Corso ITS: ARTIFICIAL INTELLIGENCE SPECIALIST

Modulo: Programmazione ad oggetti in Python e librerie esterne

Docente: Andrea Ribuoli

Martedì 18 Febbraio 2025

09:00 - 14:00

- diversamente da quanto fatto sinora non saremo più solo utilizzatori di classi
- nella definizione di una classe apriamo un blocco di istruzioni di più alto livello
- all'interno di un class Figura: troveremo una serie di def
- con la stessa sintassi con cui definivamo funzioni qui creaimo i metodi della classe

```
In [1]: class Figura :
    def __init__(self, initialVolume = 1.0) :
        self._volume = initialVolume

def volume(self) :
    return self._volume

def volumeSet(self, initialVolume = 1.0) :
    self._volume = initialVolume
```

- una prima particolarità è il primo parametro: self
- questo verrà sempre impostato e consentirà di referenziare le variabili di istanza
- cioè i dati memorizzati separatamente per ogni istanziazione della classe
- ossia per ogni **oggetto**
- il metodo avente nome __init__ è il costruttore degli oggetti della classe
- potremmo persarlo come la funzione implicitamente invocata
- questo avviene ogni qualvolta creiamo un oggetto con a = Figura()

```
In [2]: a = Figura()
print(type(a))
```

1 di 3

```
<class '__main__.Figura'>
```

• valgono le considerazioni fatte per la ispezione dei metodi offerti...

```
In [3]: metodi = dir(a)
    for metodo in metodi :
        if not metodo.startswith('__') :
            print(metodo)

_volume
    volume
    volumeSet
```

- le variabili di istanza hanno un nome che inizia per _ (underscore)
- ogni istanza ne avrà un insieme dedicato e distinto
- nella scrittura dei metodi le variabili di istanza vanno qualificate con self.
- i metodi che restituiscono il valore di una variabile di istanza sono detti anche "getter"
- i metodi che consentono di modificare il valore di una variabile di istanza sono detti anche "setter"
- lo stato di un oggetto viene descritto da un numero specifico di variabili di istanza
- siano esse inizializzate o meno durante la creazione mediante parametri forniti al costruttore

```
In [4]: class Figura :
    def __init__(self, initialVolume = 1.0, initialWeight = 1.0) :
        self._volume = initialVolume
        self._peso = initialWeight
    def volume(self) :
        return self._volume
    def weight(self) :
        return self._peso
    def volumeSet(self, initialVolume = 1.0) :
        self._volume = initialVolume
    def weightSet(self, initialWeight = 1.0) :
        self._peso = initialWeight
```

- ora abbiamo introdotto una seconda variabile di istanza e altri due metodi (1 setter e 1 getter)
- possiamo tuttavia creare metodi specifici al problema che la classe creata vuole indirizzare

```
In [5]: class Figura :
    def __init__(self, initialVolume = 1.0, initialWeight = 1.0) :
        self._volume = initialVolume
        self._peso = initialWeight
    def volume(self) :
```

2 di 3

```
return self._volume
def weight(self) :
    return self._peso
def volumeSet(self, initialVolume = 1.0) :
    self._volume = initialVolume
def weightSet(self, initialWeight = 1.0) :
    self._peso = initialWeight
def specific_weight(self) :
    if self._volume == 0.0 :
        return "Volume non impostato"
    return self._peso/self._volume
```

```
In [6]: a = Figura()
    metodi = dir(a)
    for metodo in metodi :
        if not metodo.startswith('___') :
            print(metodo)
```

_peso
_volume
specific_weight
volume
volumeSet
weight
weightSet

3 di 3