Aula 01
Introdução ao Python

Instalando o Python

- Windows : https://python.org.br/instalacao-windows/
- Linux : https://python.org.br/instalacao-linux/
- Mac OS X : https://python.org.br/instalacao-mac/

Usando o Python

O Python pode ser utilizado de diversas formas. As duas mais comuns são :
- Python Shell : ele é acessado pelo Prompt de Comando digitando python;
- Script Python : arquivo com extensão .py;

Python Shell

Para acionarmos o Python Shell, usamos o comando python no Prompt de Comando (python3 no Terminal do Linux ou do Mac).

Os comandos passados no `Python Shell` tem por característica serem executados logo que acionamos o Enter.

A versão do Python indicada no Shell representa a versão instalada na máquina e os `>>>` indicam onde os comandos são entrados.

Para sairmos do Shell, usamos a função **exit()**.

Arquivo Python

A outra forma de usar o Python é através de arquivos com a extensão ***.py***.

Quando usamos o arquivo Python (chamados de script ou módulo), os comandos colocados lá são executados apenas quando chamamos o python no Prompt de Comando.
Imagine um script chamado **resposta_da_vida.py** que, quando chamado, vai exibir a resposta para o sentido da vida.

C:\Users\gutoh>python resposta_da_vida.py
Qual é a resposta da vida, universo e tudo mais?
A resposta é : 42!

```
print('Qual é a resposta da vida, universo e tudo mais?')
print('A resposta é : 42!')
```

Comentários

Os comentários em Python são usados para explicar o código e ajudar outras pessoas a entender o que ele faz. Eles são ignorados pelo interpretador Python e não afetam a execução do programa.
Os comentários em linha são usados para incluir breves explicações ou observações sobre um pedaço de código. Eles são indicados por um sinal de # e tudo o que você escrever após o sinal # será considerado como comentário e será ignorado pelo interpretador Python.

Aqui estão alguns exemplos de comentários em linha :

```
x = 42 # atribui o valor 42 à variável x
y = 5 # atribui o valor 5 à variável y

# calcula a soma de x e y e armazena o resultado em z
z = x + y

# imprime a mensagem "Bom dia, Dave!"
print("Bom dia, Dave!") # a saída será "Bom dia, Dave!"
```

Os comentários em linha também podem ser usados para desabilitar uma parte do código temporariamente. Isso é conhecido como comentar o código e é feito adicionando um sinal de # na frente de cada linha de código que você deseja desabilitar.

```
# x = 42
# y = 5
# z = x + y
```

É importante notar que os comentários em linha só são úteis se você colocá-los em lugares estratégicos, e não escreva comentários que são óbvios ou desnecessários.

Por exemplo, não precisamos de um comentário para explicar que “x = 5 é uma atribuição”, isso é óbvio. Ao invés disso, seria mais útil colocar comentários para explicar “o que a variável x representa ou qual é o propósito do código”.

Além disso, os comentários também podem ser usados para explicar as decisões de design que você tomou no código, como por exemplo, por que você escolheu usar um determinado algoritmo ou estrutura de dados.
Mais tarde veremos sobre os "comentários de várias linhas".

Função print()

A função print() é uma das funções mais básicas do Python e é usada para exibir informações na tela. Ela pode ser usada para exibir uma ou mais strings, variáveis, números ou outros tipos de dados. A sintaxe da função é a seguinte :

print(argumento_1, argumento_2, ..., argumento_n)

Os argumentos podem ser qualquer tipo de dados, incluindo strings, números, variáveis, etc.

```
7  # uso do print com apenas um argumento
6  print('Olá, mundo!')
5  print(123)
4  x = 4
3  print(x)
2
1  # uso do print múltiplos argumentos
8  print('A resposta', 'é', 42)
```

Repare que cada print imprime em linhas diferentes e sempre insere um espaço entre os argumentos. Podemos alterar esse comportamento usando os argumentos nomeados.

Argumentos Nomeados “sep” e “end”

A função print aceita certos argumentos nomeados, que vão servir para alterar alguns funcionamentos dela.
- “sep” : altera o caracter que será inserido entre os argumentos. O padrão é o espaço;

```
1  # usamos o argumento nomeado sep= para inserir traços em vez de
   espaços
1  print('sou um', 'belo', 42, sep='-')
```

- “end” : controla o que será inserido ao final do print. Por padrão é inserido um “\n” (nova linha);

```
1  # alteramos que, ao final do print seja inserido uma exclamação
1  print('sou uma bela string ', end='!')
2  print('eu deveria estar na segunda linha')
```

O "\n” é um comando especial das strings que é usado para criar uma nova linha.

```
1  print('eu deveria\nestar todo\nna mesma\nlinha')
```

Variáveis

Temos que ter em mente que toda linguagem de programação deve ser capaz de aceitar, armazenar e nomear dados.

Nosso programa deve ser capaz de receber dados do teclado (ou de outra parte do seu programa) e associar a um nome aquele dado. Este dado pode ser um valor simples ou múltiplos valores que são

associados a um nome. Dados que são associados a um nome e que guarda dado é chamada de variável. Pense na variável como uma caixa com um nome do lado de fora dela e que pode armazenar o que quisermos dentro.

O Python tem muitas formas diferentes de armazenar listas de dados, que vamos ver mais tarde. É conveniente associar nomes aos dados. Se eu quiser armazenar uma idade em Python, eu poderia digitar **idade = 24000**. Se eu quiser armazenar um nome, eu poderia digitar **nome = 'Gandalf'**.

Algumas Regras

Temos algumas regras que temos que seguir quando damos nomes às nossas variáveis.

As variáveis podem iniciar com uma letra ou _ (sublinhado / underscore). Depois do primeiro caracter, podemos usar números como n1. Você não pode usar espaços em nomes de variáveis, “meu_nome” é permitido, mas “meu nome” não! Variáveis também não podem ter acentuação ou cedilha nos nomes, “acucar” é permitido, mas “açúcar” não!

É considerado uma boa prática separar as palavras com sublinhado.
- “minha_idade” é uma boa prática;
- “minhaIdade” é uma prática ruim;

Há diversas palavras-chave (keywords) que não podemos usar como nomes de variáveis no Python :

pass if elif else while for break continue and or not in is import from as try except raise finally with del print def return yield del global class exec lambda assert

Experimente executar um código onde você associa algum valor a uma variável com algum dos nomes acima e veja a mensagem de erro.

O Python é uma linguagem de programação de tipagem dinâmica e forte.
- dinâmica : pois uma variável pode assumir diferentes tipo ao longo de sua execução;
- forte : pois o Python é capaz de indentificar o tipo;

Tipo string

Strings é um tipo de dado caracterizado por ser um texto que que inicia com ' ou " e termina com as mesmas aspas. Ela pode conter letras, números e outros caracteres especiais.

```
print('sou uma string')
print("sou outra string")
```

Caso precisemos usar aspas simples ou aspas duplas dentro da string, podemos usar de duas formas :
- usar a string oposta;
- usar `\` (contra-barra) como caracter de escape;

```
print("sant'anna")
print('"Nós nunca realmente crescemos, nós apenas aprendemos a agir em público." - Bryan White')

nome = 'sant\'anna'
print(nome)
```

Também podemos usar uma string no formato `raw` (crua). Ele ajuda a mostrar o caracter de escape.

PS.: originalmente o uso seriam apenas para expressões regulares, mas podemos usar em strings normais.

```
print(r'c:\User\fulano\Desktop')
print(r'c:\\User\\fulano\\Desktop')
print(r'sant\'anna')
```

Tipos int e float

Em Python, nós temos distinção entre os tipos numéricos. Os principais são os tipo inteiro (`int`) e tipo ponto flutuante (`float`).

int

O tipo inteiro é caracterizado por ser um número positivo ou negativo, sem valor decimal.

```
print(42)  # int
print(0)  # int
print(-42)  # int
```

float

O tipo ponto flutuante são os números positivos ou negativos, mas com valor decimal, mesmo que seja zero.

```
print(3.14)  # float
print(0.0)  # float
print(-3.14)  # float
```

Diferente do que estamos acostumados, na programação **SEMPRE** iremos utilizar um ponto para separar a parte decimal de um número. A vírgula é usada para separar itens.

Operações Aritiméticas com int e float

É possível realizar operações matemáticas básicas com inteiros, como soma, subtração, multiplicação, divisão (sempre irá retornar um float), potência, floor division e módulo.

Alguns exemplos :

```
x = 5
y = 10
resultado = x + y  # 15
resultado = x - y  # -5
resultado = x * y  # 50
resultado = x / y  # 0.5
resultado = x ** y # 9765625
resultado = x // y # 0
resultado = x % y  # 5
```

Também é possível utilizar operadores de atribuição combinada, como +=, -=, *=, /= e %=.

Alguns exemplos :

```
x = 5
y = 10
x += y  # x = 15
x -= y  # x = -5
x *= y  # x = 50
x /= y  # x = 0.5
x %= y  # x = 5
```

Exercícios

Para todos os exercícios abaixo, mostre os resultados com o print()

1. Imprima a frase "Olá, mundo!" na tela usando a função print.
2. Crie uma variável inteira x e imprima seu valor usando a função print.
3. Crie uma variável float y e imprima seu valor usando a função print.
4. Imprima o resultado da operação 2 + 2 usando a função print.
5. Imprima o resultado da operação 10 - 5 usando a função print.
6. Imprima o resultado da operação 2 * 3 usando a função print.
7. Imprima o resultado da operação 10 / 2 usando a função print.
8. Imprima o resultado da operação 10 // 3 usando a função print.
9. Imprima o resultado da operação 10 % 3 usando a função print.
10. Crie uma variável idade com sua idade. Imprima a idade usando a função print.
11. Crie uma variável altura com sua altura em metros. Imprima a altura usando a função print.
12. Crie uma variável peso com seu peso em quilogramas. Imprima o peso usando a função print.
13. Crie uma variável mensagem com a string "Bem-vindo ao meu programa!". Imprima a mensagem usando a função print.
14. Crie uma variável inteira x e atribua o valor 10 a ela.
15. Crie uma variável float y e atribua o valor 2.5 a ela.
16. Multiplique x por y e armazene o resultado em uma nova variável chamada z.
17. Imprima o valor de z.
18. Crie uma variável inteira a e atribua o valor 5 a ela.
19. Crie uma variável float b e atribua o valor 3.2 a ela.
20. Divida a por b e armazene o resultado em uma nova variável chamada c.
21. Imprima o valor de c.
22. Crie uma variável inteira d e atribua o valor 7 a ela.
23. Crie uma variável float e e atribua o valor 4.0 a ela.
24. Crie uma variável inteira g e atribua o valor 12 a ela.
25. Crie uma variável float h e atribua o valor 2.5 a ela.
26. Divida g por h e armazene o resultado em uma nova variável chamada i.
27. Crie uma variável inteira k e atribua o valor 15 a ela.
28. Crie uma variável float l e atribua o valor 1.75 a ela.
29. Multiplique k por l. Armazene o resultado em uma nova variável chamada m.
30. Crie uma variável inteira n e atribua o valor 20 a ela.
31. Crie uma variável float o e atribua o valor 0.5 a ela.
32. Eleve o a n e armazene o resultado em uma nova variável chamada p.
33. Crie uma variável inteira q e atribua o valor 25 a ela.
34. Crie uma variável float r e atribua o valor 2.0 a ela.
35. Eleve r a q. Armazene o resultado em uma nova variável chamada s.
36. Crie uma variável inteira t e atribua o valor 30 a ela.
37. Crie uma variável float u e atribua o valor 3.5 a ela.
38. Divida t por u. Armazene o resultado em uma nova variável chamada v.
39. Crie uma variável nome com seu nome completo e uma variável idade com sua idade. Imprima uma frase com esses valores usando a função print.
40. Crie duas variáveis num1 e num2, atribua um valor para cada uma e mostre o resultado de todas as operações aritiméticas básicas com elas.