

Linguagem de Programação Python

Modularização II

Professor: Ritormar Torquato

1

Modularização II

Objetivos: Funções embutidas, importação e criação de módulos; a passagem de parâmetros; execução de programas; e alguns comandos novos.

2

Funções embutidas

- Python vem com as "baterias inclusas", funções embutidas e prontas para uso:

Nem todas funções é preciso criar. Quando você usa o `print()` ou `input()` está chamando uma função que o Python já vem pronta para você.

3

Funções embutidas

- Python vem com as "baterias inclusas", funções embutidas e prontas para uso:

```
>>> # print() - Exibe textos, separado por sep=' '
>>> # e terminado por end='\n'.
>>> print('a', 'b', 'c')
a b c
>>> print('a', 'b', 'c', sep = '\n')
a
b
c
>>> print('a', 'b', 'c', sep = ', ', end = '\n')
a, b, c.
```

4

Funções embutidas

```
>>> # abs() - Retorna o valor absoluto
>>> # de um número. O valor absoluto
>>> # é o número sem sinal.
>>> abs(5)
5
>>> abs(-5)
5
>>> abs(-4.87)
4.87
```

5

Funções embutidas

```
>>> # max() - Retorna o maior dos argumentos.
>>> max(5, 6, 4)
6
>>> max(8, 9, 2, 7)
9
>>> # min() - Retorna o menor dos argumentos.
>>> min(5, 6, 4)
4
>>> min(8, 9, 2, 7)
2
```

6

Funções embutidas

```
>>> # chr() - Retorna o caractere Unicode de
>>> #      código recebido.
>>> print(chr(48), chr(49), chr(50))
0 1 2
>>> print(chr(65), chr(66), chr(67))
A B C
>>> print(chr(97), chr(98), chr(99))
a b c
```

7

Funções embutidas

```
>>> # ord() - Retorna o código Unicode do
>>> #      caractere recebido.
>>> print(ord('0'), ord('1'), ord('2'))
48 49 50
>>> print(ord('A'), ord('B'), ord('C'))
65 66 67
>>> print(ord('a'), ord('b'), ord('c'))
97 98 99
```

8

Funções embutidas

```
>>> # len() - Retorna o comprimento (o
>>> #      número de itens) de um objeto.
>>> len('IFPI')
4
>>> len('a' + 'bc' + 'def')
6
```

9

Funções embutidas

The Standard Library (Biblioteca padrão)

Uma lista de **funções embutidas** na biblioteca padrão pode ser vista na documentação oficial do python:

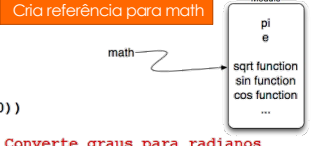
<https://docs.python.org/pt-br/3/library/functions.html>

10

Importação de Módulos

Outras funções fazem parte da Biblioteca Padrão do Python e podem ser usadas fazendo a importação dos módulos.

```
>>> import math
>>> print(math.pi)
3.141592653589793
>>> print(math.e)
2.718281828459045
>>> print(math.sqrt(2.0))
1.4142135623730951
>>> math.radians(90) # Converte graus para radianos
1.5707963267948966
>>> print(math.sin(math.radians(90))) # Seno de 90 graus
1.0
```



11

Importação de Módulos

```
>>> # Importa as funções sin, pi, cos e sqrt
>>> # do módulo math. Renomeia pi para PI.
>>> from math import sin, pi as PI, cos, sqrt
>>> PI # Retorna o valor de PI
3.141592653589793
>>> sin(PI/4) # Retorna o seno de um número
0.7071067811865476
>>> cos(2*PI/3) # Retorna o cosseno de um número
-0.4999999999999998
>>> sqrt(81) # Retorna a raiz quadrada
9.0
```

<https://docs.python.org/pt-br/3/library/math.html>

12

Importação de Módulos

```
>>> # Importa tudo do módulo random
>>> import random
>>>
>>> # Inicializa o gerador aleatório
>>> random.seed()
>>>
>>> # Gera um número real entre 0.0 e 1.0
>>> random.random()
0.4412071598449375
```

<https://docs.python.org/pt-br/3/library/random.html>

13

Importação de Módulos

```
>>> # Retorna um inteiro entre 0 e 9
>>> random.randrange(10)
7
>>>
>>> # Retorna um inteiro entre 1 e 6
>>> random.randrange(1, 7)
4
>>>
>>> # Sorteia um valor de uma lista
>>> random.choice(['vermelho', 'preto', 'verde'])
'preto'
```

<https://docs.python.org/pt-br/3/library/random.html>

14

Importação de Módulos

Dê uma olhada no Índice de Módulos Python. Você vai encontrar, em ordem alfabética, todos os módulos disponíveis.

<https://docs.python.org/pt-br/3/py-modindex.html>

15

Instalando módulos



O **Python Package Index (PyPI)** é um repositório de software para a linguagem de programação Python.

O PyPI ajuda a encontrar e instalar o software desenvolvido e compartilhado pela comunidade Python.

Os autores usam o PyPI para distribuir seu software.

Instalação com pip:

pip install AlguemPacote

16

Criação de Módulos

Módulos são apenas códigos-fonte que podemos importar. Você pode criar seus próprios módulos com funções em Python.

Criação de Módulos

```
meu_modulo.py - D:/Desktop/meu_modulo.py (3.8.1)
File Edit Format Run Options Window Help
def func_a():
    print('func_a')

def func_ab():
    func_a()
    print('func_b')

def func_abc():
    func_ab()
    print('func_c')

func_abc()
```

meu_modulo.py

Um módulo é um simples arquivo de script em Python, como este.

17

18

Criação de Módulos

A execução gera essa saída.

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Desktop/meu_modulo.py =====
func_a
func_b
func_c
>>>
```

19

Criação de Módulos

```
>>> import meu_modulo
func_a
func_b
func_c
```

A importação de um módulo faz com que ele todo seja compilado e executado.

Ou seja, todo código que estiver definido em primeiro nível, isto é, fora da definição de funções, será executado como código de inicialização do módulo.

20

Criação de Módulos

Nem sempre esse é o comportamento desejado. Na maioria das vezes só queremos que nosso código seja executado se o módulo for o nosso programa principal.

```
def func_a():
    print('func_a')

def func_ab():
    func_a()
    print('func_b')

def func_abc():
    func_ab()
    print('func_c')

func_abc()
```

Aqui, nosso módulo é o programa principal.

21

```
def func_a():
    print('func_a')

def func_ab():
    func_a()
    print('func_b')

def func_abc():
    func_ab()
    print('func_c')

def main():
    func_abc()

if __name__ == '__main__':
    main()
```

Modo correto de se fazer o módulo principal.

A variável `__name__` (interna do Python) armazena o nome do módulo atual. Neste caso, o código de inicialização verifica, através dela, se o módulo é o principal, e executa caso seja.

Você encontrará este código muito frequentemente.

22

```
*calculadora.py - D:/Desktop/calculadora.py (3.8.1)
File Edit Format Run Options Window Help

def soma(a, b):
    return a + b

def subtrai(a, b):
    return a - b

# Para controlar a execução
# criamos uma função main()
def main():
    print(soma(2, 5))
    print(subtrai(2, 5))

if __name__ == '__main__':
    main()
```

Módulo calculadora.py

23

Parâmetro opcional

É possível tornar um parâmetro opcional definindo seu valor inicial.

```
def volume_esfera(raio, PI=3.14):
    return (4/3) * PI * (raio ** 3)
```

```
print(volume_esfera(5))
523.3333333333334

print(volume_esfera(5, 3.14159265358979323846))
523.5987755982989
```

É possível chamar a função com ou sem o parâmetro opcional.

24

Parâmetro opcional

Os parâmetros `sep` e `end` são opcionais na função `print()`

```
>>> print('a', 'b', 'c', sep = ', ', end = '\n')
a, b, c.
```

O parâmetro `start` é opcional na função `random.randrange()`

```
>>> import random
>>> random.randrange(100)
47
>>> random.randrange(90, 100)
91
```

25

Pass

- O comando `pass` não faz nada. Pode ser usado quando a sintaxe exige um comando mas a semântica do programa não requer nenhuma ação. Por exemplo:

```
def alguma_funcao():
    pass # Depois veja como faz isso
```

```
x = alguma_funcao()
```

26

None

É uma ausência de valor. Uma função sem o retorno definido irá retornar `None`.

```
>>> def minha_funcao():
    pass
>>> x = minha_funcao()
>>> print(x)
None
>>> print(minha_funcao())
None
>>>
```

27

Write code in Python 3.8

```
1 def horas_e_minutos(minutos):
2     h = minutos // 60
3     m = minutos % 60
4     return h, m
5
6 min = int(input('Quantidade de minutos: '))
7 horas, minutos = horas_e_minutos(min)
8 print(f'{min} minutos é igual a {horas}
9
10
11
```

Print output (drag lower right corner to resize)

Quantidade de minutos: 190

Quantidade de minutos: 190

Frames

- Global frame
 - horas_e_minutos
- function horas_e_minutos(minutos)
 - min: 190

Objects

- tuple
 - 0: 3
 - 1: 10

Return value: (3, 10)

Step 8 of 9

Não esqueça da possibilidade de depurar o código usando o Python Tutor (<http://pythontutor.com/live.html>)

28