Python Inheritance

1. Inheritance
2. Redefinir el constructor
3. Super()
4. Afegir variables i funcions

Inheritance

1.Inheritance

Python és un llenguatge de programació interpretat **orientat a objectes**. Com a tal, disposem de classes i instàncies de les mateixes (objectes). Així com del concepte d'**herència** *(inheritance).*

- Qualsevol classe pot jugar el rol de *classe parent* o classe base i així tenir herència.
- Una *classe child* és aquella que n'ha heretat una altra *classe parent*.
- Una *classe child* també pot tenir herència.
- A la *classe child* podem redefinir les variables i mètodes de la classe base així com afegir-ne de nous.

1.Inheritance

Classe Parent:

```
class Person:
    def __init__(self, f_name, l_name):
        self.first_name = f_name
        self.last_name = l_name
    # Definim un mètode public per imprimir el nom de la persona
    def print_name(self):
        print(self.first_name + " " + self.last_name)
```

```
# Usem la classe Person per crear un objecte i després executem el mètode
públic:
x: Person = Person("John", "Doe")
x.print_name()
```

```
>>> John Doe
```

1.Inheritance

Classe Child:

```
class Student(Person):
    pass
```

- La classe Student **rep la classe Person com a paràmetre**.
- Així la classe Student hereta de Person i Person juga el rol de classe base.
- Com que hem fet servir *pass* dins del codi de Student, aquesta classe child es comporta exactament igual que la classe parent de la qual hereta.

Redefinir el constructor

2. Redefinir el constructor

Dins la classe child podem redefinir el comportament del constructor de la classe base.

- Si a la classe child redefinim el comportament del constructor específic __init__, aquest mètode ja no heretarà de la classe pare i, en conseqüència, ja no tindrà el mateix comportament.
- Podem redefinir el comportament del constructor, però obligar-lo a heretar de la classe pare amb el resultat de no aplicar cap canvi.

```
class Student(Person):
    def__init__(self, f_name, l_name):
        Person.__init__(self, f_name, l_name)
```

Super()

3. Super()

El concepte **super()** s'usa en herència de Python per referir-se a la classe pare.

Equivalents:

```
class Student(Person):
    def__init__(self, f_name, l_name):
        Person.__init__(self, f_name, l_name)
```

```
class Student(Person):
    def__init__(self, f_name, l_name):
        super().__init__(f_name, l_name)
```

Afegir variables i funcions

4. Afegir variables i funcions

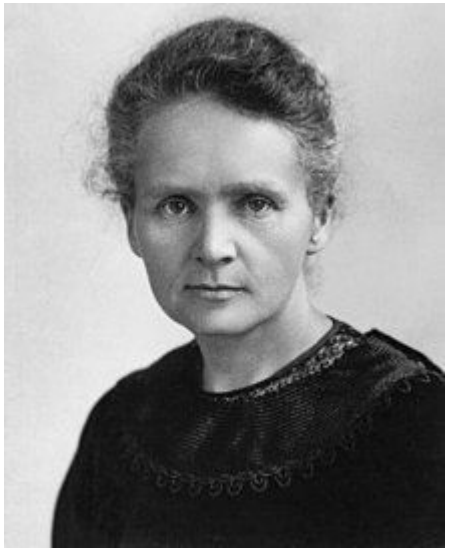
En herència de Python podem **afegir variables** i **funcions** a les classes child.

```
class Student(Person):
    def __init__(self, f_name, l_name, year):
        super().__init__(f_name, l_name)
        self.graduation_year = year

    def welcome(self):
        print("Welcome ", self.first_name , self.last_name , " to the
class.")
```

```
x: Student = Student("Pitter", "Griffin", 2019)
x.welcome()

>>> Welcome PitterGriffin to the class.
```



"No cal témer res a la vida, només tractar de comprendre"

Marie Curie (1867 - 1934)
Inventora dels Rajos X

Raimon Izard