Create a Class in Python

Shouke Wei, Ph.D. Professor

Email: shouke.wei@gmail.com

1. Classes and Objects

Object

- Python is an object oriented programming language
- An object is simply a collection of properties/attributes (data (variables) and methods (functions)

Classes

- A class is a set or category of things with similar propertyies or attributes
- In terminology, a class is something as an object constructor, or a "blueprint" for creating objects
  - A Class defines the structure, property, and behaviour of an Object.
- Object is one of instances of the class, which can perform the functionalities defined in the class.
- If a blueprint of a house is a class, houses created from the blueprint are objects
- As a blueprint can produce many houses, a class can create many objects

2. Create a Class

Step 1: Create a Class

- Suppose we will creat a class of persons, named **person()**
- To create a class, use the keyword `class`

In [1]:
```
class person():
    pass
```

Step 2: Create objects

`__init__()` Function

- A class usually has a `__init__()` function
- It is used to assign values to object properties
- Suppose the person class has `properties` / `attributes` : 'name', 'gender', 'age', 'origin' and 'phone'

In [4]:
```
class person():
    def __init__(name, gender, age, origin, phone):
        pass
```

- `__init__()` function is executed when the class is being initiated
- It allows the class to initialize the attributes of the class

`self` Parameter

- It represents current instance itself
- It is used to access the attributes and methods of the class
- It has to be the first parameter of any function in the class

In [4]:
```
class person():
    def __init__(self, name, gender, age, origin, phone):
      self.name = name
      self.gender = gender
      self.age = age
      self.origin = origin
      self.phone = phone
```

- `self` does not have to be named self, you can use other names

Create more functions

- info_func()

In [2]:
```
class person():
    def __init__(self, name, gender, age, origin, phone):
      self.name = name
      self.gender = gender
      self.age = age
      self.origin = origin
      self.phone = phone

    def info_func(self):
        print(f'This is {self.name}, {self.gender}, {self.age} years old.')

        if self.gender == 'male':
            print(f'He is from {self.origin}, and his phone number is {self.phone}.')
        else:
            print(f'She is from {self.origin}, and her phone number is {self.phone}.')
```

Step 3: Create instances

Instance

- An instance is an actual object built from the blueprint.
- It has a physical presence in memory and can be called upon to do work.

In [3]:
```
person1 = person('Susan', 'female', 23, 'Canada', 77788999)
person2 = person('Jack', 'male', 21, 'USA', 22188966)
```

- These values are assigned to the attributes of the class through `__init__` function

Step 4: Excute the classs

In [7]:
```
person1.name # access the attributes
```

Out[7]:
```
'Susan'
```

In [8]:
```
print(person1.name)
```

```
Susan
```

In [9]:
```
person2.age
```

Out[9]:
```
21
```

In [10]:
```
print(person2.age)
```

```
21
```

In [11]:
```
person2.info_func() # run the methods
```

```
This is Jack, male, 21 years old.
He is from USA, and his phone number is 22188966.
```

The attributs and methods are accessed through the `self` parameter, which represents person1 and person2 instances.

3. Inheritance

- **Inheritance**: defines a class that inherits all the methods and properties from another class.
- **Parent class**: the class being inherited from, also called base class.
- **Child class**: the class that inherits from another class, also called derived class.

3.1 Create a Parent Class

- Any class can be a parent class, whose syntax is the same as creating any other class
- So let us use the `person` class as our parent class

In [12]:
```
class person():
    def __init__(self, name, gender, age, origin, phone):
      self.name = name
      self.gender = gender
      self.age = age
      self.origin = origin
      self.phone = phone

    def info_func(self):
        print(f'This is {self.name}, {self.gender}, {self.age} years old.')

        if self.gender == 'male':
            print(f'He is from {self.origin}, and his phone number is {self.phone}.')
        else:
            print(f'She is from {self.origin}, and her phone number is {self.phone}.')
```

3.2 Creat a Child Class

- Suppose we create a child class named "student", which inherits the functionality from a parent class, the "person"
- We just put the parent class as a parameter to the child class

In [13]:
```
class student(person):
  pass
```

pass *keyword means that you do not want to add any other properties or methods to the class.*

- Now we have the **student** child class,which has the same properties and methods as the **person** parent class.

3.3 Creat instance and assign arguments

In [14]:
```
student1 = student('John', 'male', 21, 'Germany', 55788999)
```

3.4 Run the method and test the child class

In [15]:
```
print(student1.phone)
```

```
55788999
```

In [16]:
```
student1.info_func()
```

```
This is John, male, 21 years old.
He is from Germany, and his phone number is 55788999.
```