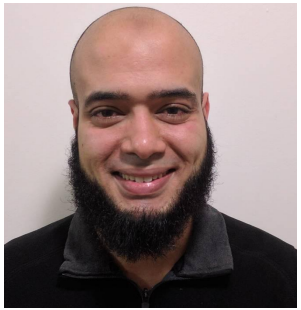
Python Programming

Inheritance 2: Single Inheritance

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Back to the Student vs Teacher

```python
class Student:
    def __init__(self):
        self.name = None
        self.email = None
        self.address = None
        self.national_id = None
        self.starting_study_year = None
        self.GPA = None
        self.studied_courses = []

    def is_valid_email(self, email)
        pass
    def add_course_frade(self, course_id, grade):
        pass
    def print_grades(self):
        pass
```

```python
class Teacher:
    def __init__(self):
        self.name = None
        self.email = None
        self.address = None
        self.national_id = None
        self.starting_employement_year = None
        self.current_salary = None
        self.teaching_courses = []

    def is_valid_email(self, email):
        pass

    def add_course(self, course_id):
        pass
```

- How can we avoid **duplicating** code in this problem?
  - Inheritance allow us to reuse code!

Reusability

- Code **Reusability**
  - The **ability** to reuse the existing coding efforts for a **new** usage instead of duplicating efforts
    - These codes are written, tested and bug-fixed! Resources were consumed in that
    - In practice: it is challenging to apply in large-scale projects. Reading
- OO has 2 reusability approaches: ***Inheritance and Composition***
  - Allows building **hierarchy** of classes and relations
  - Composition: **has-a** relationship
    - A building **has** floors. A floor **has** apartments.
    - A car **has-an** engine and **has** 4-wheels
  - Inheritance: **is-a** relationship
    - Manager **is-an** employee
    - Student **is-a** person

Inheritance in Python

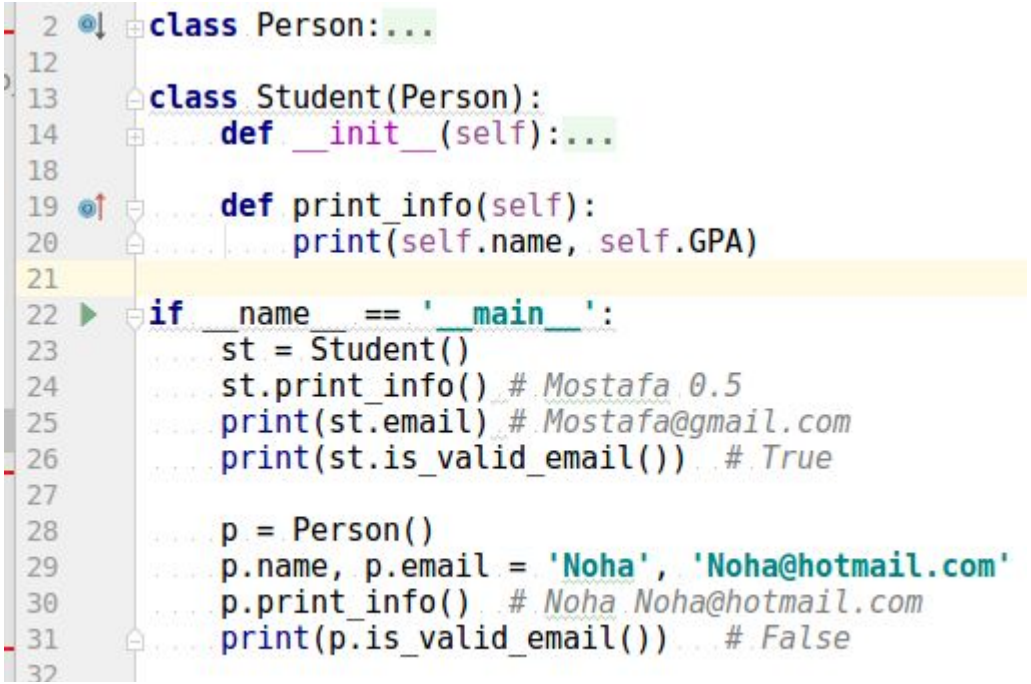
- When Class A **inherits** Class B, it inherits its created **attributes, properties & methods:** We call A (Parent/Base) and B (Child/Derived)
  - Here Person is Base and Student is Derived

```python
class Person:
    def __init__(self):
        self.name = 'Mostafa'
        self.email = 'Mostafa@gmail.com'

    def is_valid_email(self):
        return self.email.endswith('@gmail.com')

    def print_info(self):
        print(self.name, self.email)
```

```python
class Student(Person):
    def __init__(self):
        Person.__init__(self)   # Call parent init
        self.GPA = .5
        self.studied_courses = ['C++', 'Python']

    def print_info(self):
        print(self.name, self.GPA)
```

Inheritance in Python

- The parent neither know nor affected by the child
- Student class has print_info in base class, but then newone **override** it
  - Think: Reassign variable



Isinstance, issubclass and type!

```
print(type(st))      # <class '__main__.Student'>
print(isinstance(st, Student))          # True
print(isinstance(st, Person))           # True

print(type(st) is Student)              # True
print(type(st) is Person)               # False
print(type(st) in [Student, Person])    # True

print(issubclass(Student, Person))      # True
print(issubclass(Student, Student))     # True
print(issubclass(Student, list))        # False
#print(issubclass(st, Person))          # Error: class NOT object
print(issubclass(type(st), Person))     # True

# Be careful from type vs isinstance
```

The Object Super Class

- **Every** class in python is subclass from the object class (implicit)
  - Recall: everything in python is object, including functions and modules

```
print(issubclass(Person, object))       # True
print(issubclass(Student, object))      # True
print(issubclass(list, object))         # True
print(issubclass(int, object))          # True: int is object

import math
print(isinstance(math, object))          # True: moudle is object
print(isinstance(math.sqrt, object))     # True
```

Let’s create a super object!

```
# we actually inherit its attributes & methods
obj = object()
print(dir(obj)) # ['__class__', '__delattr__', '__dir__', '__doc__', '_
print(obj.__init__)
print(obj.__repr__())    # 0x7ff2b4169b90 default print memory address
print(object.__name__)   # on class level
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."