

Fundamentos de Programação

>Aulas Práticas

Fernando Manuel Soares

fernando.soares@ulusofona.pt

Aulas

desta componente prática

3 aulas práticas:

1. Hoje (duh!)
2. 21/11
3. 16/01

Como são as aulas?

- Revisão,
- Esclarecimento de dúvidas,
- Exercícios,
- “codealongs”
- Mais tarde acompanhamento ao projeto final (em princípio: a partir da Aula 2).

Estrutura da aula

1. Avaliação
2. Aferição de conhecimento e revisão
 - a. *While*
 - b. *For*
 - c. *Arrays e Arrays multidimensionais*
 - d. *Funções*
3. *Cheatsheets*
4. *Exercícios*

Avaliação

desta componente prática

Avaliação contínua:

A avaliação contínua (**20% da nota final**) é feita através de vários exercícios “hosteados” na plataforma online - ***repl.it***

Oh god, quantos exercícios D: ?

Neste momento : 16 estão *online*. Por cada aula teórica vou adicionar **4, até à próxima aula prática. (Totalizando 28)**

O que é o ***repl.it***:

Explico mais a frente durante a aula, como aceder / registar etc.

Repl.it – Avaliação Contínua

Avaliação contínua (**20% da nota final**) é feita pela plataforma online de sala de aula virtual - ***repl.it***

Passos para registo na sala de aula virtual no repl.it:

1. Aceder: <https://repl.it/classroom/invite/fhzCfY5>
2. Fazer o registo na plataforma.
3. Criar Nome de utilizador. **Atenção: usem os vossos nomes, se eu não vos conseguir identificar correm o risco de não serem avaliados!**
4. Aceder à sala de aula: “FundamentosProgramação_2019_2020_ULHT”

Ciclos condicionais: While

```
while <expressão booleana> :  
    <bloco de instruções>
```

Ciclos **while** continuam a executar se a expressão booleana for avaliada como **true**

Ciclos condicionais: While

Ciclos *while* continuam a executar se a expressão booleana for avaliada como *true*.

Tabela com valores que avaliam como true or false:

<https://bit.ly/2PbH8PS>

Ciclos condicionais: While

Exemplo :

```
user_input = None

while user_input != "Yes" :
    user_input = input("Exit the loop? ")

print("Goodbye")
```


Ciclos condicionais: While

Mini-exercício 1 :

Com base no exemplo anterior, façam com que a condição de terminação seja qualquer configuração de: “Yes” (minúsculo, maiúsculo etc.)

Ex: Yes, yEs, ...

Ciclos condicionais: While

Mini-exercício 2 :

Cria um programa que imprime este padrão usando o **while**:

#

##

###

Ciclos condicionais: While

Mini-exercício 3 :

Cria um programa que imprime este padrão usando o **while**:

```
#  
##  
###
```

Ciclos condicionais: for

```
for <item> in <objeto_iterável> :  
    <bloco de instruções>
```

Ciclos **for** executam o bloco de instruções por cada item de um objeto iterável (list, string, range, ...)

Ciclos condicionais: for

Mini-exercício 4 :

Façam um programa que imprima todos os números de 1 a 50.

Ciclos condicionais: for

Mini-exercício 5 :

Façam um programa que imprima todos os números pares de 1 a 50.

Ciclos condicionais: for

Mini-exercício 6 :

Façam um programa que imprima este padrão usando o **for**:

#

##

###

Arrays – Listas em python

Estrutura de dados que consiste numa coleção de elementos

```
l = [89, 40.2, True]
```

```
l[2] = True
```

```
l[-1] = ?
```


Arrays

Mini-exercício 7:

Ordenem a seguinte lista:

`l = [7, 4, 1, 7, 2, 3, 5, 1]`

De modo a que fique assim:

`l = [1, 1, 2, 3, 4, 5, 7, 7]`

Arrays

Mini-exercício 8:

Agora eliminem os elementos repetidos:

```
l = [7, 4, 1, 7, 2, 3, 5, 1]
```

De modo a que fique assim:

```
l = [7, 4, 1, 2, 3, 5]
```

Arrays

Mini-exercício 9:

Agora eliminem os números pares, ordenem e tirem os elementos repetidos:

`l = [7, 4, 1, 7, 2, 3, 5, 1]`

De modo a que fique assim:

`l = [1, 3, 5, 7]`

Arrays Multidimensionais

Estrutura de dados que consiste numa coleção com uma ou mais coleções de elementos

```
l = [[2,42,1], 40.2, True]
```

```
l [0] [2] = 1
```

```
l = [[2,42,1], 40.2, (30,2.0, False , [2,4,"My brain hurts"],  
"Hello")]
```

```
l[2][3][2][5] = ?
```

Arrays Multidimensionais

Mini-exercício 10:

Considera a seguinte lista:

```
num = [ [10], [1,2,3], [5,6], [0,2,3] ]
```

Crie um programa que retorne outra lista, **multi**, que contém a multiplicação dos números contidos nas diferentes listas de **num**?

O programa deve retornar:

```
[10, 6, 30, 0]
```

Arrays Multidimensionais

Mini-exercício 11:

Considera a seguinte lista:

```
resultado = [[ Figo, João, Marta], [Figo, João, Marta], [Marta, João, Figo]]
```

na qual a ordem dos nomes em cada sub-lista indica o lugar das três pessoas referidas numa competição. Cada sub-lista indica uma etapa. Ou seja, na primeira etapa o Figo ficou em primeiro lugar, o João em segundo e a Marta em terceiro, e assim sucessivamente.

Cria um programa que indique quem ficou em primeiro lugar mais vezes.

```
> Figo Wins!
```

Funções

```
def <name> (<parâmetros>) :  
    <bloco de código>
```

Funções agrupam blocos de código, tornando esses blocos **reutilizáveis**.

“**Arrumam**” o código num programa, tornando-o **mais organizado e fácil de compreender**, principalmente em programas maiores.

Funções

Mini-exercício 12:

Escreve uma função **dez** que retorna ***True*** se o seu argumento for 10 e ***False*** caso contrário. **Não podes utilizar a instrução *if*.**

Funções

Mini-exercício 13:

Define uma função que recebe uma lista como parâmetro e retorna o último elemento dessa lista.

Se a lista estiver vazia, retorna a keyword ***None***

```
print(exemplo([1, 3, 5]))
```

```
> 5
```

```
print(exemplo())
```

```
> None
```

Funções

Mini-exercício 14:

Define uma função que recebe um parâmetro: um número (**int e float**) ou uma **string**:

A função retorna diferentes coisas seguindo estas regras:

1. Se for um número retorna esse número em formato **string**.
2. Se for uma string retorna a contagem dos caracteres em formato **int**.
3. Se o argumento não for string, int, ou float. Retorna a mensagem: "Sorry, that's not a float or a string"

Dica: usar a função base do python: "isinstance()"

Exercício – Pedra, Papel, Tesoura

Crie um programa no qual um jogador joga contra o computador o jogo clássico: pedra, papel ou tesoura.

Regras:

1. O programa pede o input ao jogador:
 - 1.1. só aceita **“rock”**, **“paper”** ou **“scissors”** ou **“exit”**
 - 1.2. Se o input for **“exit”** o programa deve sair e apresentar a mensagem: **“Thanks for playing”**.
 - 1.3. Em qualquer outro input surge a mensagem **“Command not valid”** e o programa deve reiniciar.
2. A resposta do computador é aleatória (ver: **random.randint**).
3. O jogo deve indicar a jogada do computador e quem é o vencedor.
4. Depois de ser declarado o vencedor, ou se o comando não for válido deve ser reiniciado.

Exercício – Resolução

Para quem quiser ir “mais além” – Exercícios

Plataforma de ensino de python gamificado com mais de 200 exercícios

<https://py.checkio.org/>

Cheatsheets

<https://bit.ly/2N1PQ0m>