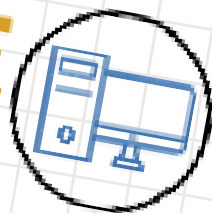


CODE
LAB TEEN

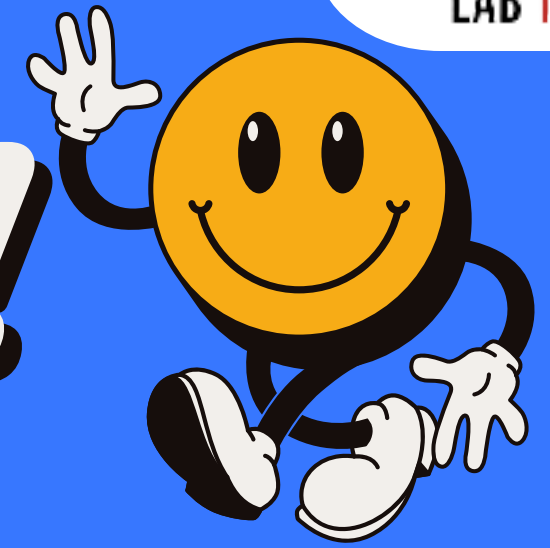


PYTHON AULA 11





BEM - VINDOS!



AGENDA

Hoje vamos revisar todo o conteúdo que foi visto até agora em Python:

REVISÃO-PRINT



A principal saída que temos é o comando de imprimir na tela, `print()`.

Para imprimir texto usamos aspas ("") dentro do `print`.

Mas também podemos manipular a saída para colocarmos o valor de variáveis no meio do texto. A forma mais simples disso, é fazer:

✓
0s



```
print("olá mundo")
```

olá mundo

```
x = 11
```

```
print("Exemplo aula",x,"de python")
```

Exemplo aula 11 de python

EXERCÍCIO

A seguir, há diferentes alternativas. Diga qual alternativa que NÃO irá resultar na seguinte saída:

"Olá! Meu nome é Ana. Eu tenho 25 anos e moro em São Paulo."

a)

```
nome = 'Ana'
```

```
idade = 25
```

```
cidade = 'São Paulo'
```

```
apresentacao = f'Olá! Meu nome é  
{nome}. Eu tenho {idade} anos e  
moro em {cidade}.'
```

```
print(apresentacao)
```

b)

```
nome = "Ana"
```

```
idade = 25
```

```
cidade = "São Paulo"
```

```
apresentacao = f"Olá! Meu nome é  
(nome). Eu tenho (idade) anos e  
moro em (cidade)."
```

```
print{apresentacao}
```

c)

```
nome = "Ana"
```

```
idade = 25
```

```
cidade = "São Paulo"
```

```
apresentacao = f"Olá! Meu nome é  
{nome}. Eu tenho {idade} anos e  
moro em {cidade}."
```

```
print(apresentacao)
```

REVISÃO-VARIÁVEIS



Variáveis **guardam** alguma informação para serem usadas nos códigos em sequência.

- 1 - São definidas com o **nome** e o **valor**, por exemplo, `variável = 1`;
- 2 - **Não** podem iniciar com números;
- 3 - **Não** podem ter caracteres especiais (ex: `'`, `@`, `#`...);
- 4 - **Não** podem ter espaços.

✓
0s

```
[3] variavel_1 = 1
    print("variavel 1: ", variavel_1)

    variavel_2 = variavel_1 + 1
    print("variavel 2: ", variavel_2)

    variavel 1:  1
    variavel 2:  2
```

REVISÃO-VARIÁVEIS



Por causa dessas regras, podemos dar nomes como:

```
a = 1  
A = 2
```

```
i = 3  
I = "String"
```

```
_nome = "Aluno"
```

```
Staphylococcus32_alfa = True
```

REVISÃO-ENTRADA E SAÍDA*

Saídas são todas as informações que o programa mostra para o usuário. A principal forma de saída que utilizamos é o comando `print()`, como revisamos anteriormente.

Entradas são as informações que o programa recebe do usuário. Utilizamos a função `input()` para capturar essas entradas.

REVISÃO-ENTRADA E SAÍDA*

Usamos a função `input()` para receber um valor e armazenamos isso em uma variável. Quando o Python encontra a função `input()`, ele pausa a execução do programa e espera o usuário digitar. Assim que o usuário pressiona Enter, a função `input()` retorna o que foi digitado como uma string.



```
idade = input("Digite sua idade: ")
```

Digite sua idade:

REVISÃO-ENTRADA E SAÍDA*

As variáveis recebidas pela função de entrada `input()` são por padrão do tipo `string` (`String` é um texto e pode ser qualquer coisa, como: letras, números e símbolos). Para converter uma entrada para inteiro, podemos usar: `int(input())` ou `int(variável)`

```
idade = int(input("Digite sua idade: "))
```

```
idade = input("Digite sua idade: ")  
idade = int(idade)
```

Dessas duas maneiras a variável `idade` vai ser um inteiro



REVISÃO-ENTRADA E SAÍDA*

Importante: Lembre-se de que, se a variável é uma string, não podemos fazer contas aritméticas sem convertê-la para inteiro (ou outro tipo numérico) antes.

```
❗ 3s ▶ idade = input("Qual sua idade: ")
    print(idade+10)

👤 Qual sua idade: 22
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-3-45c3360c48fc> in <cell line: 2>()
      1 idade = input("Qual sua idade: ")
----> 2 print(idade+10)

TypeError: can only concatenate str (not "int") to str
```

No exemplo, tentamos somar a variável idade com 10 e tivemos erro, pois elas são de tipos diferentes, e só podemos fazer operações entre duas variáveis do mesmo tipo.

REVISÃO-ENTRADA E SAÍDA*

Então, para fazermos uma operação aritmética precisamos mudar nossa variável para um tipo numérico.

```
idade = int(input("Qual é sua idade:"));  
print(idade + 10);
```

```
Qual é sua idade:13  
23
```

```
idade = input("Qual é sua idade:");  
idade = int(idade);  
print(idade + 10);
```

Dessas duas maneiras funcionam as operações

REVISÃO-ENTRADA E SAÍDA*

Em variáveis do tipo texto (String), quando executada a operação com o operador '+', isso será concatenar textos

```
✓ 0s ▶ variavel = "texto_1 " + " texto_2"  
print("Essa é minha variavel: ", variavel)  
  
Essa é minha variavel: texto_1 texto_2
```

No exemplo, fazemos uma concatenação de duas Strings "texto_1" e "texto_2" e imprimimos, lembrando que podemos fazer isso porque elas são do mesmo tipo.

Concatenar
significa juntar!



EXERCÍCIO

A seguir, há três alternativas para um código que converte uma string para inteiro e realiza uma soma com ele. Diga qual está INCORRETA:

a)
`numero_str = "10"
numero_int = numero_str(int)
soma = numero_int + 5
print("O resultado da soma é:",
soma)`

b)
`numero_str = "10"
numero_int = int(numero_str)
soma = numero_int + 5
print("O resultado da soma é:",
soma)`

c)
`numero_str = '10'
numero_int = int(numero_str)
soma = numero_int + 5
print('O resultado da soma é:',
soma)`

EXERCÍCIO

Crie um programa em Python que:

- Peça ao usuário para digitar seu nome.
- Peça ao usuário para digitar sua idade.
- Peça ao usuário para digitar a cidade em que nasceu.

Exiba uma mensagem na tela usando essas informações.

A mensagem deve ser exatamente assim, substituindo (x) pelos dados de entrada:

"Olá, me chamo (x), tenho (x) anos e nasci em (x)."

REVISÃO-REPETIÇÃO

O While repete os comandos enquanto uma condição ainda não foi cumprida:

```
while x < 10:
```

```
while x > 10:
```

```
while != < 10:
```

O While tem uma escrita simples, mas é muito importante que se lembre de atualizar o “x” dentro do while, caso contrário o programa fica em Loop eterno

REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range (0, 10, 2)
```




**ESSA É A VARIÁVEL
QUE VAI SER
ATUALIZADA
CADA VEZ QUE O
FOR SE REPETE**

REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range(0, 10, 2)
```



**ESSE É O MÉTODO QUE
DEFINE QUAIS NÚMEROS
VAMOS USAR**

REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range (0, 10, 2)
```

**AQUI ESTÁ OS NÚMEROS QUE
DEFINEM A NOSSA REPETIÇÃO**



REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range (0, 10, 2)
```

O PRIMEIRO NÚMERO
SIGNIFICA QUAL O
VALOR INICIAL DA
VARIÁVEL



REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range (0, 10, 2)
```

**O SEGUNDO NÚMERO É
O VALOR QUE A
VARIÁVEL NUNCA VAI
PASSAR**




REVISÃO-FOR

Temos o For, que possui uma escrita mais difícil, mas que não precisamos ficar pensando em atualizar a variável pois ele faz isso sozinho.

```
for i in range (0, 10, 2)
```

**O TERCEIRO NÚMERO É O
QUANTO A VARIÁVEL
VAI SER ATUALIZADA**



O For possui uma variação de escrita. Caso o programador não coloque o terceiro número, o programa considera como se existisse o número 1 no lugar.

REVISÃO-FOR

Nesse código podemos ver que a variável “i” vai começar sendo 0 e cada vez que o For se repetir vai somar +2 na variável “i”, enquanto “i” for menor do que 10

```
for i in range (0, 10, 2)
```

REVISÃO-FOR



```
for i in range (0, 10, 2):  
    print(i, "É par")
```

Execução	1	2	3	4	5
i	0	2	4	6	8

ESSE CÓDIGO DENTRO DO FOR É EXECUTADO ENQUANTO "i" É MENOR DO QUE 10. SABENDO QUE A CADA VEZ QUE É EXECUTADO ELE SOMA +2, SABEMOS QUE É EXECUTADO 5x

REVISÃO-FOR



Resultado do programa:

```
for i in range (0, 10, 2):  
    print(i, "É par")
```

```
0 É par  
2 É par  
4 É par  
6 É par  
8 É par
```


REVISÃO-FOR

Nós vimos até agora exemplos com os números de forma positiva, mas também podemos usar números negativos com o For:

```
for i in range (-10, -20, -2):  
    print(i, "É par")
```

REVISÃO-FOR



```
for i in range (-10, -20, -2):  
    print(i, "É par")
```

Execução	1	2	3	4	5
i	-10	-12	-14	-16	-18

ESSE CÓDIGO DENTRO DO FOR É EXECUTADO ENQUANTO "i" É MAIOR DO QUE -20. SABENDO QUE A CADA VEZ QUE É EXECUTADO ELE SOMA -2, SABEMOS QUE É EXECUTADO 5x

REVISÃO-FOR



Resultado do programa:

```
for i in range (-10, -20, -2):  
    print(i, "É par")
```

```
-10 É par  
-12 É par  
-14 É par  
-16 É par  
-18 É par
```

REVISÃO-EXERCÍCIO

Imagine que você irá ganhar uma mesada de R\$14,00 por semana, e por curiosidade, quer rodar um código para calcular o quanto de mesada irá juntar em 12 semanas.

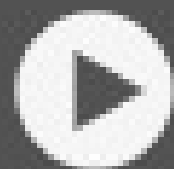
Após executar esse código, qual vai ser o valor da variável "mesada", após 12 semanas?

- a) 154
- b) 176
- c) 168
- d) 182
- e) Nenhuma das anteriores

```
mesada = 0
for i in range (1,13):
    mesada = mesada + 14
print(mesada)
```

EXERCÍCIO

Dado o código abaixo, qual será a sua saída?



```
for variavel in range(1, 10, 2):  
    print(variavel)
```

- a) 2, 4, 6, 8, 10
- b) 1, 3, 5, 7, 9
- c) 2, 4, 6, 8
- d) 0, 2, 4, 6, 8

OBRIGADO!

Contem para gente o que você achou da aula de hoje:



<https://forms.gle/3XM1Fv7jUEZSHQku6>

CODELAB TEEN