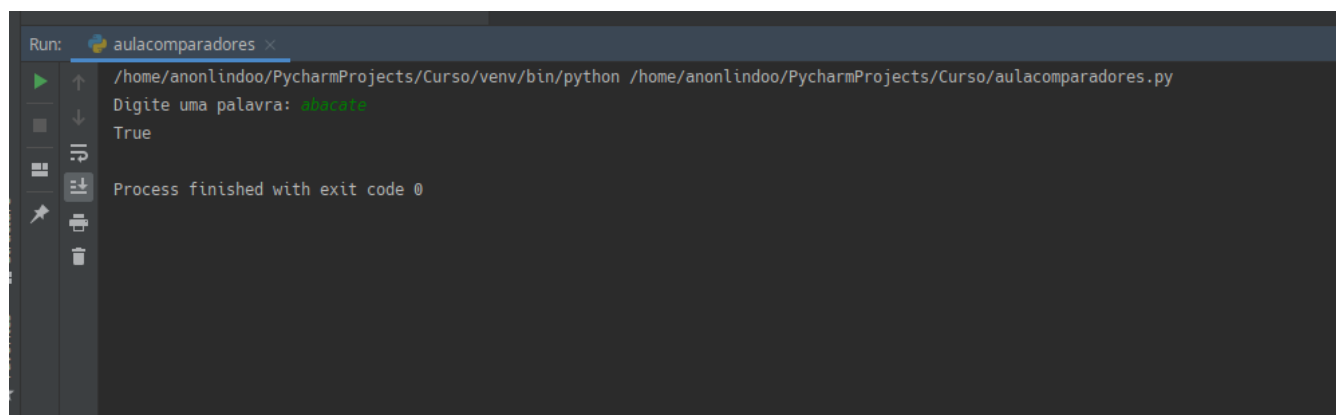
```
Run: aulacomparadores x
/home/anonlindoo/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/aulacomparadores.py
Digite uma palavra: laranja
False
Process finished with exit code 0
```

Mesmo se eu digitar “abacate” com letra “a” inicial maiúscula ele não reconhece como verdadeiro.



```
Run: aulacomparadores x
/home/anonlindoo/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/aulacomparadores.py
Digite uma palavra: Abacate
False
Process finished with exit code 0
```

Só teremos a saída “True” quando eu digitar “abacate” da mesma forma que a variável **palavra** foi declarada. Todas as letras minúsculas.



```
Run: aulacomparadores x
/home/anonlindoo/PycharmProjects/Curso/venv/bin/python /home/anonlindoo/PycharmProjects/Curso/aulacomparadores.py
Digite uma palavra: abacate
True
Process finished with exit code 0
```

O mesmo se aplica a números. No caso de números não se deve declarar: `numero = '1'` e sim `numero = 1` sem aspas. Pois queremos trabalhar com o número 1 e não a string 1.

Indico que refaça o programa mostrado e “brinque” um pouco com ele, para entender ainda mais sobre operadores relacionais.

Exercícios do capítulo 3.

1 – Faça um programa que o usuário digite **3 números decimais ou inteiros** e no final o programa mostre a média dos valores.

2 – Peça que o usuário digite 2 números inteiros no teclado, em seguida coloque os números na seguinte expressão:

```
((numero1 * 6) + (numero2 * 4)) / 5
```

e mostre o valor da expressão.

3 – Crie um programa que peça ao usuário adivinhar uma palavra(dê dicas para que ele possa chegar na palavra). Peça que o usuário digite a palavra toda em letra minúscula, se ele acertar deve aparecer na tela “True” se ele errar “False”.

4 – Faça um programa que leia(peça algo ao usuário) 4 nomes de frutas, dois números decimais e 2 inteiros. Você deve mostrar uma frase aleatória com esses dados.

***As respostas dos exercícios estão no final do livro. Mas não olhe até você ter feito as suas respostas, para aprender a programar tem que fazer, você não aprenderá nada copiando códigos. Se não conseguir não tem problema, olhe a resolução e tente fazer o código novamente. Lembrando que conforme nós avançamos no**

curso o número de exercícios deve aumentar, então não se apavore faça com calma.