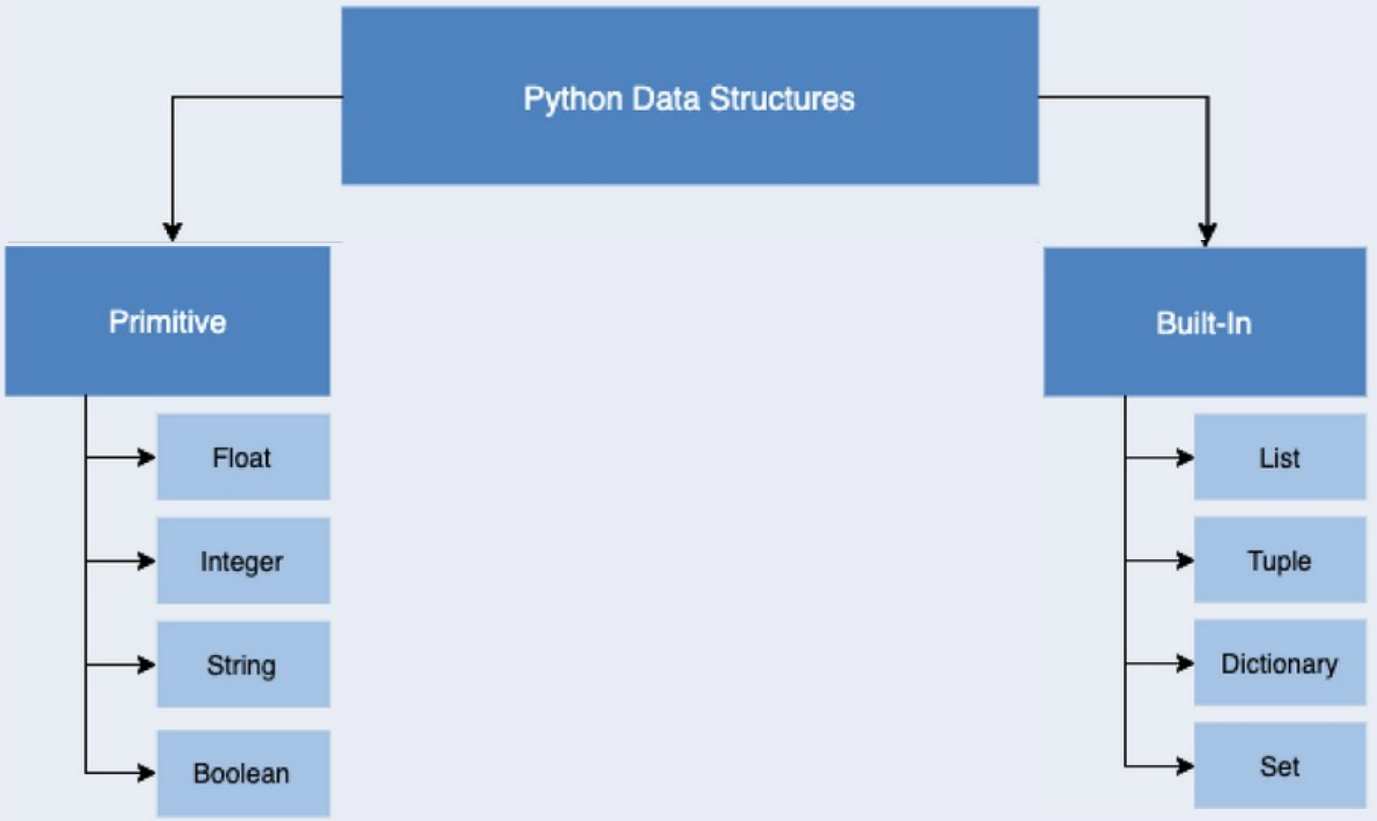
ISPGAYA
instituto superior politécnico

Python – Estruturas de Dados



Python – Estruturas de Dados

Existem quatro tipos de estruturas de dados (coleções) na linguagem de programação Python:

- **List** - é uma coleção que é ordenada e mutável. Permite membros duplicados. As listas são definidas usando parêntesis retos []. Permitem armazenar vários valores numa só variável.

- **Tuple** - é uma coleção ordenada e imutável. Permite membros duplicados. São definidas usando parêntesis curvos (). Permitem armazenar vários valores numa só variável.

- **Dictionary** - é uma coleção ordenada e mutável. Não permite membros duplicados. São definidas usando chavetas {} com pares do tipo chave:valor.

- **Set** - é uma coleção não ordenada, imutável e não indexada. Não permite membros duplicados. São definidas usando chavetas {}. Permitem armazenar vários valores numa só variável.

Python – Estruturas de Dados

Listas (*lists*)

```
# definicao da lista
lista_nomes = ["pedro", "paulo", "maria", "luisa", "ana"]

# escrever todos os elementos da lista
print(lista_nomes)

# escrever o número de elementos da lista
print( len( lista_nomes) )

# acrescentar um novo elemento à lista
lista_nomes.append( "antonio" )
print(lista_nomes)
```

```
['pedro', 'paulo', 'maria', 'luisa', 'ana', 'antonio']
```

Usando a função `append()`, os novos elementos são acrescentados no fim da lista.

A função `insert()` permite inserir um novo item num índice específico.

Python – Estruturas de Dados

Listas

- As listas podem conter conjuntos de elementos de diferentes tipos de dados.
- A mesma lista pode conter elementos de tipos diferentes.
- Podem existir elementos duplicados.
- Os elementos estão dispostos numa ordem definida que não muda.

```
lista1 = ["pedro", "paulo", "maria"]

lista2 = [50, 10, 70, 90, 20]

lista3 = [True, False, False]

lista4 = [1, "luisa", 2, 3, "ana", 4 ]

lista5 = [1, 2, 3, 1, 2, 3]
```

lista com elementos de tipos diferentes

Python – Estruturas de Dados

Acesso aos Elementos de uma Lista

- Os elementos de uma lista podem ser acedidos individualmente através de um índice numérico.
- O primeiro elemento da lista é referenciado pelo índice 0.
- Podem-se usar índices negativos, sendo que -1 corresponde ao último elemento da lista, -2 ao penúltimo e assim sucessivamente.
- Podem-se definir intervalos de índices no formato [*índice_inicial*:*índice_final*] sendo que o *índice_final* já está excluído (p.e. [1:4] corresponde aos índices 1, 2 e 3).
- Nos intervalos de índices podem-se também usar índices negativos.
- É possível indicar ainda o passo, no formato [*índice_inicial*:*índice_final*: *passo*]

Python – Estruturas de Dados

Acesso aos Elementos de uma Lista

```
lista_nomes = ["pedro", "paulo", "maria", "luisa", "ana"]

print(lista_nomes[1:4])

print(lista_nomes[-2])

print(lista_nomes[0:5:2])
```

```
['paulo', 'maria', 'luisa']
luisa
['pedro', 'maria', 'ana']


# a função len() devolve o número de elementos da lista
print( len(lista_nomes) )
```

Python – Estruturas de Dados

Operações com Listas – Funções

```
lista_nomes = ["pedro", "paulo", "maria", "luisa", "ana"]

# inserir novo elemento no final
lista_nomes.append("antonio")

# remover um elemento
lista_nomes.remove("maria")

# remover um elemento pelo índice
lista_nomes.pop(1)

# remover um elemento pelo índice
del lista_nomes[3]

# inserir um novo elemento numa posicao especifica
lista_nomes.insert( 4, "manuela" )
```

```
['pedro', 'luisa', 'ana', 'manuela']
```

Python – Estruturas de Dados

Operações com Listas

```
lista1 = [1, 2, 3]

# percorrer todos os elementos de uma lista
for elemento in lista1:
    print( elemento, end=' ' )
```

```
1 2 3
```

```
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]

# junção de duas listas
lista3 = lista1 + lista2
print( lista3 )
```

```
[1, 2, 3, 4, 5, 6]
```

```
lista4 = [5, 2, 4, 3, 6, 1]

# escrever a lista em ordem crescente
lista4.sort()
print( lista4 )

# escrever a lista em ordem decrescente
lista4.sort( reverse=True )
print( lista4 )
```

```
[1, 2, 3, 4, 5, 6]
[6, 5, 4, 3, 2, 1]
```

Python – Estruturas de Dados

Operações com Listas

```
# preencher uma lista com elementos inseridos pelo utilizador

lista = []  # cria uma lista vazia

n = int(input( "Quantos elementos pretende inserir? "))

for i in range(n):
    e = int(input( "Insira o elemento {0}: ".format( i ) ) )
    lista.append( e )

# escreve toda a lista
print( lista )
```

Python – Estruturas de Dados

Listas - Funções

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python – Estruturas de Dados

Lists

Tutorial - https://www.w3schools.com/python/python_lists.asp

Exercícios - https://www.w3schools.com/python/python_lists_exercises.asp

Exercício

Ler uma lista de números inteiros sendo o número de elementos a considerar definido previamente pelo utilizador. Somar os elementos da lista, apresentar a média respetiva (com precisão de duas casas decimais) e escrever os elementos acima da média e os maior e menor valores da lista. Usar funções para calcular a soma, calcular a média, encontrar o maior valor, encontrar o menor valor e escrever os elementos acima da média.

Python – Estruturas de Dados

Tuplos (*tuples*)

Tal como uma lista, um tuplo é uma coleção ordenada mas **imutável**. Ser imutável significa que não podem ser acrescentados novos elementos nem eliminados ou alterados elementos existentes.

Tanto os tuplos como as listas permitem guardar vários valores na mesma variável.

```
tuplo1 = (1, 2, 3, 4, 5)

tuplo2 = ("manuel", "antonio", "maria", "pedro")

tuplo3 = (True, False, True)

tuplo4 = (1, "manuel", 2, "maria", True)
```

A função `len()` devolve o número de elementos do tuplo.

O acesso aos elementos de um tuplo é feito de forma idêntica às listas.

Python – Estruturas de Dados

Operações com tuplos

As operações com tuplos são semelhantes às operações com listas.

Embora os tuplos sejam imutáveis, é possível converter um tuplo numa lista, alterar a lista e converter a lista resultante no tuplo inicial:

```
tuplo = ("manuel", "antonio", "maria", "pedro")

lista = list(tuplo)

lista[2] = "joana"

lista.append( "luis" )

tuplo = tuple(lista)

print( tuplo )
```

```
('manuel', 'antonio', 'joana', 'pedro', 'luis')
```

Python – Estruturas de Dados

Tuplos - Funções

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Tuple

- https://www.w3schools.com/python/python_tuples.asp
- https://www.w3schools.com/python/python_tuples_exercises.asp

Python – Estruturas de Dados

Dicionários (*dictionaries*)

Um dicionário é uma coleção **ordenada** e **mutável**.

São conjuntos de pares do tipo *chave:valor*.

Não permitem valores duplicados.

```
aluno = { "nome": "pedro", "curso": "ei", "ano": 1 }

capitais = { "Portugal": "Lisboa",
             "Brasil"  : "Brasilia",
             "Espanha" : "Madrid",
             "França"  : "Paris"   }
```

A função `len()` devolve o número de elementos do dicionário.

Python – Estruturas de Dados

Operações com dicionários

```
# escrever o dicionário
print(aluno)

# aceder a um elemento
print( capitais.get("Portugal") )

# escrever todas as chaves
print( capitais.keys() )

# adicionar elemento
aluno["morada"] = "gaia"

# atualizar um elemento
aluno.update( { "ano": 2} )

# remover um elemento
capitais.pop("França")

# escever as chaves (keys) de um dicionario
for key in capitais:
    print(key)

# escever os valores (values) de um dicionario
for key in capitais:
    print(capitais[key])
```

```
aluno = { "nome": "pedro", "curso": "ei", "ano": 1 }

capitais = { "Portugal": "Lisboa",
             "Brasil"  : "Brasilia",
             "Espanha" : "Madrid",
             "França"  : "Paris"    }
```

chaves *(keys)*

valores

Python – Estruturas de Dados

Conjuntos – Funções

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Dictionary

- https://www.w3schools.com/python/python_dictionaries.asp
- https://www.w3schools.com/python/python_dictionaries_exercises.asp

Conjuntos (*sets*)

Um conjunto é uma coleção **não ordenada**, **imutável** (podem-se adicionar e eliminar elementos mas não se podem alterar elementos existentes) e **não indexada**.

Permitem também guardar vários valores na mesma variável.

Não permitem valores duplicados.

Podem ser criados conjuntos de qualquer tipo de dados; no mesmo conjunto podem existir elementos de diferentes tipos de dados.

```
set1 = {1, 2, 3, 4, 5}

set2 = {"manuel", "antonio", "maria", "pedro"}

set3 = {1, "manuel", 2, "maria", True}
```

A função `len()` devolve o número de elementos do conjunto.

Python – Estruturas de Dados

Operações com conjuntos

```
set1 = {1, 2, 3, 4, 5}
set2 = {"manuel", "antonio", "maria", "pedro"}

# adicionar um elemento
set1.add( 6 )

# remover um elemento
set2.remove( "antonio" )

print(set1)
print(set2)
```

```
{1, 2, 3, 4, 5, 6}
{'manuel', 'pedro', 'maria'}
```

```
# percorrer todos os elementos
for elemento in set1:
    print( elemento, end=' ' )
```

```
1 2 3 4 5 6
```

Python – Estruturas de Dados

Conjuntos – Algumas Funções

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
pop()	Removes an element from the set
remove()	Removes the specified element

Sets

- https://www.w3schools.com/python/python_sets.asp
- https://www.w3schools.com/python/python_sets_exercises.asp