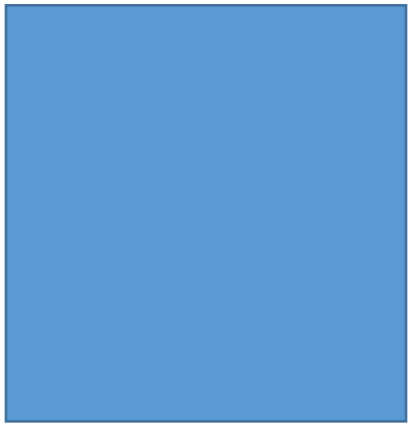
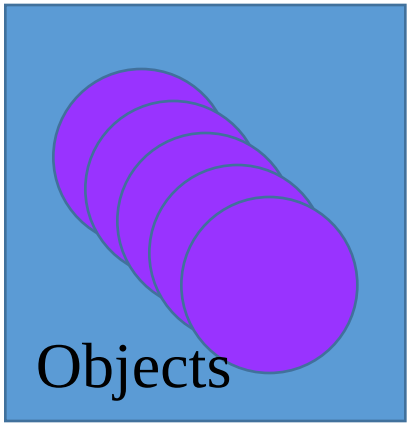
CMPUT 175 - Lab2
Winter 2018

Classes in Python

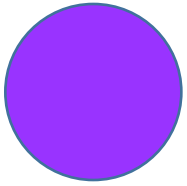
Class



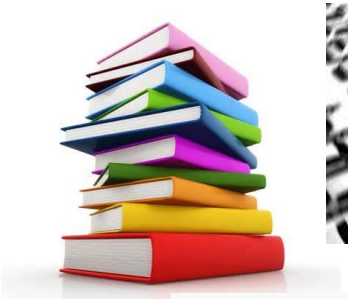
Class

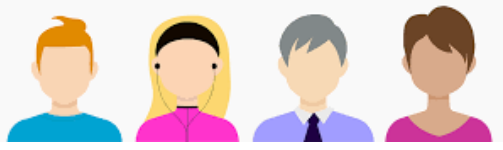


- You can define a new Abstract Data Type (ADT) by defining a new class in Python
- A class can have many instances
- For example you can have a class for books, a class for cars, a class of students, etc.



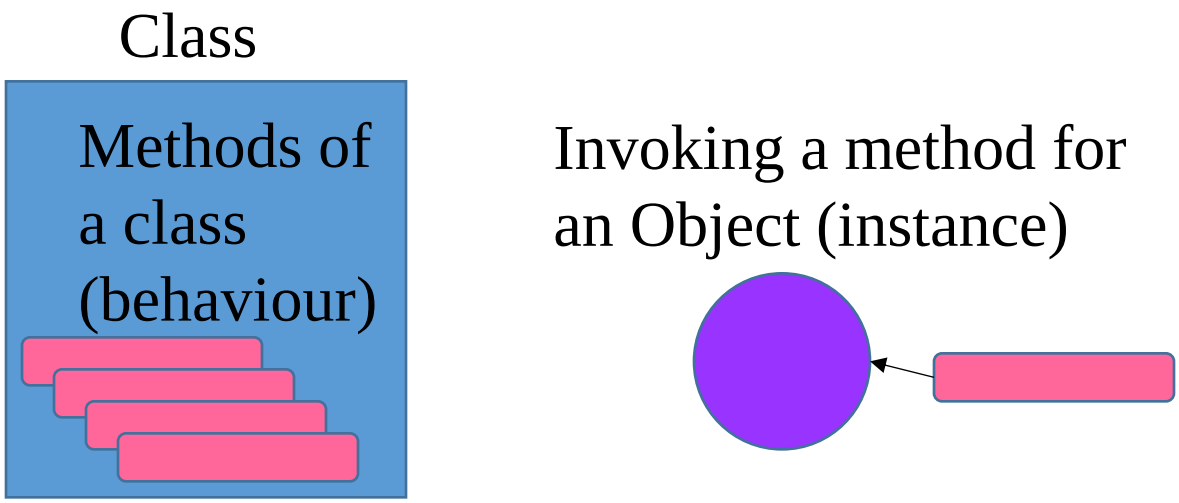
One Object instance from the Class





Classes in Python

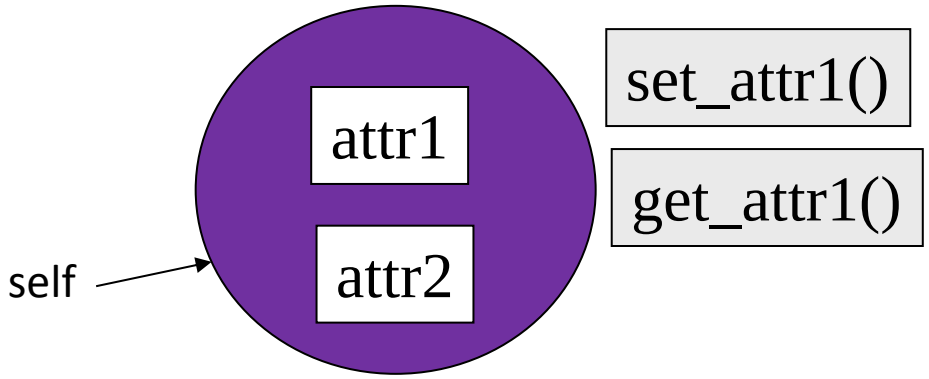
- A class can have methods that describe the expected behaviour of instances of the class
- You create a new instance for a class myClass by assigning anInstance=myClass(*arguments*)
- You can invoke a method for an instance by calling anInstance.myMethod(*arguments*)



Classes in Python

- Instances of a class have attributes. All instances of the same class have the same attributes but possibly with different values.
- For example all cars have a colour, a model, a horsepower value, etc.
- Methods are functions or procedures defined in the class by def menthodName (*arguments*)
- self is always the first arguments when defining methods. It is a reference that is bound to the instance.
- The constructor is called __init__() which creates a new instance and initializes its attributes

General Example



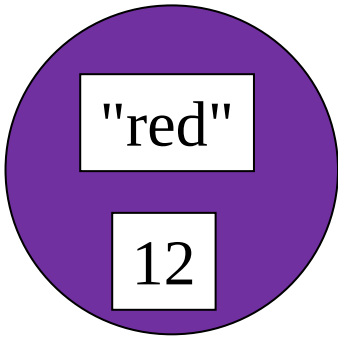

```
class class_name :
    def __init__ (self , init_attr1 , init_attr2 ):
        self . attr1 = init_attr1
        self .attr2 = init_attr2
    def get_attr1 ( self ):
        return self .attr1
    def set_attr1 (self , new_attr1 ):
        self .attr1 = new_attr1
```

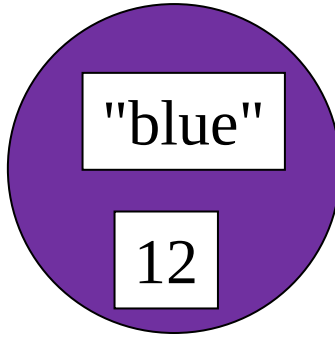
General Example, cont'd

```
def main ():
    # creating an instance of the class
    oneInstance = class_name ("red ", 12)

    # invoking get_attr1() method of the instance
    print ( oneInstance . get_attr1 ()) # red

    oneInstance . set_attr1 (" blue ")
    print ( oneInstance . get_attr1 ()) # blue
```





Dictionary in Python

- A Dictionary is a collection of associated (key, value) pairs, such that each possible key appears just once in the collection.
- A pair consists of a key and a value (key: value)

AB	Edmonton
BC	Victoria
ON	Toronto

Capitals={"AB":"Edmonton","BC":"Victoria","ON":"Toronto"}

- Like Lists, Dictionaries are mutable
- Unlike Lists, elements in a Dictionary do not have an order
- Values are accessed via keys : Capitals["AB"]
- Think of it as an unordered set of unique keys where each key is associated with a value. This value can be anything.

Using Dictionaries

- Looking up the value associated with a particular key
- Getting all keys
- Getting all values
- Getting all pairs
- Adding pairs to the collection
- Modifying the values of existing pairs
- Removing pairs from the collection

Basic Syntax in Dictionary (Lookup)

```
>>> phone={"7804922860": "Osmar Zaiane", "7804923330":"Anup Basu"}
>>> phone["7804922860"]
'Osmar Zaiane'
>>> phone["7804923330"]
'Anup Basu'
>>> phone["7801234567"]
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    phone["7801234567"]
KeyError: '7801234567'
>>>
>>> phone.keys()
dict_keys(['7804922860', '7804923330'])
>>>
>>> phone.values()
dict_values(['Osmar Zaiane', 'Anup Basu'])
>>>
>>> phone.items()
dict_items([('7804922860', 'Osmar Zaiane'), ('7804923330', 'Anup Basu')])
>>>
```

View or lookup in a dictionary

a key that doesn't exist

getting all keys

getting all values

getting all pairs

Basic Syntax in Dictionary (Add)

```
>>> phone={"7804922860": "Osmar Zaiane", "7804923330":"Anup Basu"}
>>> phone
{'7804922860': 'Osmar Zaiane', '7804923330': 'Anup Basu'}
>>> phone["7804922253"]="Duane Szafron"
>>> phone
{'7804922860': 'Osmar Zaiane', '7804922253': 'Duane Szafron', '7804923330': 'Anup Basu'}
>>>
```

Assign value to a new key

Note that the order is not the order of insertion
Dictionaries are unordered.

```
>>> for k in sorted(phone.keys()):
        print (k, phone[k])
```

Get content sorted on key

```
7804922253 Duane Szafron
7804922860 Osmar Zaiane
7804923330 Anup Basu
>>>
```

Basic Syntax in Dictionary (Modify)

```
>>> phone={"7804922860": "Osmar Zaiane", "7804923330":"Anup Basu"}
>>> phone
{'7804922860': 'Osmar Zaiane', '7804923330': 'Anup Basu'}
>>> phone["7804923330"]=“Someone Else"
>>> phone
{'7804922860': 'Osmar Zaiane', '7804923330': ‘Someone Else'}
>>>
```

Assign value to existing key

Keys are unique.

Assigning to an existing key just replaces its value

Basic Syntax in Dictionary (Delete)

```
>>> phone={"7804922860": "Osmar Zaiane", "7804923330":"Anup Basu"}
>>> phone
{'7804922860': 'Osmar Zaiane', '7804923330': 'Anup Basu'}
>>> phone["7804922253"]="Duane Szafron"
>>> phone
{'7804922860': 'Osmar Zaiane', '7804922253': 'Duane Szafron', '7804923330': 'Anup Basu'}
>>>
```

del removes a key and its value

```
>>> del phone["7804923330"]
>>> phone
{'7804922860': 'Osmar Zaiane', '7804922253': 'Duane Szafron'}
>>>
```

The method *clear()* removes all pairs

```
>>> phone.clear()
>>> phone
{}
>>>
```