Módulo I – Fundamentos Computacionais

Ses 3 - Aula 1 - Funções

Prof. Dr. Luiz Alberto Vieira Dias
Prof. Dr. Lineu Mialaret

Funções

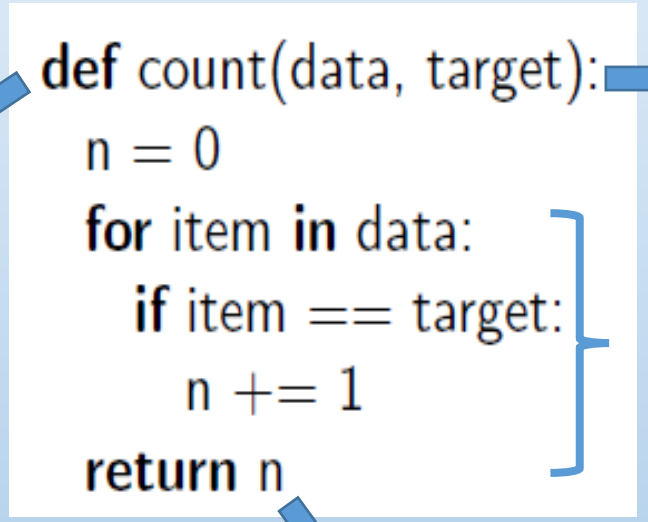
- Bloco de código que implementa alguma funcionalidade
  - Invocada:
    - sem o contexto de uma classe
    - uma instância desta classe
  - Exemplo:

    ```
    sorted (data)
    ```
- Método é uma função de um objeto
  - Exemplo:

    ```
    data.sort()
    ```

Funções (cont.)

- Exemplo: Uma função que conta o número de ocorrências de um dado valor dentro um conjunto de valores.



A palavra chave def define a assinatura da função, especificando o nome da função e a quantidade de parâmetros que ela espera

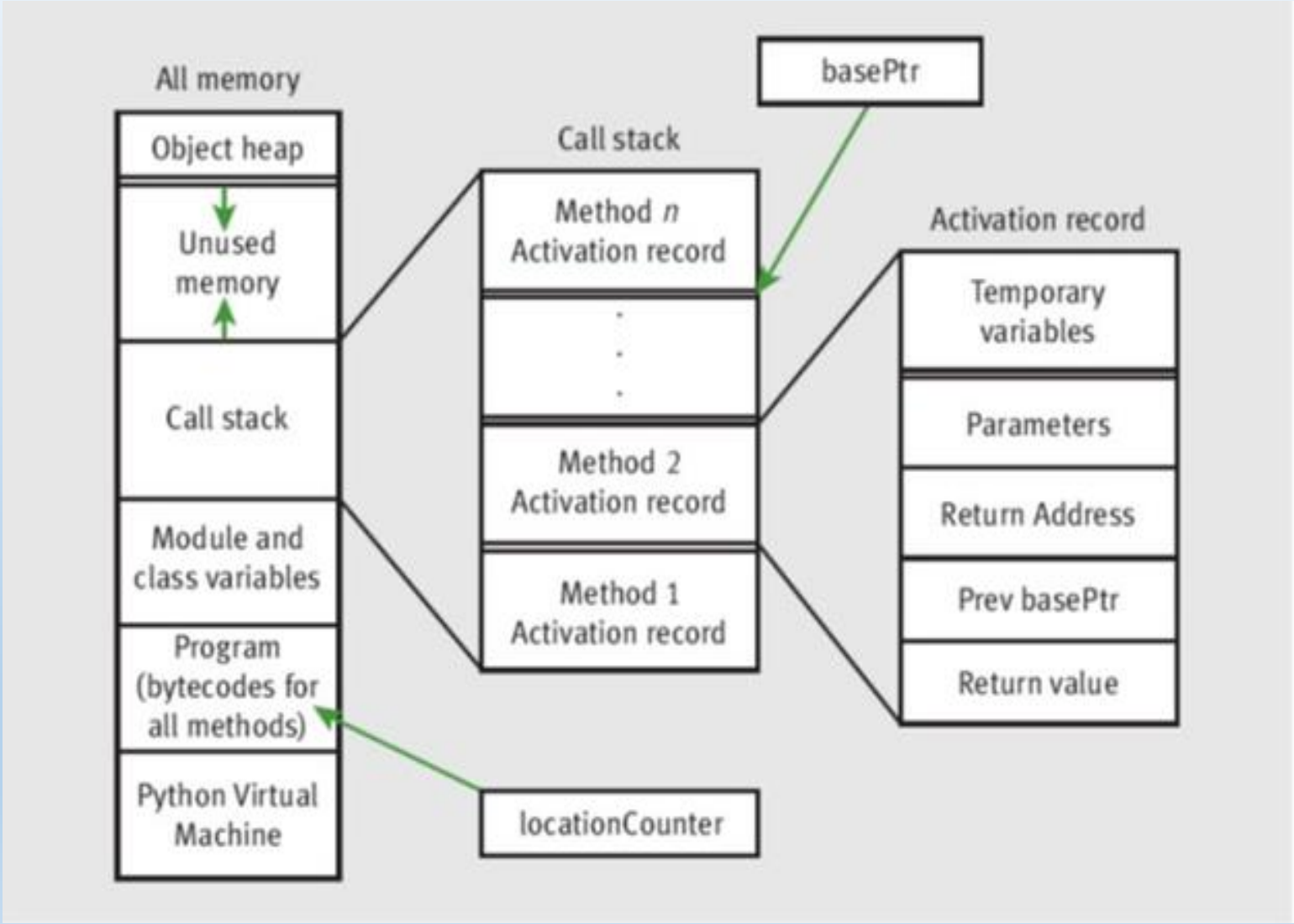
Parâmetros da função

Corpo da função

Retorna da função

Funções (cont.)

A cada chamada de função o Python cria um registro de ativação (*activation record*) que armazena informações relevantes da chamada, incluindo o espaço de nomes (*namespace*), o qual inclui os parâmetros da função e identificadores locais criados no corpo da função



Instrução return

- Uma instrução return é utilizada dentro do corpo de uma função para sinalizar que a função terminou sua execução e que um valor será retornado ao chamador

```
def contains(data, target):
    for item in target:
        if item == target:
            return True
    return False
```

- Se não existe uma valor de retorno explicito, o objeto None é retornado
- Se a instrução return não for executada, o objeto None é retornado

```
def count(data, target):
    n = 0
    for item in data:
        if item == target:
            n += 1
    return n
```

- Pode existir mais de uma instrução return no corpo da função

Instrução return (cont.)

- Exemplos:

```
def count(data, target):
    n = 0
    for item in data:
        if item == target:
            n += 1
```

```
text = "Pyton"
targ = "t"
x = count(text,targ)
print ("Na palavra " +
       text + " há "+ str(x)
       + " letra(s) " + targ)
```

```
Na palavra Pyton há None letra(s) t
```

```
def count(data, target):
    n = 0
    for item in data:
        if item == target:
            n +=1
    return n
```

```
text = "Pyton"
targ = "t"
x = count(text,targ)
print ("Na palavra " +
       text + " há "+ str(x)
       + " letra(s) " + targ)
```

```
Na palavra Pyton há 1 letra(s) t
```

Parametrização

- *Assignment Statement*
  - Quando uma função é invocada, cada identificador de parâmetro formal é atribuído, no escopo local da função, para o respectivo parâmetro atual fornecido pelo chamador da função
  - Identificadores – descrevem os parâmetros esperados (Parâmetros Formais)
  - Valores – enviados quando invoca a função (Parâmetros Atuais)

Parametrização (cont.)

- Função

```
def count(data, target):
    n = 0
    for item in data:
        if item == target:
            n += 1
    return n
```

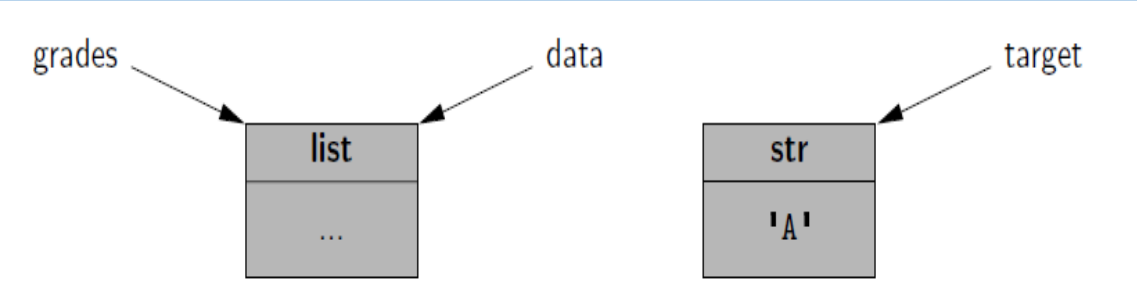
- Chamada da função

```
grades = ['B','C','B','A']

qtdA = count(grades, 'A')
```

- Descrição dos parâmetros

```
data = grades
target = 'A'
```



- Resultado

```
grades = ["B","C","B","A"]
qtdA = count(grades,"A")
print("Você teve " + str(qtdA) + " nota(s) A")
```

```
Você teve 1 nota(s) A
```

Parametrização (cont.)

- Variando a assinatura – Polimórfica
- Funções podem declarar um ou mais parâmetros com valores default

```
def foo(a, b=20, c=30):
    print("Vizualizando valores\na = " + str(a) + "\nb = " + str(b) + "\nc = "+ str(c))
```

```
foo(4,1,8)
```

```
foo(4)
```

```
foo(8,10)
```

```
foo(a=8, c=20)
```

Parametrização (cont.)

- Exemplo:

Função range

```
print(list(range(10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print(list(range(10,20)))
```

```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
print(list(range(10,20,2)))
```

```
[10, 12, 14, 16, 18]
```

range() constructor has two forms of definition:

```
range(stop)
range(start, stop[, step])
```

range() Parameters

range() takes mainly three arguments having the same use in both definitions:

- **start** - integer starting from which the sequence of integers is to be returned
- **stop** - integer before which the sequence of integers is to be returned.
  The range of integers end at **stop - 1**.
- **step (Optional)** - integer value which determines the increment between each integer in the sequence

Return value from range()

range() returns an immutable sequence object of integers depending upon the definitions used:

Parametrização (cont.)

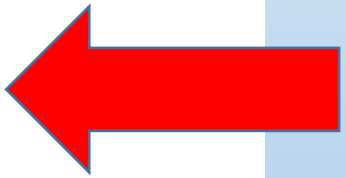
- Atenção nos parâmetros

```
def greet(name,msg):
    print("Hello",name + ', ' + msg)
```

```
greet("Monica")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-64e0768f0ada> in <module>()
----> 1 greet("Monica")

TypeError: greet() missing 1 required positional argument: 'msg'
```



Correção do erro

```
greet("Monica","Good morning!")
```

```
Hello Monica, Good morning!
```

Parametrização (cont.)

- *keyword argument*
  - explicitar o argumento passado para a função

```
def greet(name,msg):
    print("Hello",name + ', ' + msg)
```

```
greet(name = "Bruce",msg = "How do you do?")
```

```
Hello Bruce, How do you do?
```

```
greet(msg = "How do you do?",name = "Bruce")
```

```
Hello Bruce, How do you do?
```

Parametrização (cont.)

- Parâmetros default - podem ou não ser requeridos
- A função pode ter qualquer número de parâmetros default, mas uma vez que o parâmetro seja definido como default, os próximos tem que ser default

```
def greet(name, msg = "Good morning!"):
    print("Hello",name + ', ' + msg)
```

```
greet("Kate")
```

```
Hello Kate, Good morning!
```

```
greet(name = "Bruce",msg = "How do you do?")
```

```
Hello Bruce, How do you do?
```

```
def greet(msg = "Good morning!", name):
    print("Hello",name + ', ' + msg)
```

```
  File "<ipython-input-4-a1a7d6a04523>", line 1
    def greet(msg = "Good morning!", name):
                                    ^
SyntaxError: non-default argument follows default argument
```

Parametrização (cont.)

- Em determinados cenários pode-se não saber o número exato de parâmetros a serem passados para a função
  - O Python permite que sejam realizadas chamadas de funções com número arbitrário de parâmetros
  - Usa-se o * antes do nome do parâmetro (*args)

```
def greet(*names):
    for name in names:
        print("Hello",name)
```

```
greet("Monica")
```

```
Hello Monica
```

```
greet("Monica","Luke")
```

```
Hello Monica
Hello Luke
```

```
greet("Monica","Luke","Steve")
```

```
Hello Monica
Hello Luke
Hello Steve
```

Parametrização (cont.)

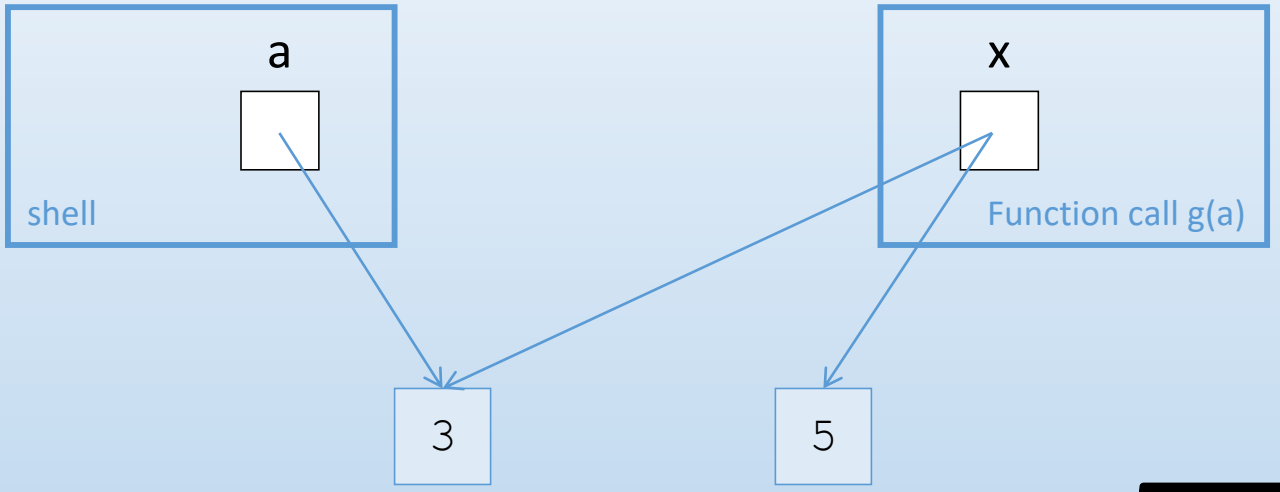
- Pode-se também fazer chamadas de funções com número arbitrário de parâmetros chaves (*keyword arguments*) nomeados
- Usa-se o * antes do nome do parâmetro (*kargs)

```
def my_func(**kwargs):
    for i, j in kwargs.items():
        print(i, j)
```

```
my_func(name='tim', sport='football', age=19)
```

```
name tim
sport football
age 19
```

Passagem de Parâmetros Imutáveis



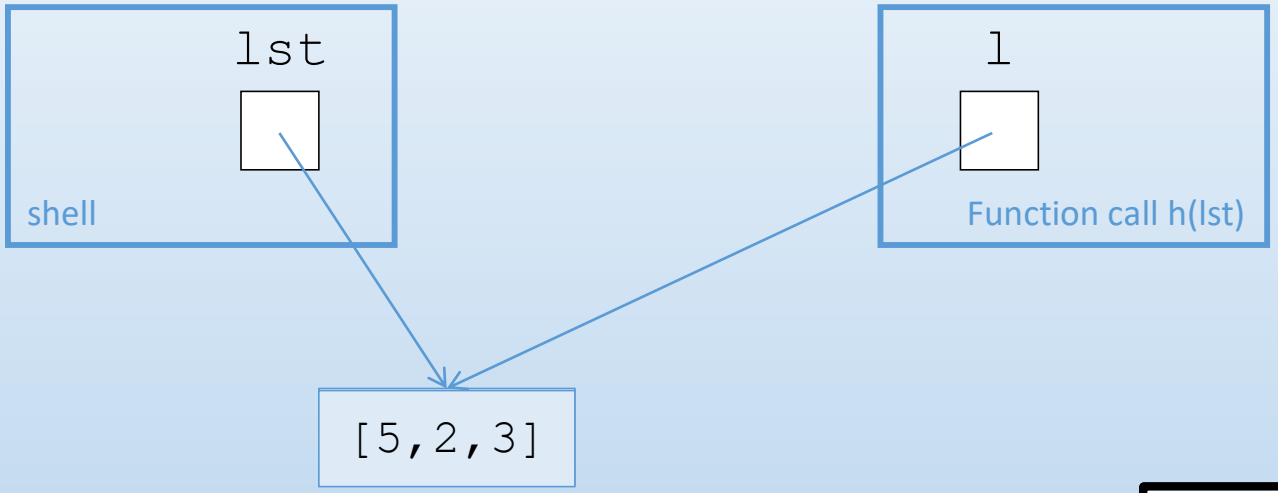

```
def g(x):
    x = 5
```

```
>>> a = 3
>>> g(a)
>>>
```

Variable `x` inside `g()` refers to the object `a` refers to
As if we executed x = a

Function `g()` did not, and cannot, modify the value of `a` in the interactive shell.
This is because `a` refers to an immutable object.

Passagem de Parâmetros Mutáveis




```
def h(l):
    l[0] = 5
```

```
>>> lst = [1,2,3]
>>> h(lst)
>>>
```

Function `h()` did modify the value of `lst` in the interactive shell.
Variable `l` inside `h()` refers to the object `lst` refers to
This is because `lst` and `l` refer to an mutable object.
As if we executed `l = lst`