



LISBON  
SCHOOL OF  
ECONOMICS &  
MANAGEMENT  
UNIVERSIDADE DE LISBOA

Carlos J. Costa

# PYTHON DATA STRUCTURES

- Learning Objectives
- Lists
- Tuples
- Sets
- Dictionaries



# Learning Objectives

- Know the main built-ins Python data structures
- Understand how to manipulate data organized in lists, tuples, sets and dictionaries
- Use built-ins Python data structures to solve problems



# Abstract Data Type (ADT)

- is a type (or class) for objects
- behaviour is defined by a set of value and a set of operations.

# Data Structure

- concrete representations of data,
- perspective of an implementer



# Data Structure

## Primitive

- Integer
- Float
- String
- Boolean

## Non-Primitive

- Array
- List
- Tuple
- Dictionary
- Set
- File



# Lists

- A list stores a series of items in a specific order
- You can access each item using an index or cycle
- The lists are mutable

```
# Construct a list  
shoppingList = ['potatoes', 'carrots', 'cod', 'sprouts']
```

Or else

```
shoppingList = list (('potatoes', 'carrots', 'cod', 'sprouts'))
```



# Lists

- Operations with lists

```
# Get the first element of the list  
shoppingList[0]
```

```
# Get the last element from the list  
shoppingList[-1]
```



# Lists

- Operations with lists

```
# Iterate though a list  
for purchase in shoppingList:  
    print(purchase)
```



# Lists

- Add Items

```
# Add an item to a list  
films = []  
films.append('Vice')  
films.append('Green Book')  
films.append('Roma')  
films.append('A Star Is Born')  
print(films)
```



# Lists

- Remove elements

```
films.pop()
```

```
films.remove('A Star Is Born')
```

```
del films[1]
```

```
print(films)
```

- Delete List elements:

```
films.clear
```

- Delete list

```
del films
```



# Lists

- List comprehension

```
# Compress list
```

```
squares = [x**2 for x in range(1, 11)]
```



# Lists

- **Slicing the list**

```
# Obtain the first 3 elements of the list
shoppingList = ['potatoes', 'carrots', 'cod', 'sprouts']
firstThree = shoppingList [: 3]
print (firstThree)
```



# Lists

- What's the result of?

```
shopping = shoppingList
shoppingList.append("orange")
print(shopping)
```



# Lists

- Yes.. Do not copy... To copy, you make:

```
# copy a list  
shoppingListCopy = shoppingList[::]
```



# Tuples

- The tuples are identical to the lists, but they cannot be modified
- They are immutable

```
newPurchases= ("bananas", "beans", "rice")
print (newPurchases [1])
newPurchases [0] = "apple"
```

- What is the output?



# Dictionaries

- Dictionaries store links between pieces of information
- Each item in a dictionary is a key-value pair
- Keys are not repeatable

```
fruit = {1: 'orange', 2: 'apple', 3: 'pear', 4: 'grape', 5:  
'peach'}
```



# Dictionary

- Add a new key-value pair

```
fruit[10] = 'pomegranate'
```

- Iterating through key-value pair

```
for key , value in fruit.items():  
    print('The fruit' + str(key) +' is ' + value)
```



# Dictionary

- Iterating through the key

```
for key in fruit.keys():  
    print(str(key) + ' is fruit')
```

- Iterating through the values

```
for value in fruit.values():  
    print(value + ' is fruit ')
```



# Sets

- Sets
- These are structures available in Python, used to represent unordered collections of unique elements

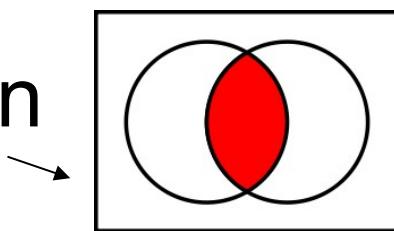
```
s = {1, 2, 3, 4}  
print (s)
```



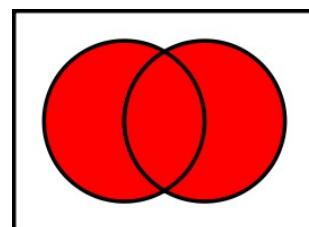
# Conjuntos

- In sets, a set of typical operations of mathematical set theory can be performed, such as:

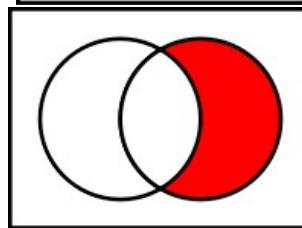
Intersection



Union



Difference



Symmetric Difference

