RESUMO PYTHON (3)

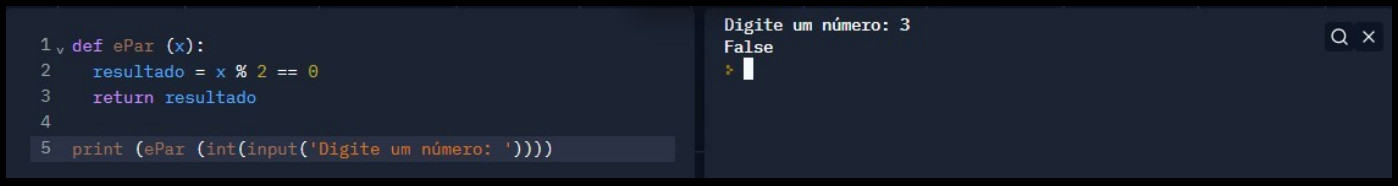
Interpretador online: https://replit.com/languages/python3;
Recomendado: Thonny (https://thonny.org/)

FUNÇÕES

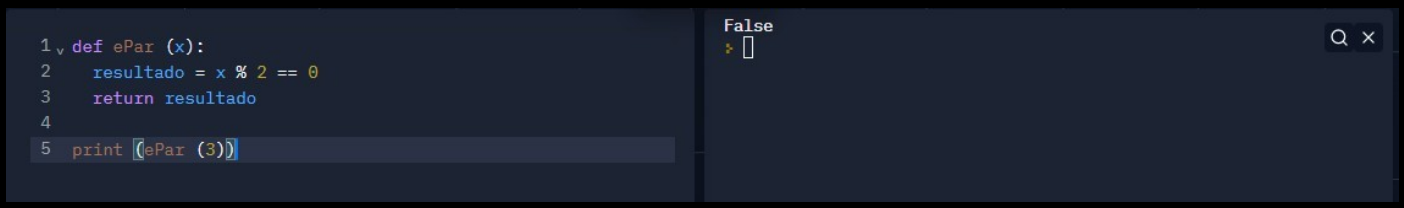
Um conjunto de ações. Sempre possui parênteses no final! Ex: int(), print (), input ().

```
def [nome da função] (parâmetros):
        ações a serem realizadas
        return [o que você deseja obter]
```

PARÂMETROS: O que são os **parâmetros**? São as informações necessárias para o programa rodar. Por exemplo, essa é uma função com um parâmetro. O que isso significa? Que para sabermos se um número é par ou ímpar só precisamos desse próprio número.



OBS: da maneira acima o usuário digita o número. No exemplo abaixo o próprio programador coloca seu número. Observe que ao colocarmos o nome da função e dentro dos parênteses um número, o parâmetro x da função adquire o valor que foi escolhido.



Segue um exemplo de função que possui 2 parâmetros, desejamos fazer a soma de dois números. Logo, são necessárias 2 informações para o programa ser executado.

RETURN: se observarmos, ao final de toda função há um comando 'return'. Assim indicamos qual informação desejamos obter como 'resposta' da função. Logo, no primeiro e no segundo exemplo desejamos obter se é verdade ou mentira que o número é par. Já no terceiro exemplo precisávamos saber a soma dos números.

OBS: algumas funções não possuem retorno, logo aparece 'none' como resposta. Um exemplo é o print (), que não retorna nada, somente exibe algo na tela.

MÓDULOS

Quando construímos um código com funções, podemos criar uma biblioteca, e essas funções podem ser usadas em outros códigos. Funciona simplesmente estando na mesma pasta.

Então como podemos chamar essas funções de outros documentos?

Import [nome do doc]

Dessa forma salvamos todo o código e todas as funções do doc.

OBS: o documento deve estar salvo como '.py'

video funcoes 3.py | exemploModulo.py | video funcoes externas].py

```
"""MÓDULOS
1. O QUE É UM MÓDULO
2. USANDO MÓDULOS

"""
import exemploModulo

texto = exemploModulo.funcao1()
texto2 = exemploModulo.funcao2()
```

Logo, vemos que o arquivo exemploModulo.py possui duas funções especificadas 'dentro' de si, a funcao1 e a funcao2

MAS certas vezes não é vantajoso importar todas as funções de um arquivo. Logo podemos importar somente uma função específica:

```
from exemploModulo import funcao2
```

Nesse caso estamos importando somente a funcao2 do arquivo exemploModulo. Logo, se chamarmos a funcao1, dará erro.
OBS: dessa forma também estamos importando tudo:

```
from exemploModulo import *
```

MUDANDO O NOME DA FUNÇÃO – conseguimos mudar o nome da função após importá-la, usando o 'as'

```
from exemploModulo import funcao1 as f1
```

OBS: sempre colocamos a documentação da função logo no início dela, comentada, para caso outras pessoas forem usar nossas funções elas saberem como funcionam.

```
def funcao1():
    """documentação da função"""
    return 'você chamou a funcão 1 do módulo'
```

EXEMPLO + PRÁTICO:

funcao_primos.py | capac_função.py

```python
def ePrimo (n):
    divisor = 1
    num_de_divisores = 0
    if (n == 0) or (n == 1):
        resposta = 'Não é primo!'
    else:
        while divisor <= n:
            res = n % divisor == 0
            if res == True:
                num_de_divisores = num_de_divisores + 1
            divisor = divisor + 1
        if num_de_divisores == 2:
            resposta = 'É primo!'
        else:
            resposta = 'Não é primo!'
    return resposta

#print (ePrimo (int (input ())))

```

Essa é uma função que nos responde se o número inserido é primo ou não.
Agora queremos usar essa função em outro arquivo:

funcao_primos.py | capac_função.py

```python
from funcao_primos import *

a = ePrimo (3)
print (f'{a}')
```

Shell

```
>>> %Run 'capac_função.py'

  É primo!
```

OU

funcao_primos.py | capac_função.py *

```python
import funcao_primos

a = funcao_primos.ePrimo (3)
print (f'{a}')
```

Shell

```
>>> %Run 'capac_função.py'

  É primo!
```

O python já possui bibliotecas prontas para uso, como o print (), int(), ... Não precisamos importá-las. Porém nesse site (https://docs.python.org/3/library/index.html) há diversas outras funções disponíveis. As funções built in já estão prontas para uso, enquanto as outras precisam ser importadas.

Por exemplo, estaremos usando a função built in abs(), que retorna o módulo de um número.

abs(*x*)

Return the absolute value of a number. The argument may be an integer, a floating point number, or an object implementing `__abs__()`. If the argument is a complex number, its magnitude is returned.

funcao_primos.py × capac_função.py ×

```
1  n = -2
2  print (abs(n))
```

Shell ×

```
>>> %Run 'capac_função.py'

  2
```

Outro exemplo, agora de uma função que ainda não está disponível.

`math.` **factorial**(*x*)

Return *x* factorial as an integer. Raises `ValueError` if *x* is not integral or is negative.

Deprecated since version 3.9: Accepting floats with integral values (like `5.0`) is deprecated.

funcao_primos.py × capac_função.py ×

```
1
2  from math import factorial
3
4  print (f'{factorial (4)}')
5  print (f'{factorial (5)}')
```

Shell ×

```
24
120
```