ZerotoHero_1

February 19, 2020

[6]:
```
print("prima riga", end="")
print(" e questa è ancora la prima riga")
```

```
prima riga e questa è ancora la prima riga
```

0.1 Variabili

In python non serve dichiarare le variabili. Ogni volta che si assegna un valore ad una etichetta viene automaticamente creata una variabile del tipo adeguato.

[13]:
```
numero = 3
decimale = 7.5
frase = "ciao mondo"
print(type(numero).__name__,numero) #doppio underscore si chiama Dunder
print(type(decimale).__name__,decimale)
print(type(frase).__name__,frase)
```

```
int 3
float 7.5
str ciao mondo
```

0.2 Gestione dinamica dei numeri INT:

gli interi di python sono **infiniti**, tanta RAM avete, tanto grande può essere il numero! Non è la stessa cosa per il Float, abbiamo una gestione dinamica a livello di bit per gli INT

0.3 Riutilizzo delle variabili (sconsigliato):

Se assegniamo un nuovo valore ad una variabile questa lo mantiene da quel punto in poi!

0.4 in Python tutto è un oggetto

[ ]:
```

```

[16]:
```
a = int("10")
print(a.bit_length())
a = int("65000")
print(a.bit_length())
```

```
4
16
```

[19]:
```
b = "facciamo una prova" #anche le stringhe sono oggetti
print("nella stringa ci sono",b.count("a"), "a")
print(b.upper())
print(b.title())
```

```
nella stringa ci sono 4 a
FACCIAMO UNA PROVA
Facciamo Una Prova
```

0.5 Formattazione delle stringhe

[26]:
```
modo1 = "numero di %s, %d" % ("prova1", 1)
modo2 = "numero di {},{}".format("prova2", 2)
modo3 = "numero di {1}, {0}".format(3, "prova3")
numero, testo = 4, "prova4"
modo4 = f"numero di {testo}, {numero}"

print(modo1)
print(modo2)
print(modo3)
print(modo4)
```

```
numero di prova1, 1
numero di prova2,2
numero di prova3, 3
numero di prova4, 4
```

1 Tuple: oggetti *immutabili*

[27]:
```
var_tupla = (1, 2, "terzo")
print(var_tupla)
```

```
(1, 2, 'terzo')
```

2 Liste

[29]:
```
var_lista = ["a", "b", 3]
print(var_lista)
var_lista[1] = "secondo"
print(var_lista)
var_lista.append("andrea")
print(var_lista)
```

```
['a', 'b', 3]
['a', 'secondo', 3]
['a', 'secondo', 3, 'andrea']
```

3 Dizionari

[30]:
```
var_voti = {"luca": 7, "mario": 5, "francesca": 9}
print(var_voti)
var_voti["luca"] = 8
print(var_voti)
```

```
{'luca': 7, 'mario': 5, 'francesca': 9}
{'luca': 8, 'mario': 5, 'francesca': 9}
```

4 Set: sono insiemi

[34]:
```
primo = {"luigi", "mario", "francesca"}
secondo = {"andrea", "luigi", "silvia"}
print(primo)
print(secondo)
print(primo | secondo)
print(primo - secondo)
print(primo & secondo)
```

```
{'mario', 'luigi', 'francesca'}
{'luigi', 'andrea', 'silvia'}
{'luigi', 'andrea', 'francesca', 'mario', 'silvia'}
{'mario', 'francesca'}
{'luigi'}
```

4.1 Flussi

[ ]:
```
numero_stringa = input("Inserisci un numero: ")
numero = int(numero_stringa)

if(numero > 10):
    print("il numero è maggiore di 10")
elif(numero < 10):
    print("il numero è minore di 10")
else:
    print("il numero scelto è 10")
```

[18]:
```
print("vero" if 20 > 5 else "falso")
print("andrea" if "andrea" else "falso") #la stringa non è vuota
```

```
vero
andrea
```

stessa cosa per verificare se una lista o tupla sono vuote: eseguo if sulla variabile e se il risultato è True significa che la stessa non è vuota!

[51]:
```
my_str = ""
print("VERO" if my_str else "FALSO")
```

```
FALSO
```

[58]:
```
my_str = "casa"
print("VERO" if my_str else "FALSO")
```

```
VERO
```

[59]:
```
if my_str:
    print("vero")
else:
    print("falso")
print(my_str)
```

```
vero
casa
```

[62]:
```
lista_test = ["margherita", "tulipano", "orchidea", "rosa", "papavero"]
for fiore in lista_test:
    if fiore == "rosa":
        print(f"ho trovato {fiore}")
        break
    else:
        print(fiore)
```

```
margherita
tulipano
orchidea
ho trovato rosa
papavero
```

[64]:
```
for i in range(10):
    if i % 2 == 0:
        print(f"{i} è numero pari")
        continue
    print(f"{i} è numero dispari")
```

```
0 è numero pari
1 è numero dispari
2 è numero pari
3 è numero dispari
4 è numero pari
5 è numero dispari
6 è numero pari
```

```
7 è numero dispari
8 è numero pari
9 è numero dispari
```

[68]:
```python
cicli = 3
for i in range(cicli):
    if i > 5:
        break
    else:
        print(f"ciclo {i}")
else:
    print("ho eseguito tutti i cicli")
```

```
ciclo 0
ciclo 1
ciclo 2
ho eseguito tutti i cicli
```

[3]:
```python
cicli = 3000
for i in range(cicli):
    if i > 5:
        break
    else:
        print(f"ciclo {i}")
else:
    print("ho eseguito tutti i cicli") #passa ad else se esegue TUTTI i cicli!
```

```
ciclo 0
ciclo 1
ciclo 2
ciclo 3
ciclo 4
ciclo 5
```

5 Funzioni

[5]:
```python
def somma(a,b):
    """Somma due numeri"""
    return a + b

print(somma(1,2))
```

```
3
```

[6]:
```python
def somma(a,b):
    print(a+b)
```

```
somma(1,4)
```

```
5
```

[13]:
```
def somma(a,b):
    return a+b, 11, 13

print(somma(5,5))
```

```
(10, 11, 13)
```

[17]:
```
""" attenzione l'assegnazione delle variabili avviene per RIFERIMENTI!"""
a = 1
print(id(a), "id di a")
b = a
print(id(b), "id di b")
print(a is b)
a = 2
print(id(a), "id di a dopo cambio valore")
print(id(b), "id di b è rimasto invariato ma ora diverso da id di a")
print(a is b)
a = 1
print(id(a), "id di a con vecchio valore, stesso id dell'inizio")
print(a is b)
```

```
9739840 id di a
9739840 id di b
True
9739872 id di a dopo cambio valore
9739840 id di b è rimasto invariato ma ora diverso da id di a
False
9739840 id di a con vecchio valore, stesso id dell'inizio
True
```

6 Classi

[35]:
```
class automobile():
    """rappresenta il concetto di auto"""

    def __init__(self, colore, alimentazione):
        self.colore = colore
        self.alimentazione = alimentazione
        self.accesa = False

    def accendi(self):
        self.accesa = True
```

```
    def spegni(self):
        self.accesa = False
```

[36]:
```
macchina1 = automobile("verde", "GPL")
macchina2 = automobile("rossa", "Benzina")
print(macchina1.colore)
print(macchina1.alimentazione)
```

```
verde
GPL
```

[37]:
```
macchina1.colore = "nera"
print(macchina1.colore)
print(macchina2.colore)
```

```
nera
rossa
```

[41]:
```
macchina1.accendi()
print("è accesa la macchina 1? ", macchina1.accesa)
print("è accesa la macchina 2? ",macchina2.accesa)
```

```
è accesa la macchina 1?  True
è accesa la macchina 2?  False
```

6.1 Metodi e attributi privati

[42]:
```
class automobile():

    def __init__(self):
        self.__accesa = False

    def accendi(self):
        self.__accesa = True

    def spegni(slef):
        self.__accesa = False

    def stato(self):
        return self.__accesa
```

[47]:
```
toyota = automobile()
print(toyota.stato())
print(toyota._automobile__accesa) #accesso consapevole all'attributo privato
print(toyota.__accesa)
```

```
False
False
```

```
      ␣
↪-------------------------------------------------------------------------

       AttributeError                            Traceback (most recent call␣
↪last)

       <ipython-input-47-34a1002dca5e> in <module>
         2 print(toyota.stato())
         3 print(toyota._automobile__accesa)
   ----> 4 print(toyota.__accesa)


       AttributeError: 'automobile' object has no attribute '__accesa'
```

[ ]: