**OBJECT ORIENTED PROGRAMMING**

**INTRODUZIONE**

La programmazione OOP è un tipo di programmazione più comodo della programmazione procedurale, ossia della programmazione vista finora.

**CLASSI**

Per creare una classe in python si utilizz la keyword class. Al suo interno si crea un costruttore, contenente le VARIABILI della classe. Tale costruttore si inizializza tramite la keyword def, keyword già utilizzata per l’introduzione delle funzioni. Vediamo un esempo banale di classe.

Codice Python:

class lavoratore:

    def \_\_init\_\_(self):

        self.nome = "Arnaldo"

        self.cognome="Blablo"

        self.data\_di\_nascita = 1980

Per accedere alla classe, si utilizza la seguente sintassi strana

Codice Python:

class lavoratore:

    def \_\_init\_\_(self):

        self.nome = "Arnaldo"

        self.cognome="Blablo"

        self.data\_di\_nascita = 1980

lavoratore = lavoratore()

print(lavoratore)

print()

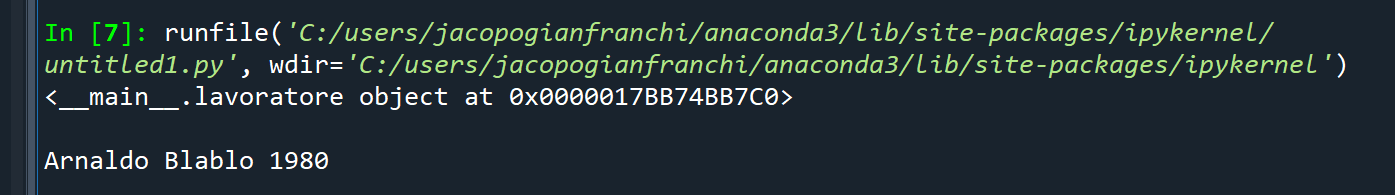
nome\_lavoratore = lavoratore.nome

cognome\_lavoratore = lavoratore.cognome

nascita\_lavoratore = lavoratore.data\_di\_nascita

print(nome\_lavoratore,cognome\_lavoratore,nascita\_lavoratore)

Console:



Solitamente i valori non vengono definiti all’interno della classe, ma vengono passati come parametri.

Codice Python:

    def \_\_init\_\_(self,nome,cognome,data\_di\_nascita):

        self.nome = "Arnaldo"

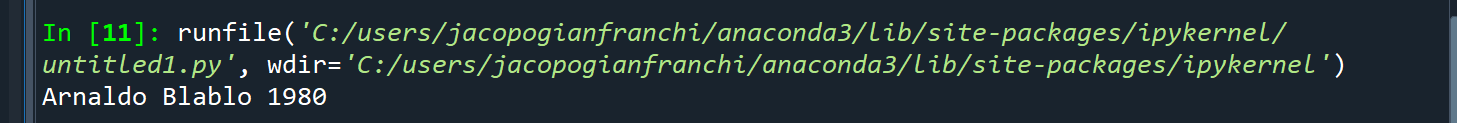
        self.cognome="Blablo"

        self.data\_di\_nascita = 1980

lavoratore1 = lavoratore("mario","Bla",1290)

print(lavoratore1.nome,lavoratore1.cognome,lavoratore1.data\_di\_nascita)

Console:



Si noti che, anche se abbiamo passato dei valore per i parametri nome, cognome, età, sono stati stampati i valori di default. Questo è normale, perché abbiamo forzato il valore all’interno della classe. Per rendere dinamici i valori, si passano gli elementi del costruttore dopo self.valore = :

Codice Python

class lavoratore:

    def \_\_init\_\_(self,nome,cognome,data\_di\_nascita):

        self.nome = nome

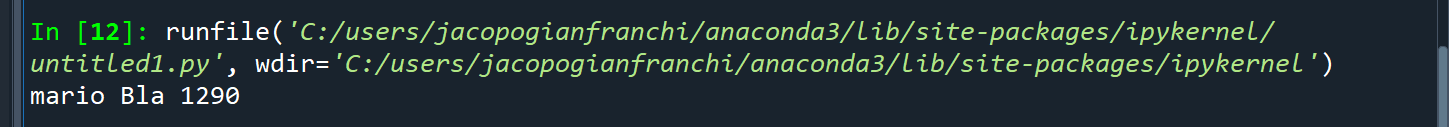
        self.cognome=cognome

        self.data\_di\_nascita = data\_di\_nascita

lavoratore1 = lavoratore("mario","Bla",1290)

print(lavoratore1.nome,lavoratore1.cognome,lavoratore1.data\_di\_nascita)

Console:



Finora abbiamo inserito solo variabili all’interno della classe.

Possiamo anche creare metodi e richiamarli all’esterno.   
Costruiamo quindi una classe che presenta sia variabili sia metodi, e richiamiamo i metodi al loro interno:

Codice Python:

class rettangolo:

    def \_\_init\_\_(self,base,altezza):

        self.base=base

        self.altezza=altezza

    def area\_rettangolo(self):

        return f"La base è {self.base} e l'altezza è  {self.altezza}. L'area del rettangolo è {self.base\*self.altezza}"

    def perimetro\_rettangolo(self):

        return f"La base è {self.base} e l'altezza è  {self.altezza}. Il perimetro del rettangolo è {self.base\*2 + self.altezza\*2}"

rettangolo1 = rettangolo(10,30)

# Calcola area del rettangolo:

print(rettangolo1.area\_rettangolo())

print()

# Calcola area del rettangolo:

print(rettangolo1.perimetro\_rettangolo())

Console:

