Programmazione avanzata a.a. 2021-22

A. De Bonis

Introduzione a Python

II parte della V lezione

19

Python e OOP

- Python supporta tutte le caratteristiche standard della OOP
  - Derivazione multipla
  - Una classe derivata può sovrascrivere qualsiasi metodo della classe base
- Tutti i membri di una classe (dati e metodi) **sono pubblici**

20

Ereditarietà

- Le superclassi di una classe vengono elencate tra le parentesi nell'intestazione della classe
- Le superclassi potrebbero trovarsi in altri moduli
  - Esempio: supponiamo che FirstClass sia nel modulo modulename

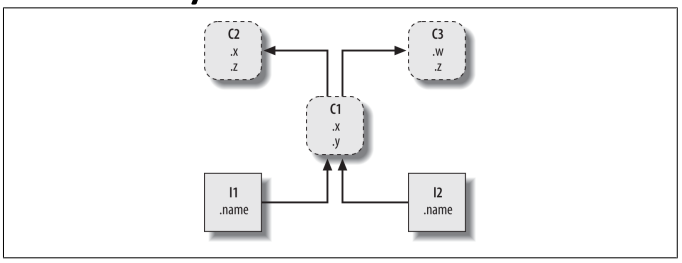
```
from modulename import FirstClass
class SecondClass(FirstClass):
    def display(self): ...
```

oppure

```
import modulename
class SecondClass(modulename.FirstClass):
    def display(self): ...
```


21

Python e OOP



- I1.w viene risolto in C3.w
- Python cerca l'attributo nell'oggetto e poi risale man mano nelle classi sopra di esso dal basso verso l'alto e da sinistra verso destra
  - I2.z viene risolto in C2.z

22

Classi in Python

- In Python in una classe possiamo avere
  - variabili di istanza (dette anche membri dati)
  - variabili di classe
    - condivise tra tutte le istanze della classe
  - metodi (detti anche membri funzione)
    - metodi specifici della classe
    - overloading di operatori
- Per far riferimento ad una variabile di istanza si fa precedere l'identificatore dalla parola chiave **self**
  - se non esiste una variabile di istanza con lo stesso nome, **self** puo` essere usato anche per far riferimento ad una variabile di classe

23

Attributi di classe e attributi di istanza

- Le variabili di classe sono di solito (ma non solo) aggiunte alla classe mediante assegnamenti all'esterno delle funzioni.
- Le variabili di istanza possono essere aggiunte all'istanza mediante assegnamenti effettuati all'interno di funzioni che hanno self tra gli argomenti.

24

Attributi di classe e attributi di istanza

```
class myClass:
      a=3
      def method(self):
            self.a=4


x=myClass()
print(x.a)
x.method()
print(x.a)
y=myClass()
print(y.a)
print(myClass.a)

x.b=10
print(x.b)
```

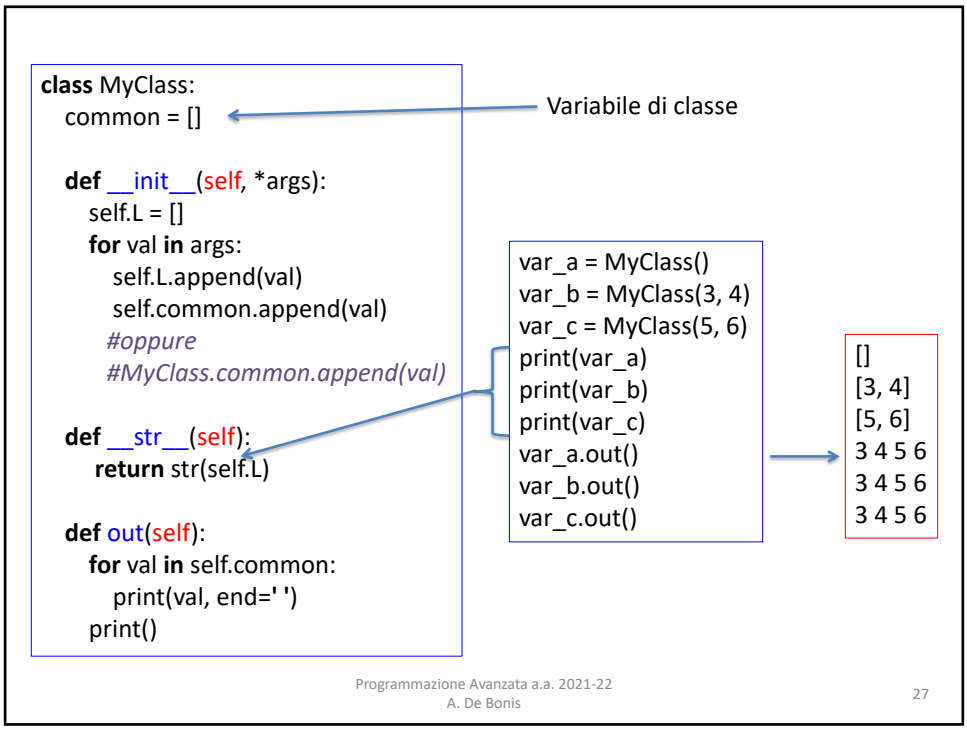
```
3
4
3
3
10
```


25

Costruttori in Python

- Nelle classi Python ci può essere un solo costruttore chiamato **__init__**
- Per simulare differenti costruttori si possono usare
  - parametri inizializzati di default
  - numero di parametri variabile
  - parametri keyword
- Se __init__ non è fornito né dalla classe né da nessuna delle classi più in alto nella gerarchia delle classi allora vengono create istanze vuote

26



27

Metodi di una classe

- Tutti i **metodi di istanza** della classe hanno come primo parametro **self** che rappresenta l'istanza dell'oggetto su cui è chiamato il metodo
  - self è un riferimento esplicito all'oggetto su cui andare ad operare
    - Simile a this in Java

a istanza di una classe A
func metodo della classe A
a.func(b) è convertito in A.func(a,b)

A è considerato un namespace

28

Assegnamenti dinamici

- Data un'istanza della classe è possibile aggiungere e/o rimuovere dinamicamente membri all'istanza stessa
- Possiamo aggiungere anche variabili di classe

```
def add_var():
    var_a.nuovo = 3
    print('nuovo attributo: ', var_a.nuovo)
    try:
       print('nuovo attributo: ', var_b.nuovo)
    except Exception as e: print(e)

    MyClass.nuovo = 0
    try:
       print('nuovo attributo: ', var_b.nuovo)
    except Exception as e: print(e)
```

```
nuovo attributo:  3
'MyClass' object has no attribute 'nuovo'
nuovo attributo:  0
```

Per cancellare un attributo si usa **del**
del var_a.nuovo

29

Ulteriori esempi

```
x = MyClass()
x.counter = 1
while x.counter < 10:
   x.counter = x.counter * 2
print(x.counter)
del x.counter
```

16

la classe MyClass ha il metodo out

```
MyClass.common = []
x = MyClass([1,'x','das'])
xout = x.out
xout()
```

[1, 'x', 'das']

30

Altro sui metodi

- I metodi di istanza di una classe possono chiamare altri metodi di istanza della stessa classe utilizzando **self**

```
class Bag:
   def __init__(self):
      self.data = []

   def add(self, x):
      self.data.append(x)

   def addtwice(self, x):
      self.add(x)
      self.add(x)
```

- I metodi di una classe possono essere definiti fuori la classe stessa

```
def f1(self, x, y):
   return min(x, x+y)

class C:
   f = f1

   def g(self):
      return 'Ciao Mondo!'

c = C()
print(c.f(2,3))
```