Esercitazione 4

November 11, 2019

1 Lezione 4

1.1 Funzione print()

Gli **argomenti** della funzione **print**() sono i seguenti:

print(value 1, . . . , sep=' ', end='', file=sys.stdout, flush=False)

La funzione **print**() può stampare un numero arbitrario di valori (**"value 1, value 2, . . . "**), che sono separati da una virgola.

Questi valori, una volta stampati, sono separati da uno spazio bianco:

[1]:
```
print("Hello","World")
```

```
Hello World
```

1.1.1 Esempio 1:

[2]:
```
a = 3.14
```

[3]:
```
print("a = ", a)
```

```
a =  3.14
```

1.1.2 Esempio 2:

[4]:
```
a = 3.14
```

[5]:
```
print("a = \n", a)
```

```
a =
 3.14
```

Il simbolo "" è sempre stampato

1.1.3 Esempio 3:

[6]:
```
print("a","b")
```

```
a b
```

1.1.4 Esempio 4:

[7]:
```
print("a","b",sep="")
```

```
ab
```

1.1.5 Esempio 5:

[8]:
```
print("192","168","1","1",sep=".")
```

```
192.168.1.1
```

1.1.6 Esempio 6:

[9]:
```
print("a","b",sep=":-)")
```

```
a:-)b
```

Alla fine di qualsiasi output stampato da una print viene sempre aggiunta una **newline**:

[10]:
```
print("Hello")
print("World")
```

```
Hello
World
```

Per cambiare questo comportamento possiamo assegnare una stringa arbitraria al parametro con keyword **end**. Questa stringa verrà usata per chiudere l'output stampato dalla print:

[11]:
```
i = 10
j = 12
print(i,end=" :-) ")
print(j,end=" :-) ")
```

L'output della funzione print è generalmente inviato verso lo standard stream output (sys.stdout). Facendo riferimento al parametro con keyword **file** è possibile inviare l'output verso flussi alternativi: per esempio verso sys.stderr o un file:

[12]:
```
fh = open("data.txt","w")
print("42 è la risposta, ma qual è la domanda?", file=fh)
fh.close()
```

L'output viene inviato nel file data.txt. E' possibile redirigere l'output verso lo standard error channel:

[13]:
```
import sys
print("Error: 42", file=sys.stderr)
```

```
Error: 42
```

Il parametro flush=False viene utilizzato soprattutto quando lo stream di output è un file. In questo caso il buffer viene forzatamente pulito e inviato totalmente al file. Di default è False.

1.2 Input da tastiera

[14]:
```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("So, you are already " + age + " years old, " + name + "!")
```

```
What's your name? Pippo
Nice to meet you Pippo!
Your age? 42
So, you are already 42 years old, Pippo!
```

La funzione input() prende come argomento opzionale una stringa. La funzione stampa l'argomento e si ferma attendendo che l'utente digiti qualcosa. Quando l'utente preme Invio la funzione restituisce una stringa che contiene i caratteri scritti dall'utente.

input() non va a capo e non lascia spazi dopo aver stampato il prompt, quindi è necessario inserire uno spazio alla fine della stringa per separare la domanda dalla risposta.

2 Esercizio 1

Scrivere un programma nell'IDLE python e chiamalo Esercizio1.py. L'Input/Output richiesto è:

Come ti chiami? Tuo nome
Qual è il tuo cognome? Tuo cognome
Benvenuto Tuo nome Tuo cognome

2.0.1 Esercizio 2

Scrivi il seguente programma nell'IDLE python e chiamalo Esercizio2.py. Esegui il programma assegnando 12 ad a:

a = input("Scrivi un numero")
print("Il doppio di", a, "e"', 2 * a)
print("Il quadrato di", a, "e"', a ** 2)

---

[15]:
```
Scrivi un numero 12
Il doppio di 12 e' 1212
Traceback (most recent call last):
  File "/Users/darthrooster/Documents/Esercizio2.py", line 3, in <module>
    print("Il quadrato di", a, "e'", a ** 2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

```
  File "<ipython-input-15-625543fbbde9>", line 1
    Scrivi un numero 12
               ^
SyntaxError: invalid syntax
```

La prima print() scrive che il doppio di 12 è 1212, mentre la seconda addirittura manda in crash il programma provocando un TypeError!

Questo comportamento si spiega se ripensiamo al fatto che ogni variabile ha un tipo di dato associato ad essa: nella prima istruzione assegnamo ad a non il numero 12 ma la stringa "12" (cioè una stringa che contiene la sequenza di caratteri 12).

Nella seconda istruzione, quando Python trova l'espressione 2 * a, esegue non la moltiplicazione di due numeri, ma la moltiplicazione di un numero per una stringa (ottenendo la concatenazione "1212").

Nella terza istruzione si ottiene infine un errore perchè non è possibile fare il quadrato di una stringa.

Per ovviare a queste situazioni esistono delle apposite funzioni di conversione: Le funzioni int() e float() prendono un argomento di qualsiasi tipo e restituiscono un numero **intero** o **float** ottenuto dal dato di partenza.

[16]:
```
age = "42"
```

[17]:
```
print(age)
```

```
42
```

[18]:
```
print(age + 1)
```

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

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-18-d9cf328f2a24> in <module>
   ----> 1 print(age + 1)


       TypeError: can only concatenate str (not "int") to str
```

[19]:
```
age = int(age)
age = float(age)
```

[20]:
```
print(age + 1)
```

```
43.0
```

2.0.2 Esercizio 3

Modificare l'Esercizio 2 al fine di renderlo corretto.

2.0.3 Esercizio 4

In un file Operazioni.py scrivere un programma che chieda in input due numeri (float) e stampi la loro somma e il loro prodotto.

2.0.4 Esercizio 5

Scrivere un programma che chieda in input di immettere la temperatura in gradi Celsius e stampi il valore della stessa temperatura in gradi Fahrenheit e Kelvin. Le formule di conversione sono: řF = řC * 1.8 + 32 K = řC + 273.15

2.1 Operazioni su Stringhe

[21]:
```
stringa = "Io sono una stringa"
print(stringa)
type(stringa)
```

```
Io sono una stringa
```

[21]:
```
str
```

[22]:
```
stringa1 = 'Io sono una stringa'
print(stringa)
type(stringa)
```

```
Io sono una stringa
```

[22]:
```
str
```

Possiamo scrivere espressioni anche con le stringhe: infatti su di esse sono possibili due operazioni:

1. **addizione tra stringhe** (che produce la loro concatenazione)
2. **moltiplicazione per un intero** (che concatena più volte la stessa stringa)

[23]:
```
s = "Buona"
t = "Lezione"
st = s + " " + t
print(st)
```

```
Buona Lezione
```

[24]:
```
s = "Buona"
st = s * 2
print(st)
```

```
BuonaBuona
```

2.2 Esercizio 6

Scrivere un programma che stampa la seguente figura: Vedi Allegato 1

2.3 Esercizio 7

Scrivere un programma che definisce un orologio binario: Vedi Allegato 2

2.4 Esercizio 8

Rappresentare il barchart definito in figura: Vedi Allegato 3

2.5 Esercizio 9

Consideriamo la seguente espressione:

E1:

$$x_0 = 1 x_{n+1} = \frac{x_n}{2} + \frac{z}{2x_n}$$

E2:

$$x_0 = 0.5 x_{n+1} = 0.5 x_n (3 - z x_n^2)$$

Scomporre in step le espressioni ed eseguire n = 5 passi per z=7.