Dall'Algoritmo al Codice

Il Pensiero Computazionale

Percorso Formativo per i Docenti della Scuola Secondaria di II Grado

Lezione 1 - Parte 1

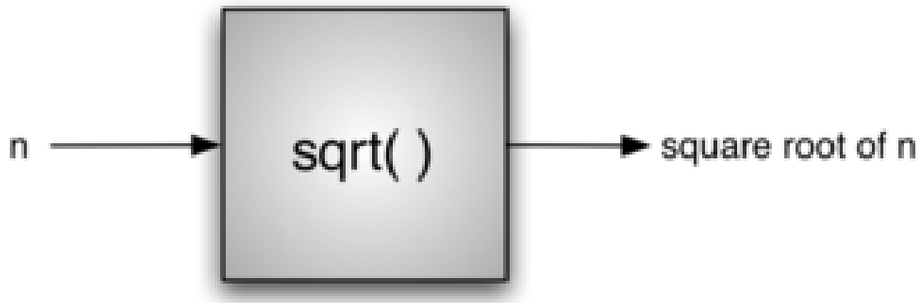
Stefano Forti and Davide Neri

Sommario - Introduzione a Python

- Introduzione alla programmazione
- Variabili
- Tipi di dato (`int`, `float`, `bool`, `str`)
- Liste
- Input Output I/O (`input()`, `print()`)
- Costrutti iterativi (`while`, `for`)
- Costrutti condizionali (`if`, `if-else`, `if-elif-else`)

La Programmazione

In generale, possiamo immaginare un programma come una scatola magica che, presi dei **dati in ingresso** (*input*) produce, seguendo un algoritmo, un **risultato in uscita** (*output*).



La funzione `math.sqrt(n)` restituisce la radice quadrata di `n`:

```
import math           # importiamo la libreria matematica

n = 16                # assegniamo ad n il valore 16
print(math.sqrt(n)) # il risultato è...?
```

Python: da 0 a 100 (o quasi) in 2 ore... ⏳

import this

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Variabili

I linguaggi di programmazione usano dei barattoli etichettati per contenere dati.

L'operazione fondamentale sulle variabili è l'**assegnamento** (`=`):

```
anni = 27
print(anni) # il risultato è...?
anni = anni + 1
print(anni) # il risultato è...?
```

Tipi di Dato

In Python i dati possono essere di tre tipi principali, ciascuno con le sue operazioni:

- `int` o `float` (numeri interi o decimali),
- `bool` (valori booleani, vero/falso),
- `str` (stringhe di testo, sempre tra virgolette `"ciao"` o apici singoli `'ciao'`).

Per identificare il tipo di una variabile `v` basta usare la funzione `print(type(v))`.

```
n = 16
print(type(n))

b = False
print(type(b))

s = 'ciao'
print(type(s))
```

Numeri (`int` o `float`)

Proviamo queste operazioni in Python interattivo! 😄

```
print(2+3*4)
print((2+3)*4)
print(2**10)
print(7/3)
print(7//3)
print(7%3)
```

A quali operazioni corrispondono `**`, `//` e `%`?

Booleani (`bool`)

I tipi booleani sono equipaggiati con i tre principali operatori `and` ($\wedge$), `or` ($\vee$), `not` ($\neg$) definiti secondo l'algebra di Boole:

- `a1 and ... and ak` è `True` se e solo se `a1, ..., ak` sono tutte `True`,
- `a1 or ... or ak` è `True` se e solo se almeno una tra `a1, ..., ak` è `True`,
- `not(a)` è `True` se `a` è `False` e, viceversa, è `False` se `a` è `True`.

```
print(5==10)
print(10 > 5)
print((5 >= 1) and (5 <= 10))

a = 30 > 8
b = 6 != 4          # != corrisponde a 'diverso da', 'non è uguale'
print(a and b)
print(a or b)
print(a and not(b))
```

Stringhe (`str`)

Sono dichiarate come una sequenza di **caratteri** fra apici singoli (`'foo'`) o doppi (`"foo"`).

```
s = "Python"

# Accesso al carattere i-esimo
print(s[0])   # elemento in posizione 0 (il primo)
print(s[5])   # elemento in posizione 5 (il sesto)

# Slicing
print(s[0:2])  # sottostringa con elementi da 0 (incluso) a 2 (escluso) -> 'Py'
print(s[:2])   # dall'inizio all'elemento con indice 2 (escluso)        -> 'Py'
print(s[4:])   # dall'elemento con indice 4 (incluso) alla fine         -> 'on'
```

Stringhe (cont.)

```
# concatenazione e ripetizione
print('Py' + 'thon')
print('Py' * 2)
print('Ba' + 'na' * 2)

# Presenza di una stringa in un'altra stringa
s = 'Python'
print('Py' in s)
print( 'x' not in s)

# Lunghezza di una stringa
s = 'Precipitevolissimevolmente'
print(len(s))
```

Stringhe (cont.)

```
# concatenazione e ripetizione
print('Py' + 'thon')     # Concatena la stringa "Py" con "thon"
print('Py' * 2)          # Ripeti per due volte la stringa "Py"
print('Ba' + 'na' * 2)   # Concatena "ba" con una doppia ripetizione di "na"

# Presenza di una stringa in un'altra tringa
s = 'Python'
print( 'x' not in s)    # "not in" esegue l'operazione inversa
print('Py' in s)        # controlla se 'Py' è contenuto in s

# Lunghezza di una stringa
s = 'Precipitevolissimevolmente'
print(len(s))            # stampa la lunghezza della stringa -> 6
```

Liste

Una lista è una collezione ordinata di zero o più elementi (⚠️ anche eterogenei per tipo!).

```
myList = [1, 3, True, 6.5]
print(myList)

# lunghezza lista e contenuto
print(len(myList))
print(3 in myList)

# accedere un elemento
print(myList[0])
print(myList[2])

# concatenare liste
myList2 = [6, 95.2, 'ciao']
myList3 = myList + myList2
print(myList3)
```

Gli elementi di una lista di `n` elementi sono indicizzati da `0` a `n-1`.

Funzioni su Liste

```
myList = [1024, 3, 7, 6.5]

# aggiungere un elemento
myList.append(1)

#indice di un elmento
print(myList.index(7))

# rimuovere un elemento
myList.remove(3)
myList.pop(1)

# ordinare
myList.sort()
myList.reverse()

# Numero di elementi contenuti nella lista
print(len(myList))

print(myList)
```

Input e Output

- La funzione di **output** in Python è, come abbiamo visto, la `print()`.
- La funzione di **input** in Python è invece la `input(msg)`. Prende come parametro opzionale `msg` un messaggio di cortesia destinato all'utente. Il risultato della `input(msg)` è una stringa che contiene quanto digitato dall'utente prima di premere invio.

```
nome = input('Inserisci il tuo nome: ')
anni = int(input('Inserisci la tua età: ')) # int converte in intero la stringa

print("Ciao", nome, "di anni", anni)
```

Strutture di Controllo

Il **costrutto iterativo** (ovvero che ripete un dato comando) e quello **condizionale** (che sceglie se eseguire un dato comando) sono alla base di tutti i linguaggi di programmazione.

Costrutti Iterativi (Cicli)

Costrutto Iterativo Indefinito (`while`)

```
counter = 0
while counter < 5: # ripete finché la condizione (counter < 5) è vera
    print("Hello, World!")
    counter = counter + 1
```

Costrutto Iterativo Definito (`for`)

```
for i in range(5):
    print("Hello, World!")

for elemento in ['ciao', 'come', 'stai', '?']:
    print(elemento)
```

Costrutto Condizionale

Costrutto Condizionale Semplice (`if`)

```
numero = int(input('Inserire il dividendo:'))

if divisore != 0:
   risultato = dividendo / divisore
   print(dividendo, 'diviso', divisore, 'è uguale a', 'risultato')
```

Costrutto Condizionale a Due Rami (`if-else`)

```
dividendo = int(input('Inserire il dividendo:'))
divisore = int(input('Inserire il divisore:'))

if divisore != 0:
   risultato = dividendo / divisore
   print(dividendo, 'diviso', divisore, 'è uguale a', 'risultato')
else:
   print('Impossibile dividere per 0!')
```

Costrutto Condizionale

Costrutto Condizionale (`switch` -like)

```
risultato = int(input('Inserire il voto del compito:'))

if risultato >= 90:
    print('A')
elif risultato >=80:
    print('B')
elif risultato >= 70:
    print('C')
elif risultato >= 60:
    print('D')
else:
    print('F')
```