

Funções e Classes

Ferramentas e Aplicações em Biotecnologia

Vasco Ferrinho Lopes
vasco.lopes@ubi.pt

UBI
Ano letivo 2023-2024



Funções

Uma função é um **bloco de código** que executa uma **tarefa específica**. Pode ser reutilizado quantas vezes necessário de forma a reduzir a duplicação de código. Uma função pode receber dados (parametros) e “retorna” resultados.

```
x = 10
y = 20

if x<y:
    print("Y é maior")
elif x>y:
    print("X é maior")
else:
    print("São iguais")

t = 20
p = 50

if t<p:
    print("P é maior")
elif t>p:
    print("T é maior")
else:
    print("São iguais")
```

Y é maior
P é maior

```
def check_bigger_between_two(first_value, second_value):
    if first_value<second_value:
        print("second_value é maior")
    elif first_value>second_value:
        print("first_value é maior")
    else:
        print("São iguais")


x, y = 10, 20
check_bigger_between_two(x, y)
t, p = 70, 50
check_bigger_between_two(t, p)
```

second_value é maior
first_value é maior



Funções

Uma **função** é definida usando a palavra-chave **def**



```
def my_function():  
    print("Hello from a function")
```

e é utilizada “chamando-a”:




```
def my_function():  
    print("Hello from a function")  
  
my_function()  
  
Hello from a function
```




Funções

Pode passar-se informações para funções como **argumentos**.

Os **argumentos** são especificados após o nome da função, entre parênteses. Pode adicionar-se quantos argumentos se quiser, basta separá-los com vírgula.



```
def combine_names(fname, lname):  
    print(f"O meu nome é: {fname} {lname}")  
  
combine_names("Vasco", "Lopes")  
  
O meu nome é: Vasco Lopes
```



Funções

Os **argumentos** são especificados após o nome da função, entre parênteses. Pode adicionar-se quantos argumentos se quiser, basta separá-los com vírgula.

É possível especificar a **qual argumento** se está a atribuir **valor**.


```
def combine_names(fname, lname):  
    print(f"0 meu nome é: {fname} {lname}")  
  
combine_names(fname="Vasco", lname="Lopes")  
  
0 meu nome é: Vasco Lopes
```




Funções

Os **argumentos** são especificados após o nome da função, entre parênteses. Pode adicionar-se quantos argumentos se quiser, basta separá-los com vírgula.

É também possível especificar o **tipo de dados** de cada parâmetro.



```
def combine_names(fname = str, lname = str):  
    print(f"O meu nome é: {fname} {lname}")  
  
combine_names(fname="Vasco", lname="Lopes")  
  
O meu nome é: Vasco Lopes
```



Funções

Os **argumentos** são especificados após o nome da função, entre parênteses. Pode adicionar-se quantos argumentos se quiser, basta separá-los com vírgula.

Pode também definir-se o **tipo de dados** de retorno da função.

```
def combine_names(fname = str, lname = str) -> str:  
    print(f"0 meu nome é: {fname} {lname}")
```

```
combine_names(fname="Vasco", lname="Lopes")
```

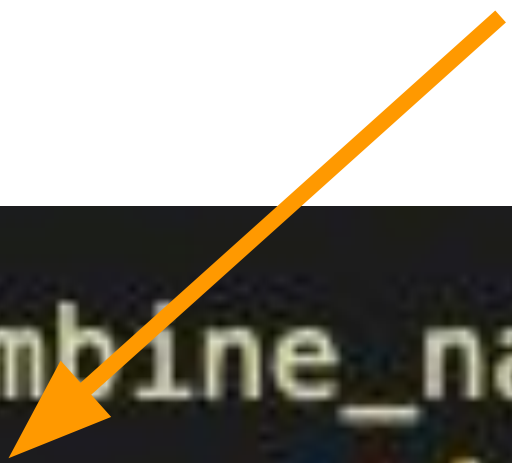
```
0 meu nome é: Vasco Lopes
```




Funções

Os **argumentos** são especificados após o nome da função, entre parênteses. Pode adicionar-se quantos argumentos se quiser, basta separá-los com vírgula.

Para devolver valores, usamos a palavra-chave **return**



```
def combine_names(fname = str, lname = str) -> str:  
    return f"{fname} {lname}"  
  
nome_completo = combine_names("Vasco", "Lopes")  
print(nome_completo)
```




Funções - Exercícios

Crie uma função que pede ao utilizador uma lista de 10 inteiros.

Crie uma função que devolve o máximo e mínimo de uma lista.

Escreva uma função que verifique se um número é par ou ímpar. Ela deve retornar True se o número for par e False se for ímpar. Faça chamadas para testar sua função.

Crie uma função que receba o raio de um círculo como parâmetro e retorne a área do círculo. Teste chamando a função para diferentes raios.



Funções - Exercícios

Crie uma função que pede ao utilizador uma lista de 10 inteiros.

Crie uma função que devolve o máximo e mínimo de uma lista.

Escreva uma função que verifique se um número é par ou ímpar. Ela deve retornar True se o número for par e False se for ímpar. Faça chamadas para testar sua função.

Crie uma função que receba o raio de um círculo como parâmetro e retorne a área do círculo. Teste chamando a função para diferentes raios.



Classes

Python é uma **linguagem orientada a objetos**, isto é, quase tudo em python é um **object**, com propriedades e métodos internos. Uma classe é como um **construtor**, podendo se pensar nesta como uma “**planta**” para criar objetos.

```
class Docente():  
    def __init__(self, nome, idade):  
        self.nome = nome  
        self.idade = idade
```

```
vasco = Docente("Vasco", 99)  
print(vasco.nome)  
print(vasco.idade)
```

```
Vasco  
99
```

Para se criar uma classe, utiliza-se a palavra chave **class**

Pode ter um **método/função construtor** para inicializar variáveis e instruções que sejam necessárias: **`__init__()`**

Pode aceder-se às variáveis internas



Classes

```
class Docente():
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def __str__(self):
        return f"0 {self.nome} tem {self.idade} anos"

vasco = Docente("Vasco", 99)
print(vasco.nome)
print(vasco.idade)
print(vasco)
```

```
Vasco
99
0 Vasco tem 99 anos
```

É possível reescrever métodos, como o `__str__` para se adequar às necessidades da classe. Neste caso, permite fazer *print* diretamente de variáveis do tipo “Docente”.

O parâmetro ***self*** é uma referência à instância atual da classe e é usado para aceder a variáveis que pertencem à instância da classe.



Classes

```
class Docente():
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def __str__(self):
        return f"0 {self.nome} tem {self.idade} anos"

    def print_name_lower(self):
        print(self.nome.lower())

vasco = Docente("Vasco", 99)
print(vasco.nome)
print(vasco.idade)
print(vasco)
vasco.print_name_lower()
```

```
Vasco
99
0 Vasco tem 99 anos
vasco
```

Podem ser feitos métodos para o que seja necessário efetuar. De notar que estes métodos são transversais às instâncias desse tipo de dados (classe), então, devem ser métodos interessantes para esse objeto. Métodos generalistas e fora do âmbito do objeto, devem ser colocados fora da classe.



Classes - Exercícios

Crie uma classe que represente animais (animal).

Essa classe deve conter nome, idade e espécie.

Adicione os seguintes métodos à classe:

`__str__():`

`alterar_nome()`

`alterar_idade()`

Crie uma classe que herde as propriedades da classe animal e seja específico para cães.