Pensamento Computacional

2022/23

Aula 5

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Objetivos

- Funções:
  - Definição e chamada.
  - Parâmetros e variáveis locais.
  - Expressões lambda

Funções

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Até agora, estudamos e usamos apenas funções pré-definidas em Python, como:

```
name = input("Name? ")
print("Hello", name, "!")
root2 = math.sqrt(2)
```

Funções

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Mas também podemos definir novas funções:

```
def square(x):
    y = x**2
    return y
```

- Após a definição, podemos chamar a função, tal como qualquer outra função.

```
print(  square(2)  +  square(3)  )
x = 3
print( 2 + square(1-square(x)) )
```

Definição de Função

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- A definição da função permite especificar:
  - o nome da nova função,
  - uma lista de parâmetros, e
  - um bloco de delarações a executar quando a função é chamada.

Syntax	Exemplo

`def functionName(parameters):`
`    statements`
	`def hms2sec(h, m, s):`
	`    sec = (h*60+m)*60+s`
	`    return sec`

Definição de Função

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- A primeira linha da definição da função é designado de *header* (cabeçalho), o resto é designado de ***body*** (corpo).
- O cabeçalho começa com a palavra-chave **`def`** e termina com '**:**' . O corpo tem de estar **indentado**.
- Os nomes das funções, seguem as mesmas regras que os nomes das variáveis.

Syntax	Exemplo

`def functionName(parameters):`
`    statements`
	`def hms2sec(h, m, s):`
	`    sec = (h*60+m)*60+s`
	`    return sec`

Definição Vs Invocação

- Não confundir **Definição** com **Invocação**!

```
def square(x):              #definition
    return x**2


                            #invocations
print(square(3))
area = square(size)
h = math.sqrt(square(x2-x1) + square(y2-y1))
```

Definição Vs Invocação

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Não confundir **Definição** com **Invocação**!

- Na **definição** da função, as declarações **não são executadas**: apenas são **guardadas** para uso posterior.
- As declarações são **executadas** apenas se e **quando** a função é **invocada**.
- A função tem de ser definida antes de ser chamada (invocada).
- Define-se uma vez, chama-se quantas vezes necessário.

Exemplo

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

```
def hello():
    print("Hello!")

def helloTwice():
    hello()
    hello()

#calling the function
helloTwice()
```

- Este exemplo, contém 2 definições de função: `hello` and `helloTwice`.
- Posteriormente, a `helloTwice` é chamada (invocada).
- Quando a `helloTwice` é executada, chama 2x a `hello`.

Fluxo da Execução

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- A execução começa sempre na primeira declaração do programa. As declarações são executadas uma de cada vez, por ordem, de cima para baixo.
- As definições de funções não alteram o fluxo de execução de um programa. Estas apenas guardam as declarações no corpo da função para serem usadas posteriormente. O corpo da função não é executado no momento.
- A chamada de uma função é como um desvio no fluxo da execução:
  - Em vez de ir para a próxima declaração, o fluxo salta para o corpo da função, executa todas as declarações no corpo, e, posteriormente, retorna ao local do programa onde desviou.

Parâmetros e Argumentos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Em algumas das funções pré-definidas analisadas em aulas anteriores, vimos argumentos obrigatórios.
- Por exemplo, quando chamamos `math.sin` passamos um número como argumento.
- Algumas funções, precisam de mais de um argumento
  - `math.pow` necessita de 2 argumentos: a base e o expoente.

Parâmetros e Argumentos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Quando a função é chamada, os argumentos são atribuidos às variáveis, as quais são designadas parâmetros.

```
def print2times(msg):
    print(msg)
    print(msg)

print2times("bye")
```

msg="bye"

Valores de Retorno

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Algumas funções, tais como as de `math`, devolvem resultados.
- Outras, como `print`, executam uma ação mas não devolvem um valor.

- Estas funções são denominadas *void functions*.

  (Na realidade, estas retornam o valor especial `None`.)

Valores de Retorno

- A declaração

```
return expression
```

termina a execução da função e devolve o resultado da expressão `expression`.

- Uma declaração **`return`** sem argumentos,

```
return
```

é equivalente a **`return`** `None`.

Variáveis locais Vs globais

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Variáveis definidas dentro do corpo de uma função têm um âmbito local. As definidas fora do corpo da função são variáveis globais.

```
def add(a, b):
    total = a + b    # Here total is local variable
    print("Inside: ", total)
    return total

total = 0                # This is a global variable
print( add(10, 20))      # Call add function
print("Outside: ", total)
```

Parâmetros são variáveis locais

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Os parâmetros são também variáveis locais. Podemos alterá-los, mas o efeito é local.

```
def double(x):
    x *= 2          # you may modify parameters
    return x

x = 3
y = double(x)   # <=> double(3)
print(x, y)     # What's the value of x and y?
```

- Quando a função é chamada, o parâmetro recebe (apenas) o valor do argumento.
- Esta forma de passagem de argumento é denominada de *passagem por valor*.

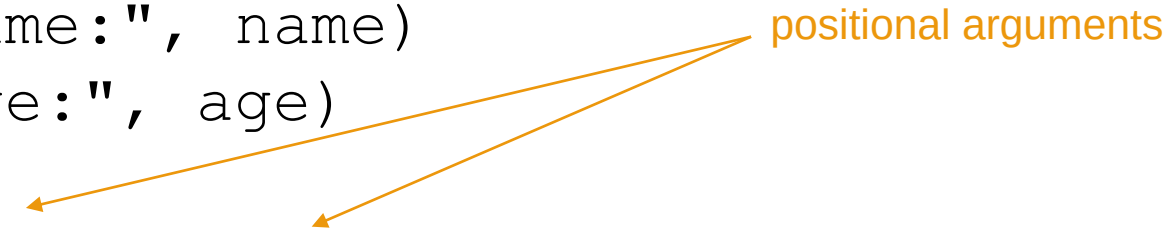
Argumentos *keywork*

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Numa chamada de função, os argumentos posicionais são atribuidos aos parâmetros de acordo com a sua posição.

```
def printinfo( name, age ):
    print("Name:", name)
    print("Age:", age)

printinfo( "miki", 50 )
```



Argumentos *keywork*

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Quando se usam **argumentos keyword**, identificam-se os argumentos por nome de parâmetro.

```
printinfo( "miki", age=50 )
printinfo( age=50, name="miki" )
```

- Com argumentos keyword, não tem de se lembrar da ordem, apenas dos seus nomes.

Valores Padrão de Argumento

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- A definição da função pode contem **valores padrão de argumento** para alguns dos seus parâmetros.

```
def printinfo( name, age=35 ):
    print("Name: ", name)
    print("Age ", age)
```

- Quando a função é chamada, se o valor do argumento não é fornecido, este toma o valor padrão.

```
printinfo( "miki", 50 )
printinfo( "miki" )          # here, age is 35!
printinfo( name="miki" )     # same here
```

Valores Padrão de Argumento

- Isto é particularmente útil para argumentos opcionais.

```
print(1, 2, 3)
print(1, 2, 3, sep='->')
print(1, 2, 3, sep='->', end='\n-FIM-\n')
```

Expressões Lambda

(Tópico avançado, não será avaliado)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Uma *expressão lambda* é uma expressão cujo resultado é uma função.

- É possível guardá-la numa variável e usá-la mais tarde, por exemplo.

```
add =  lambda a, b: a + b ← #lambda expression
# Now you can call add as a function
print("Total: ", add(10, 20))  #Total: 30
```

Expressões Lambda

(Tópico avançado, não será avaliado)

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

```
add =  lambda a, b: a + b ← #lambda expression
# Now you can call add as a function
print("Total: ", add(10, 20))  #Total: 30
```

- São também conhecidas por *funções anónimas*.
- A lambda pode ter 0 ou mais parâmetros, mas o seu resultado é sempre um resultado único.
- Não podem conter declarações, apenas uma única expressão.
- São muito úteis para passar argumentos para outras funções.

Exercício

O programa `bmi.py` serve para calcular o índice de massa corporal, mas está incompleto.

O programa inclui três funções.

Analise o seu funcionamento.

- Complete a definição da função `bodyMassIndex` para calcular o índice pela razão $bmi = \frac{weight}{height^2}$.
- Complete a função bmiCategory para devolver uma string com a categoria correspondente ao índice de massa corporal dado.
- Complete os argumentos na invocação da função, dentro da função principal.

Teste o programa.

Pensamento Computacional

2022/23

TPC: exercícios propostos aula 5