LEARNING TO CODE

python™

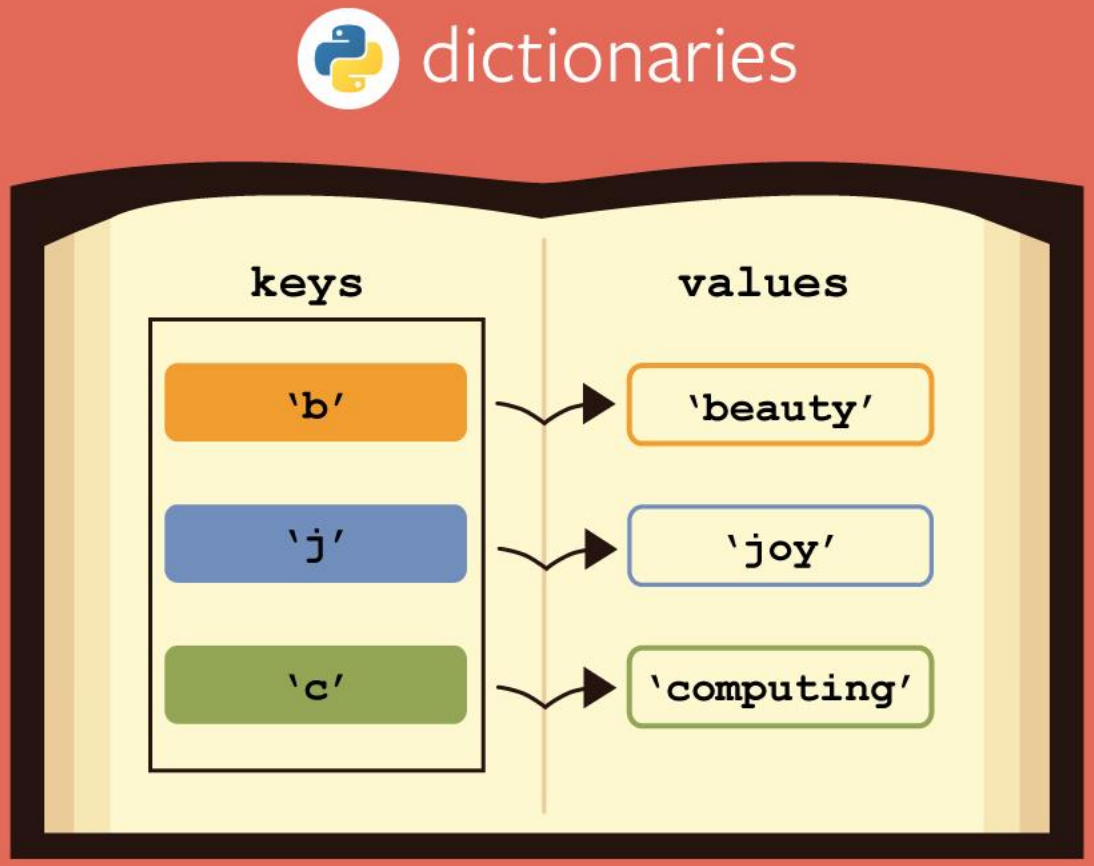
Coordinator: Konstantinos Emmanouilidis

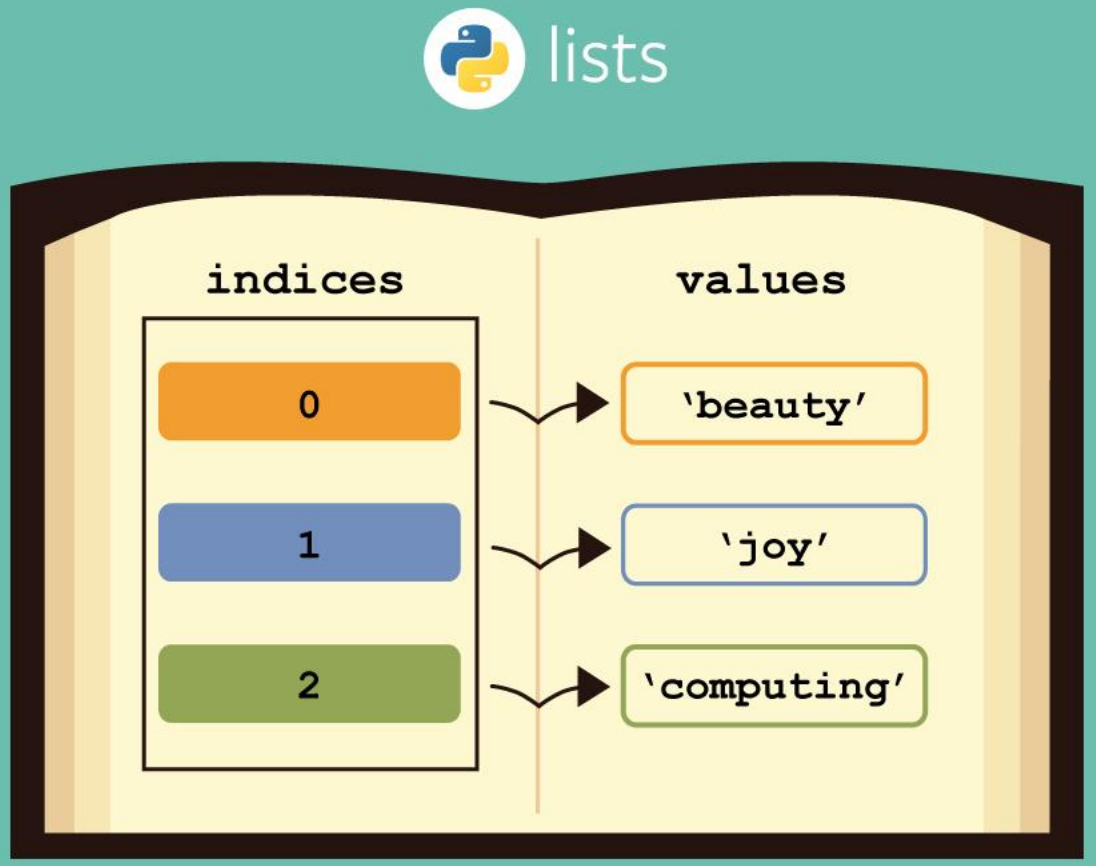
IEEE SB OF THRACE

2017-2018

- DICTIONARIES
- CLASSES

THE DICTIONARY DATA TYPE





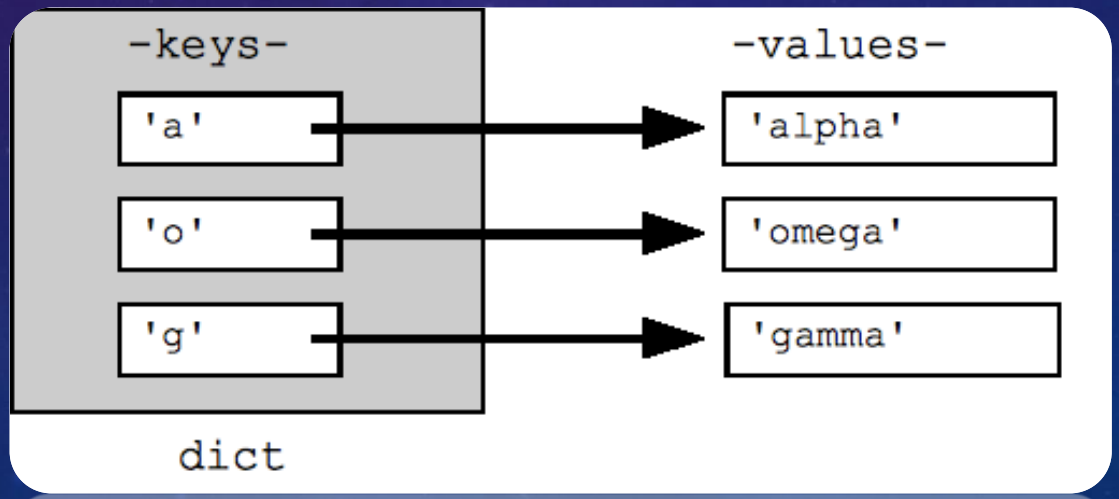
THE DICTIONARY DATA TYPE

- **DECLARATION:** with {}

  myDict={key1 : value1 , key2 : value2 , key3 : value3 ,…}

Example:

```
>>> MyDict={'a':'aplha','o':'omega','g':'gamma'}
```



THE DICTIONARY DATA TYPE

- **Accessing the value of a key:** myDict[key]

Example:

```
>>> myDict['a']
'alpha'
>>> myDict['o']
'omega'
>>> myDict['g']
'gamma'
```

DICTIONARIES VS LISTS

- Keys in dictionaries can use many different data types. Indices in lists use only integers.
- The fact that you can have arbitrary values for the keys allows you to organize your data in powerful ways.
- Unlike lists, items in dictionaries are unordered.
- The order of items matters for determining whether two lists are the same. In a dictionary the order of key-values pairs does not matter.
- Dictionaries can’t be sliced

METHODS FOR DICTIONARIES

- **keys():** returns list-like values of the dictionary's keys
- **values():** returns list-like values of the dictionary's values
- **items():** returns list-like values of the dictionary's keys and values

```
>>> spam = {'color': 'red', 'age': 42}
>>> for v in spam.values():
        print(v)

red
42
```

```
>>> for k in spam.keys():
        print(k)

color
age
```

```
>>> for i in spam.items():
        print(i)

('color', 'red')
('age', 42)
```

CHECKING IF A KEY/VALUE EXISTS IN A DICTIONARY

- Use membership operators (in,not in).

Example:

```
>>> spam = {'name': 'Zophie', 'age': 7}
>>> 'name' in spam.keys()
True
>>> 'Zophie' in spam.values()
True
>>> 'color' in spam.keys()
False
>>> 'color' not in spam.keys()
True
>>> 'color' in spam
False
```

OBJECT ORIENTED PROGRAMMING

CLASSES

- They are the basic element of Object Oriented Programming.
- A way to make new type of data.
- Provide flexibility through polymorphism.
- Reuse code with inheritance.

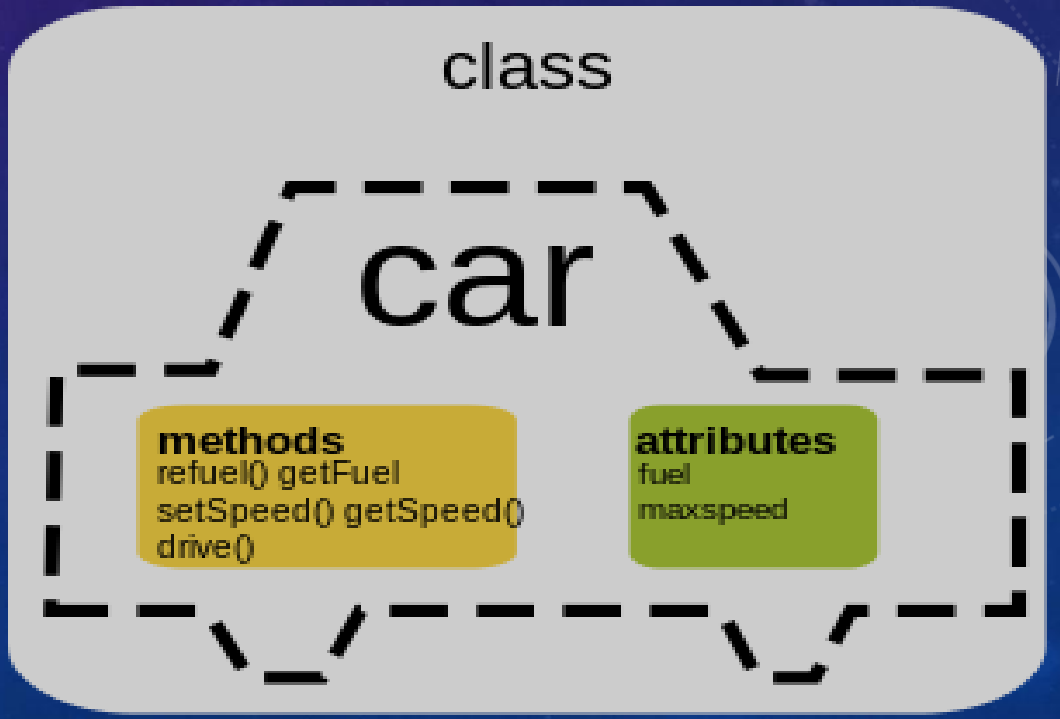
CLASSES

- The members of a class are variables and methods(functions).
- **Declaration:**

  class Myclass**:**

    class'_variables

    class'_methods



THE SELF VARIABLE

- Every function defined in a class has as first parameter the variable *self.*
- *self* refers to the object that calls the function.
- You can use self.variable inside a method of a class.

EXAMPLES

script.py

```
1   # Prints out the numbers 0,1,2,3,4
2   for x in range(5):
3       print(x)
4
5   # Prints out 3,4,5
6   for x in range(3, 6):
7       print(x)
8
9   # Prints out 3,5,7
10  for x in range(3, 8, 2):
11      print(x)
```

Result:

IPython Shell

```
0
1
2
3
4
3
4
5
3
5
7
```

script.py

```
1   primes = [2, 3, 5, 7]
2   for prime in primes:
3       print(prime)
```

Result:

script.py

```
1   primes = [2, 3, 5, 7]
2   for prime in primes:
3       print(prime)
```

CONSTRUCTORS

- Function called when a object is created.
- Declaration:

def _init_(self,parameters):

commands

.

.

.

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

DESTRUCTORS

- Function called when an object is destroyed.
- Used especially when there is dynamic memory allocation.
- Declaration:

```
def _del_(self):
commands
.
.
.
```