Aula 4
12/11/2025

UNIVERSIDADE LUSÓFONA

Fundamentos de Programação (T)

Licenciatura em Videojogos

Ano 1 / Semestre 1

Filipe Pinto
filipe.pinto@ulusofona.pt

UNIVERSIDADE LUSÓFONA

Sumário

- Dicionários
- Sets

UNIVERSIDADE LUSÓFONA

Dicionários

UNIVERSIDADE LUSÓFONA

Dicionários

- Um dicionário é uma estrutura de dados que associa um valor a outro
- Enquanto um *array* indexa uma indíce inteiro a um valor, um dicionário não está restringido a valores inteiros

```
release_year = {
    "God of War" : 2018,
    "Subnautica" : 2018,
    "Asteroids" : 1979,
    "Super Mario Bros" : 1985
}
print(release_year[“Subnautica”])
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> &
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
2018
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

Dicionários

- Normalmente são usadas strings para indexar, mas qualquer valor imutável serve (números, strings, booleanos, tuplos)
- As chaves são únicas (não podem existir 2 valores com a mesma chave)
- As chaves podem ser de tipos diferentes, mas é considerada má pratica de programação. **Evitem!**
- Vantagens:
  - Tamanho dinâmico
  - Permitem associar facilmente pares de chave, valor
- Desvantagens:
  - Requer mais memória
  - Iterar sobre dicionários é mais lento comparado com as listas

UNIVERSIDADE LUSÓFONA

Dicionários

- Um dicionário pode ser inicializado vazio:

- Se tentarmos aceder a um indíce que não existe:

```
release_year = { }
print(release_year["God of War"])
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula4_fp.py", line 13, in <module>
    print(release_year["God of War"])
          ~~~~~~~~~~~~^^^^^^^^^^^^^^
KeyError: 'God of War'
```

Dicionários

- Se tivermos dúvidas se uma elemento existe podemos fazer uma simples verificação

```
release_year = { }

if "God of War" in release_year:
    print(release_year["God of War"])
else:
    print("Don't know the release year!")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Don't know the release year!
```

Dicionários

- Podemos acrescentar ou modificar elementos:

```
release_year = {}

release_year["God of War"] = 2018


if "God of War" in release_year:
    print(release_year["God of War"])
else:
    print("Don't know the release year!")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
2018
```

Dicionários

- Podemos apagar elementos:

```
release_year = {}

release_year["God of War"] = 2018
del release_year[“God of War”]


if "God of War" in release_year:
   print(release_year["God of War"])
else:
   print("Don't know the release year!")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Don't know the release year!
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

UNIVERSIDADE LUSÓFONA

Dicionários

- Podemos iterar as chaves de um dicionário:

```
release_year = { }

release_year["God of War"] = 2018

release_year["Subnautica"] = 2018

release_year["Asteroids"] = 1979

release_year["Super Mario Bros"] = 1985


for name in release_year:
    print(name)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
God of War
Subnautica
Asteroids
Super Mario Bros
```

UNIVERSIDADE LUSÓFONA

Dicionários

- Podemos iterar as chaves de um dicionário:

```
release_year = { }
release_year["God of War"] = 2018
release_year["Subnautica"] = 2018
release_year["Asteroids"] = 1979
release_year["Super Mario Bros"] = 1985

for name, year in release_year.items():
   print(name + " = " str(year))
```

```
God of War = 2018
Subnautica = 2018
Asteroids = 1979
Super Mario Bros = 1985
```

UNIVERSIDADE LUSÓFONA

Dicionários

- Podemos iterar só os valores de um dicionário:

```
release_year = { }

release_year["God of War"] = 2018

release_year["Subnautica"] = 2018

release_year["Asteroids"] = 1979

release_year["Super Mario Bros"] = 1985


for year in release_year.values():
    print("year: " + str(year))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
year: 2018
year: 2018
year: 1979
year: 1985
```

UNIVERSIDADE LUSÓFONA

Tuplos como chaves

- Por serem imutáveis, podem ser usados como chaves para dicionários:

```
test_dic = {}
test_dic[(1,2)] = "Posição (1,2)"
test_dic[(3,4)] = "Posição (3,4)"
test_dic[(5,6)] = "Posição (5,6)"

print(test_dic[1,2])
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Posição (1,2)
```

UNIVERSIDADE LUSÓFONA

Tuplos como chaves

- Por serem imutáveis, podem ser usados como chaves para dicionários:

```
test_dic = {
    (1,2) = "Posição (1,2)"
    (3,4) = "Posição (3,4)"
    (5,6) = "Posição (5,6)"
}

print(test_dic[1,2])
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Posição (1,2)
```

Dicionários

- Posso guardar como valor num dicionário qualquer tipo de dados:
  - Inclusivamente, outros dicionários

```
games = {
    "God Of War" : {"release" : 2018, "genre": "Action"},
    "Super Mario Bros" : {"release" : 1985, "genre": "Platformer" },
    "Asteroids" : {"release" : 1979, "genre": "Arcade"}
}

for name, details in games.items():
    print(name)
    for key, value in details.items():
        print(key, "=", value)
```

```
God Of War
release = 2018
genre = Action
Super Mario Bros
release = 1985
genre = Platformer
Asteroids
release = 1979
genre = Arcade
```

UNIVERSIDADE LUSÓFONA

Exercícios

UNIVERSIDADE LUSÓFONA

Exercícios

1. Cria um dicionário chamado weapon com as seguintes informações:
    - name ->Longsword
    - Damage -> 25
    - Weight -> 8.5
    1. Imprime o nome da arma e o seu dano.
    2. Adiciona uma nova chave "durability" com valor 100.
    3. Imprime novamente o dicionário completo.

2. Cria um dicionário chamado droids com a seguinte informação:
    - name ->K-2SO
    - Type -> KX
    - Stats -> hp = 200, strength = 300, mobility = 50
    1. Imprime o valor de hp e mobility.
    2. Aumenta o hp em 25 e imprime novamente.
    3. Mostra todos os detalhes da personagem incluindo *stats*.

3. Cria um dicionário shop com os seguintes items:
    - potion -> price = 10, stock = 5
    - elixir -> price = 25, stock = 2
    - sword -> price = 100, stock = 1
    1. Mostra todos os itens e respetivos preços.
    2. Pede ao jogador o nome de um item, e mostra o preço correspondente.
    3. se o jogador comprar um item, diminui o valor de "stock".

Conjuntos (Sets)

Conjuntos (Sets)

- Um set é um agregado de elementos únicos, não ordenados
  - Exactamente como um conjunto em matemática.

```
arenas_played = { “Blood Bowl”, ”Graveyard” }
print(arenas_played)
```

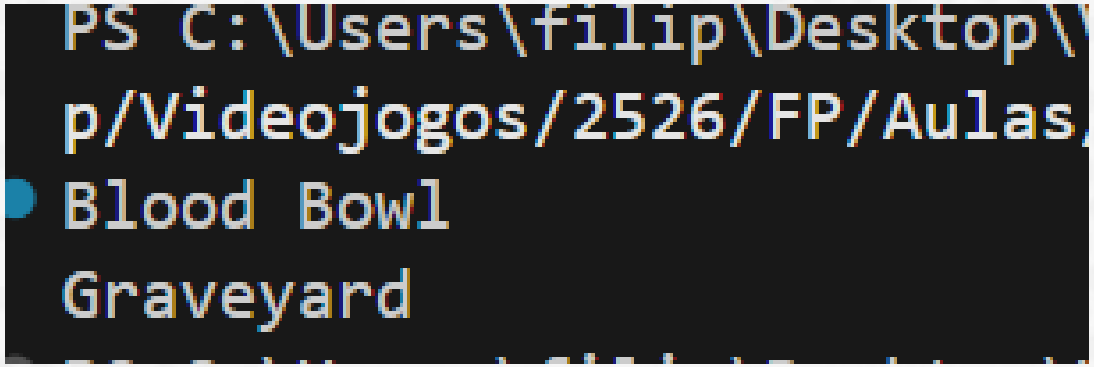
```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
{'Blood Bowl', 'Graveyard'}
```

Conjuntos (Sets)

- Pode ser iterado

```
arenas_played = { "Blood Bowl", "Graveyard" }

for arena in arenas_played:
    print(arena)
```



Conjuntos (Sets)

- A ordem é irrelevante

```
a = { 1, 2, 3 }
b = { 3, 2, 1 }
if (a == b):
   print("Sets are equal")
else:
   print("Sets are different")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Sets are equal
```

Conjuntos (Sets)

- Cada elemento só pode estar uma vez no conjunto:

```
a = { 1, 2, 3, 2 }
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
{1, 2, 3}
```

Conjuntos (Sets)

- Mesmo na comparação:

```
a = { 1, 2, 3, 1 }
b = { 3, 2, 1 }
if (a == b):
   print("Sets are equal")
else:
   print("Sets are different")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Sets are equal
```

Conjuntos (Sets)

- Podemos acrescentar elementos:

```
a = { 1, 2, 3 }
print(a)
a.add(4)
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
{1, 2, 3}
{1, 2, 3, 4}
```

Conjuntos (Sets)

- Se o elemento já estiver no conjunto, nada é alterado:

```
a = { 1, 2, 3 }
print(a)
a.add(3)
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
{1, 2, 3}
{1, 2, 3}
```

Conjuntos (Sets)

- Podemos remover elementos:

```
a = { 1, 2, 3 }
print(a)
a.remove(2)
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
{1, 2, 3}
{1, 3}
```

UNIVERSIDADE LUSÓFONA

Conjuntos (Sets)

- Mas se tentarmos remover um element que não existe, dá erro:

```
a = { 1, 2, 3 }
a.remove(4)
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula4_fp.py", line 80, in <module>
    a.remove(4)
KeyError: 4
```

UNIVERSIDADE LUSÓFONA

Conjuntos (Sets)

- Se não tivermos a certeza se um elemento existe, devemos verificar:

```
a = { 1, 2, 3 }
if (4 in a):
   a.remove(4)
else:
   print("key doesn't exist")
print(a)
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
p/Videojogos/2526/FP/Aulas/Aula4_fp.py
key doesn't exist
{1, 2, 3}
```

UNIVERSIDADE LUSÓFONA

Exercícios

UNIVERSIDADE LUSÓFONA

Exercícios

1. Cria um programa que mantém o registo de caças e pilotos da Rebelião.
   - Adiciona um novo piloto "Luke Skywalker" com o caça "X-Wing".
   - "Luke Skywalker" também pilotou um "Snowspeeder".
   - Mostra todos os caças pilotados por "Poe Dameron".
   - Mostra os caças em comum entre "Luke Skywalker" e "Wedge Antilles".
   - Mostra os caças que "Red Leader" pilotou mas "Poe Dameron" nunca usou.
   - Cria um conjunto `all_ships` com todos os caças únicos usados por todos os pilotos.
   - Mostra a lista completa dos caças usados pela Rebelião, sem repetições.

```
fleet = {
    "Red Leader": {"X-Wing", "A-Wing"},
    "Wedge Antilles": {"X-Wing"},
    "Poe Dameron": {"X-Wing", "T-70", "Black One"}
}
```

2. Cria um programa que gere recompensas imperiais sobre alvos procurados.
   - Adiciona um novo caçador "IG-88" com um set vazio de alvos.
   - Atribui-lhe o alvo "Han Solo".
   - Mostra todos os alvos de "Boba Fett".
   - Mostra os alvos que "Boba Fett" e "Bossk" têm em comum.
   - Mostra os alvos que "Cad Bane" persegue mas "Bossk" não.
   - Calcula o valor total da recompensa de Boba Fett, somando os valores de cada alvo usando o dicionário reward_values.
   - Cria um set com todos os alvos únicos procurados pelo Império.

```
bounties = {
    "Boba Fett": {"Han Solo", "Chewbacca"},
    "Cad Bane": {"Ahsoka Tano", "Fennec Shand"},
    "Bossk": {"Han Solo", "Leia Organa"}
}

reward_values = {
    "Han Solo": 50000,
    "Chewbacca": 40000,
    "Leia Organa": 60000,
    "Ahsoka Tano": 80000,
    "Fennec Shand": 45000
}
```