Python for Technologies #1

I primi passi

Alvaro Gaiotti — alvaro.gaiotti@randstad.it

Indice

Il primo programma

Tradizionalmente, il primo programma che si scrive è il classico Hello World, quindi riportate il seguente codice nel pannello editor, all'interno del file `main.py`:

> **Suggerimento**
>
> Non copiate e incollate, ma scrivete ed eseguite sempre!

```python
"""Il nostro primo programma in Python"""
print("Hello, World!")  # Stampa una saluto al mondo
```

Eseguiamo poi il nostro script digitando nella shell il seguente comando e premendo `Invio`:

```bash
python main.py
```

La nostra shell stamperà

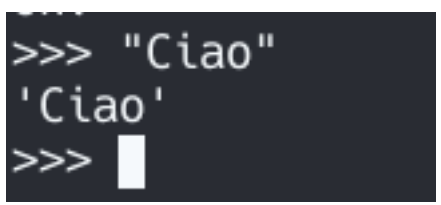
```
Hello, World!
```

La REPL

Per iniziare ad utilizzare la REPL Python, digitate `python` nella shell e premete `Invio`.

In questa modalità l'interprete Python legge riga per riga quello che digitate e, una volta premuto `Invio`, lo valuta, restituendone il risultato.

Provate a digitare `"Ciao"`, il risultato sarà il seguente:



Proviamo invece a digitare `print("Ciao")` e osserviamo il risultato: notate delle differenze?

Proviamo infine a digitare:

```
"""Questa è una stringa
su più righe"""
```

Le operazioni aritmetiche

Python ha le classiche abilità di esecuzione di operazioni aritmetiche che hanno tutti i linguaggi di programmazione:

- Addizione: `+`
- Sottrazione: `-`
- Moltiplicazione: `*`
- Divisione: `/`

> **Suggerimento**
>
> Provate a sperimentare con alcuni calcoli

Python può valutare intere espressioni, formate da numeri, variabili, operatori aritmetici, parentesi tonde e funzioni:

- Si parte dalle espressioni contenute nelle parentesi più interne, passando via via a quelle più esterne
- Si segue l'ordine di precedenza degli operatori
- A parità di ordine, si valutano le operazioni da sinistra a destra

```python
6 + 3 * 12    # 42
(6 + 3) * 12  # 108
```

Esistono anche operatori per l'elevamento a potenza (`**`), la divisione intera (`//`) e il modulo o resto (`%`)

Variabili e assegnazione

Possiamo pensare ad una variabile come ad una scatola, con un'etichetta dal nome univoco, con qualcosa al suo interno.

Una variabile è un nome che associamo ad una posizione nella memoria del computer, per un più comodo utilizzo.

Per inserire un qualcosa all'interno della variabile/scatola si usa l'operatore "=" :

```python
a = 10          # Inserisci 10 nella scatola `a`
caramelle = 6   # Inserisci 6 nella scatola `caramelle`
```

Una variabile viene chiamata in questo modo proprio perché, generalmente, durante un programma essa «varia» il proprio contenuto, e può accomodare cose differenti in differenti momenti, al prezzo di dimenticare quelle precedentemente assegnate:

```python
a = 3
a = 5   # Il valore 3 è andato perso ora
```

Assegnare una variabile permette di salvarla nel «vocabolario» di Python per la sessione corrente, che si occuperà, una volta incontrato il nome della variabile in un espressione, di sostituirla con il suo valore.

```python
a = 5
b = a + 3   # Equivalente a b = 5 + 3
```

Possiamo combinare l'assegnazione con un operatore aritmetico:

```python
a = 3
a += 5    # a = a + 5
a -= 2    # a = a - 2
a /= 3    # a = a / 3
a *= 2    # a = a * 2
```

Funzioni

Una funzione è qualcosa che riceve dei dati in ingresso e ne restituisce in uscita.

Python ci fornisce alcune funzioni predefinite, dette *funzioni built-in*. Ad esempio:

- `max(x1, ..., xn)`: calcola il massimo di una lista di valori
- `min(y1, ..., yn)`: calcola il minimo di una lista di valori
- `abs(n)`: calcola il valore assoluto di uno numero

Suggerimento

Provate a sperimentare con queste funzioni con diversi input, es.

```python
max(1,2,3,4)   # 4
abs(-3)   # 3
min("ditta", "hotel", "Damiano", "forno", "a")    # ?
```

Bonus points per chi riesce a scoprire il perché dell'output dell'ultima riga: cercate `ASCII` su Google e provate a capirlo.

I tipi

Possiamo pensare ad un tipo di dato come ad una classe di valori, definita da:

- Nome («intero»)
- Valori ammissibili (`x ≥ 0`)
- Operazioni ammissibili (addizione, sottrazione, ...)
- Dimensione (quanta memoria occupa)
- Codifica (come viene rappresentato, es. `0b00001101 = 13`)

Quando si tratta di tipi, la funzione `type()` è nostra amica:

```python
type(42)   # <class "int">
```

I tipi «int» e «float»

Per il momento, ci interessano i tipi numerici utilizzati da Python, ovvero:

- `int`
- `float`

```python
type(3)      # <class "int">
type(3 / 2)  # <class "float">
type(3 // 2) # <class "int"> Divisione intera!
type(5 % 2)  # <class "int">
```

Esercizio

Provate a calcolare la velocità oraria e in metri al secondo di un corridore che percorre 4.1`km` in 21 minuti e 34 secondi. Salvate il risultato in un file nominato `velocita.py`. Lo script deve stampare a schermo le velocità arrontondate a due cifre decimali.

Potete usare la funzione `print(contenuto1, ..., contenutoN)` per stampare a schermo il risultato dei vostri calcoli e la funzione `round(da_arrotondare, n_cifre_decimali)` per arrotondare un numero al numero di cifre decimali richiesto.

Soluzione Suggerita

Python

```python
"""Calcolo della velocità in km/h e m/s"""
# Dati iniziali
minuti = 21
secondi = 34
km = 4.1
# Elaborazione
secondi_totali = minuti * 60 + secondi

```

```
9   # Velocità in m/s
10  metri = km * 1000
11  m_s = metri / secondi_totali
12  # Velocità in km/h
13  ore_totali = secondi_totali / 3600
14  km_h = km / ore_totali
15  # Output
16  print("Velocità:", round(m_s, 2), "m/s")
17  print("Velocità:", round(km_h, 2), "km/h")
```