Introdução à programação em Python

Neste notebook, vamos aprender alguns elementos básicos de programação em Python.

Lembre-se que para executar uma célula de código, basta clicar nela e digitar `SHIFT+ENTER`.

Você deve ler os textos e ir executando as células de código para fixar o aprendizado.

Variáveis ¶

Uma variável é uma porção de memória que guarda um valor. Ela tem um nome que a identifica.

Uma expressão da forma `variavel = valor` é chamada de **atribuição**.

Comandos de atribuição servem para atribuir valores à variáveis.

In [1]:
```
var = 5   # comando de atribuição (atribuindo o valor 5 à variável "var")
print(var)  # comando de impressão (para mostrar um valor na tela)
```

```
5
```

O conceito de variável em programação é diferente do conceito matemático. Aqui, você pode mudar o conteúdo de uma variável a qualquer momento.

In [2]:
```
var = 134   # valor inteiro
print(var)
var = 4.2   # valor decimal
print(var)
var = "hello world"  # texto
print(var)
```

```
134
4.2
hello world
```

Nomes de variáveis só podem conter letras (maiúsculas ou minúsculas, mas sem acentos), números e o símbolo underline (_) e, além disso, nunca podem começar com número.

Nomes de variáveis devem ser descritivos, mas procure não usar nomes muito grandes.

In [3]:
```
numero = -5
print(numero)
```

```
-5
```

In [4]:
```
nome_professora = "Carla"
print("O nome da professora é", nome_professora)
```

```
O nome da professora é Carla
```

In [5]:
```
Valor1 = 12.4
print(Valor1)
Valor2 = 56
print(Valor2)
```

```
12.4
56
```

In [15]:
```
# 3invalido = 4

invalido = 4
print(invalido)
```

```
4
```

Execute as próximas células e veja um erro comum que pode acontecer:

In [11]:
```
# nome_professora = "Carla Negri Lintzmayer"
# print(nome_profesora)

nome_professora = "Carla Negri Lintzmayer"
print(nome_professora)
```

```
Carla Negri Lintzmayer
```

In [16]:
```
# sala_de_aula = "L502"
# print("A sala de aula é", Sala_de_aula)

sala_de_aula = "L502"
print("A sala de aula é", sala_de_aula)
```

```
A sala de aula é L502
```

Quais foram os erros? **Volte e conserte-os.**

Números

Lidar com números é bem fácil em Python.

Inteiros

Você pode fazer todas as operações básicas com inteiros: adição com símbolo `+`, subtração com símbolo `-`, multiplicação com símbolo `*`, divisão com símbolo `/`, divisão inteira com símbolo `//`, resto de divisão com símbolo `%` e exponenciação com símbolo `**`.

In [17]:
```
print(4 + 67)
```

```
71
```

In [18]:
```
print(9 - 4)
```

```
5
```

In [19]:
```
print(3 * 2)
```

```
6
```

In [20]:
```
# o símbolo "/" indica divisão exata
print(19 / 5)
```

```
3.8
```

In [21]:
```
# o símbolo "//" indica divisão inteira
print(19 // 5)
```

```
3
```

In [22]:
```
# o símbolo "%" indica o resto da divisão
print(19 % 5)
```

```
4
```

In [23]:
```
# o símbolo "**" indica exponenciação
print(3**2)
```

```
9
```

Você pode criar expressões maiores, que utilizem mais de um operador ao mesmo tempo.

Tente adivinhar o resultado da expressão a seguir antes de executá-la:

In [24]:
```
print(4 + 3 * 5 - 8 + 2**3 / 4) # 4 + 15 - 8 + 2  = 13
```

```
13.0
```

O Python executa as operações na seguinte ordem:

- `**`
- `*`, `/`, `//` e `%`
- `+` e `-`

Na dúvida, você pode colocar parênteses para deixar a ordem clara.

In [25]:
```
ordem_padrao = 1 + 5 * 3 # 16
print(ordem_padrao)
```

```
16
```

In [26]:
```
minha_ordem = (1 + 5) * 3 # 18
print(minha_ordem)
```

```
18
```

Note pelos exemplos anteriores que também podemos atribuir resultados de expressões a variáveis.

Faça você mesmo!

Coloque parênteses abaixo de modo que a resposta seja `600`

Dica: Tente fazer com que `2**3-1+2` seja `6` e depois seja multiplicado por `100`.

In [27]:
```
expressao = ((2 ** (3 - 1)) + 2) * 100
print(expressao)
```

```
600
```

Faça você mesmo!

João faz 10 anos de idade hoje, neste exato momento!

Na célula abaixo, calcule quantos segundos de vida ele acabou de comemorar.

Suponha que um ano sempre tem 365 dias, ou seja, ignore os anos bissextos.

In [29]:
```
expressao = ((((60 * 60) * 24) * 365) * 10)
print(expressao)
```

```
315360000
```

Para salvar o seu trabalho, cliquem em `File > Save and Checkpoint` ou no ícone do disquete.

Mas se você estiver usando jupyter em um site, ele não salva arquivos!

Nesse caso, clique em ***File > Download as > IPython Notebook (.ipynb)*** para baixar uma cópia para o seu computador. Baixe cópias frequentemente para não perder o seu trabalho!

Pontos flutuantes

Números de ponto flutuante se referem a números com casas decimais e em geral eles vão se comportar como tais.

In [34]:
```
print(0.1 + 0.1)
print(0.366 + 0.236)
```

```
0.2
0.602
```

In [36]:
```
print(0.54 * 4)
print(0.54 * 0.4)
```

```
2.16
0.21600000000000003
```

Mas pode ser que às vezes o resultado não saia como esperado:

In [2]:
```
print(0.1 + 0.2)
```

```
0.30000000000000004
```

Isso acontece devido à forma como os computadores armazenam os números de ponto flutuante: não tem nada a ver com Python ou com você ter feito algo errado.

Mas não se preocupe com isso, esse exemplo foi apenas para que você não se surpreenda em outros momentos.

Em disciplinas futuras você entenderá isso melhor.

Strings

Conjuntos de caracteres, em linguagens de programação, são chamados de "strings".

Strings em Python são qualquer conjunto de caracteres que estejam dentro de aspas.

In [38]:
```
uma_string = "essa é uma string válida"
print(uma_string)
outra_string = "Essa também, mesmo contendo números (3 5 4) e símbolos (* $ & @ %)"
print(outra_string)
```

```
essa é uma string válida
Essa também, mesmo contendo números (3 5 4) e símbolos (* $ & @ %)
```

O operador `+` funciona em strings como um concatenador: ele une em uma só as strings que estiverem ao seu redor.

In [39]:
```
nome = "Carla"
sobrenome1 = "Negri"
sobrenome2 = "Lintzmayer"
print(nome + sobrenome1 + sobrenome2)
print(nome + " " + sobrenome1 + " " + sobrenome2)
```

```
CarlaNegriLintzmayer
Carla Negri Lintzmayer
```

Mais sobre atribuições

Vimos que variáveis podem receber valores (inteiros, pontos flutuantes ou strings).

Vimos também que elas podem receber resultados de expressões envolvendo esses valores (`soma = 2 + 5`).

Mas variáveis também podem receber resultados de expressões que envolvem outras variáveis. Veja a seguir.

In [40]:
```
x = 3
f = 2 * x + 4
print(f)
```

```
10
```

No código anterior, `x` é utilizada na expressão que é atribuída a `f` . Como `x` tem um valor inteiro atribuído à ela, faz sentido multiplicá-la por `2` e então somar esse resultado com `4` .

Mais especificamente, o que acontece é o seguinte:

- primeiro é calculado o valor da expressão `2 * x + 4` , onde `x` está bem definido com o valor `3`
- em seguida guarda-se esse resultado na variável `f`

Agora veja o código a seguir:

In [41]:
```
x = 7
print(x)
x = x + 4
print(x)
```

```
7
11
```

Achou essa expressão mais estranha que a anterior?

Pois o que está acontecendo é exatamente a mesma coisa do exemplo anterior: primeiro é calculado o valor da expressão `x + 4` (onde `x` claramente vale 7) e então esse resultado é atribuído à variável `x` (pois uma variável pode receber qualquer valor).

Assim, o novo valor de `x` é o valor anterior que ela tinha somado com o valor 4.

Execute o exemplo a seguir para ver um erro comum que pode acontecer:

In [44]:
```
resultado = 8
expressao = resultado**2 + 4

print(expressao)
```

```
68
```

Moral da história: você pode usar variáveis em expressões, contanto que as variáveis estejam definidas (isto é, que elas tenham valores atribuídos a elas anteriormente).

Exemplo

Vamos fazer o cálculo de conversão de temperatura de Celsius para Fahrenheit.

Se C é o valor da temperatura em graus Celsius, então $\frac{9}{5}C + 32$ é a temperatura em Fahrenheit.

Veja a seguir um programa que faz esse cálculo.

In [52]:
```
C = 55
# a expressão a seguir converte uma temperatura de Celsius para Fahrenheit
F = C * 9/5 + 32
print(F)

C = 100
# a expressão a seguir converte uma temperatura de Celsius para Fahrenheit
F = C * 9/5 + 32
print(F)

C = 100
# a expressão a seguir converte uma temperatura de Celsius para temperatura absoluta(K)
TK = C + 273
print(TK)

TK = 373
# a expressão a seguir converte uma temperatura de Kelvin para temperatura Celsius(°C)
C = TK - 273
print(C)
```

```
131.0
212.0
373
100
```

Mas e se você quisesse converter 65 em vez de 55? Ou 39, ou 27?

Basta voltar na linha em que você definiu `C = 55` e mudar para `C = 65` (ou qualquer outro valor desejado) e reexecutar a célula.

Faça você mesmo!

Preencha a célula abaixo com os dados que você quiser para calcular a área de um trapézio cuja base menor é b, a base maior é B e a altura é h.

Preencha também a fórmula para a área, que é dada por

$$\frac{(b + B) \cdot h}{2}$$

Não se esqueça de colocar todos os operadores (por exemplo o `*`).

In [54]:
```
b = 6
B = 9
h = 4
area = ((b + B) * h) / 2
print(area)
```

```
30.0
```

Outro exemplo

No futuro, será útil fazer troca de valores entre variáveis.

Por exemplo, se temos as variáveis `x` com valor `4` e `y` com valor `-9`, vamos querer realizar uma troca de forma que `x` fique com o valor `-9` e `y` fique com o valor `4`.

Se fizermos `x = y`, então estaremos atribuindo o valor `-9` a `x` e a primeira parte do problema está resolvida:

In [55]:
```
x = 4
y = -9
print("O valor de x é", x, "e o valor de y é", y)
x = y
print("O valor de x é", x, "e o valor de y é", y)
```

```
O valor de x é 4 e o valor de y é -9
O valor de x é -9 e o valor de y é -9
```

Mas agora o valor anterior de `x` (o `4`) se perdeu!

Precisamos então de uma variável auxiliar, que guarde o valor anterior de `x` enquanto ele é atualizado para o novo valor.

In [56]:
```
x = 4
y = -9
print("O valor de x é", x, "e o valor de y é", y)

# salve na variável "aux" o valor atual de x:
aux = x # aux é igual a 4
# agora atualize o valor de x:
x = y # x é igual a -9
# o valor de y recebe o antigo valor de x, que está salvo em "aux":
y = aux # y é igual a 4
print("O valor de x é", x, "e o valor de y é", y)
```

```
O valor de x é 4 e o valor de y é -9
O valor de x é -9 e o valor de y é 4
```

Entrada e saída

Todo programa de computador

- recebe dados de alguma fonte (teclado, mouse, arquivo, etc.),
- processa esses dados e,
- envia o resultado do processamento para algum lugar (tela, arquivo, impressora, web, etc.).

Então, quando tratamos do fluxo de dados por um programa de computador, é comum classificar os dados como

- dados de entrada: são aqueles que o programa **recebe**
- dados de saída: são aqueles que programa **mostra** ao usuário

Mostrando dados (saída)

A forma mais comum de mostrar dados ao usuário é usando a função `print()`, que já estamos usando.

In [ ]:
```
C = 55
F = C * 9 / 5 + 32
print(F)
```

Ela pode receber vários valores entre vírgulas para imprimir.

In [1]:
```
C = 55
F = C * 9 / 5 + 32
print(C, "graus Celcius equivale a", F, "Fahrenheit")
```

```
55 graus Celcius equivale a 131.0 Fahrenheit
```

Recebendo dados (entrada)

É importante também que você aprenda a fazer perguntas ao usuário e saiba receber a reposta que ele dará.

Para pedir que o usuário entre com um dado de texto, use a função `input()`. Veja os exemplos:

In [4]:
```
nome = input("Qual é  seu nome? ")
print("Olá,", nome)

sorvete = input("Qual o sabor do sorvete? ")
print("Sabor " + sorvete+ " saindo")
```

```
Qual é  seu nome? luis
Olá, luis
Qual o sabor do sorvete? uva
Sabor uva saindo
```

No código a seguir, tente entender o erro antes de prosseguir com a explicação:

In [5]:
```
idade = input("Qual é a sua idade? ")
dobro = idade + idade
print("O dobro da sua idade é", dobro)
```

```
Qual é a sua idade? 27
O dobro da sua idade é 2727
```

Se você digitou o valor, por exemplo, 18, então a resposta foi 1818 (ao invés de 36).

O que acontece é que o Python assume que a entrada é uma string, independente do conteúdo.

E o que fazer se queremos lidar com números ao invés de texto?

Basta converter o texto para um número:

In [3]:
```
idade_texto = input("Qual é a sua idade? ")
idade = int(idade_texto)
dobro = idade + idade
print("O dobro da sua idade é", dobro)
```

```
Qual é a sua idade? 36
O dobro da sua idade é 72
```

A linha `idade = int(idade_texto)` converte idade_texto para o tipo inteiro e guarda o resultado em `idade`.

Escolhemos converter para inteiro pois consideramos idade como sendo um número inteiro.

Se você precisar que o dado lido seja um número decimal, você pode fazer o seguinte:

In [7]:
```
temp_texto = input("Qual é a temperatura atual? ")
temp = float(temp_texto)
dobro = temp + temp
print("O dobro da temperatura atual é", dobro)
```

```
Qual é a temperatura atual? 25
O dobro da temperatura atual é 50.0
```

Faça você mesmo!

Escreva na célula abaixo um programa que auxilia na hora de fazer uma venda. Seu programa deve:

1. perguntar e receber o preço do produto;
2. perguntar e receber o número de parcelas em que o cliente deseja pagar (sem juros);
3. mostrar o valor de cada parcela.

In [2]:
```
preco = float(input("Qual é o preço do produto? "))

parcelas = int(input("Em quantas parcelas você deseja pagar? "))

valor_parc = preco / parcelas

print("O valor de cada parcela será: ",valor_parc)
```

```
Qual é o preço do produto? 600
Em quantas parcelas você deseja pagar? 5
O valor de cada parcela será:  120.0
```

Listas

Uma lista é uma coleção de valores quaisquer, mantidos em ordem. Ela é armazenada em uma única variável.

In [4]:
```
lista = {'primeiro', 'segundo', 'terceiro'}
print (lista)
```

```
{'terceiro', 'primeiro', 'segundo'}
```

Apesar de uma lista poder manter valores de qualquer tipo, é interessante que eles tenham alguma relação entre si.

In [3]:
```
aula_bases = ["terça-feira", 8, "L502", "Carla"]
```

Os valores são mantidos em ordem, como dissemos antes, e por isso eles podem ser acessados por meio de índices inteiros.

Em várias linguagens de computação, incluindo Python, o primeiro elemento da lista sempre está na posição 0.

De forma geral: o i-ésimo elemento armazenado está na posição i-1.

In [3]:
```
print(lista)
print(lista[2])
print(lista[0])
print(lista[1])
```

```
['primeiro', 'segundo', 'terceiro']
terceiro
primeiro
segundo
```

In [6]:
```
print("A aula de bases computacionais acontece toda", aula_bases[0], "na sala", aula_bases[2])
```

```
A aula de bases computacionais acontece toda terça-feira na sala L502
```

Listas numéricas

No futuro, nos será útil criar listas que contêm apenas números.

Por exemplo, a lista a seguir contém os 10 primeiros números inteiros.

In [2]:
```
primeiros_10 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(primeiros_10)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

É até fácil criá-la, mas e se precisássemos de uma lista com os 100 primeiros números inteiros? E os primeiros 1000?

Para isso, o Python fornece o comando `range()`.

Seu resultado não é exatamente uma lista, mas podemos usar o comando `list()` para convertê-lo (como fizemos com `int()` e `float()` sobre o resultado do `input()`).

In [9]:
```
x = 1
y = 16
primeiros_15 = list(range(x,y+1))
print(primeiros_15)
print()
primeiros_30 = list(range(1,30))
print(primeiros_30)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
```

Note que `range(x,y)` cria uma lista com valores entre `x` e `y-1` (e não `y`).

In [4]:
```
lista = list(range(6,17))
print(lista)
```

```
[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

Se especificarmos um único valor, `range(x)`, o que teremos é uma lista com valores entre `0` e `x-1`.

In [8]:
```
lista = list(range(8))
print(lista)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

Podemos ainda especificar um terceiro valor, que indica qual a diferença entre dois números da lista.

Isto é, `range(x,y,z)` cria uma lista que só possui valores entre `x` e `y-1`, começando em `x` e seguindo a cada `z` unidades.

In [6]:
```
lista = list(range(6,17,2))
print(lista)
lista = list(range(6,30,5))
print(lista)
```

```
[6, 8, 10, 12, 14, 16]
[6, 11, 16, 21, 26]
```

Agora é a sua vez!

Resolva a lista de exercícios dessa aula para fixar melhor todo esse conteúdo!

Boa sorte e não se esqueça de ir ao horário de atendimento da professora caso esteja com dúvidas ou tenha algum problema.