Introduzione a Python

Basi di informatica, gestione dati e statistica (BIGDS)

Python

- Python è uno dei *più diffusi* linguaggi di programmazione e ha una *curva di apprendimento* pressoché piatta. Enorme disponibilità di *librerie*: NumPy, Pandas, scikit-learn...
- Funzionale e orientato agli oggetti
- Consolidato nel tempo
- Semplice e versatile

Download

https://www.python.org/downloads/

Dipendenza dal SO

IDE – Integrated Development Enviroment

Windows (multipiattaforma)

- **pycharm**: https://www.jetbrains.com/pycharm/
- **wing**: http://wingware.com/downloads
- **spyder**: https://www.spyder-ide.org/

web-based tools

- **codeboard**: codeboard.io
- **replit**: https://replit.com/
- **colab**: https://colab.research.google.com/

Setup e primo programma

- Preferenze del IDE
- Creazione di un nuovo progetto

Primo programma: `print("Hello world")`

dove `print` rappresenta una **funzione** predefinita (*built-in*)

- Ogni istruzione termina con un *a capo*
- Importanza della *indentazione*

I commenti

```
# Il mio primo programma
print("Hello world")
```

Variabili

Contenitori che posso *accogliere* al loro interno dei dati (*tipo e nome*)

Esempio:

```
print("Matteo insegna Informatica")
print("Federica insegna Matematica")
```

con le **variabili** e operatore di **assegnamento**:

```
char_nome = "Matteo"
char_materia = "Informatica"
print(char_nome + " insegna " + char_materia)
char_nome = "Federica"
char_materia = "Matematica"
print(char_nome + " insegna " + char_materia)
```

Essenziali

- assegnazione `=` (da non confondere con `==`)
- operatori aritmetici `+ - * / %`
- operatori logici `and or not`
- NO definizione tipo delle variabili
- indentatura
- usare una variabile ha l'effetto di impostare una `etichetta (o nome)` che contiene un `riferimento` ad un `oggetto`
- le etichette *non hanno un tipo* - gli oggetti *hanno un tipo*

Essenziali

`x = 0`

- viene creato un intero e memorizzato
- viene creata un etichetta di nome `x`
- `x` contiene un riferimento alla locazione di memoria che contiene il valore `0`

Essenziali

Con `x = x + 1` cosa succede?

- viene recuperato il riferimento (indirizzo) contenuto in `x`
- il valore contenuto nella corrispondende locazione di memoria viene recuperato (`0`)
- viene eseguito in calcolo `0 + 1`
- viene creato un nuovo intero `1` (oggetto)
- viene aggiornato il riferimento di `x` , ora con l'indirizzo dell'oggetto `1`
- il vecchio oggetto `0` viene perso e quella locazione di memoria resa disponibile

Differenti tipo di dato

La definizione degli *oggetti* (e delle variabili) ha come conseguenza l'assegnazione di uno specifico spazio in *memoria centrale*

Stringa `"Matteo insegna Informatica"`

Numeri

```
anni = 49
```

Boolean `True` o `False`

Il **tipo** definisce l'insieme di operazioni che possono essere svolte su quella specifica variabile

Le stringhe

```
frase = "Matteo insegna Informatica"
print(frase + " e Statistica")
```

Le stringhe e funzioni

```
frase = "Matteo insegna Informatica"
print(frase.lower())
```

```
frase = "Matteo insegna Informatica"
print(frase.upper())
```

```
frase = "Matteo insegna Informatica"
print(frase.replace("Matteo", "Luca"))
```

Le stringhe e funzioni

```
frase = "Matteo insegna Informatica"
print(len(frase))
```

```
frase = "Matteo insegna Informatica"
print(frase[0])
```

Python usa lo 0 come indice del primo elemento di una stringa

Numeri

```
print(1)
print(1.234)
print(-1)
print(-1 + 1)
print(4 / 2)
print(4 % 3)
```

```
num = 3
print(num)
```

Numeri e stringhe

```
frase = "Matteo insegna Informatica"
anni = 49
print(frase + " e ha " + str(anni) + " anni")
```

Numeri e funzioni

```
num = -3
print(abs(num))
```

```
num = 3
print(pow(num, 2))
```

```
num = 3.3
print(round(num))
```

```
from math import *
num = 3.3
print(sqrt(num))
```

Input da utente

`input()`

```
nome = input("Inserisci il tuo nome: ")
print("Ciao " + nome + "!")
```

La variabile `nome` conterrà la **stringa** inserita dall'utente

Calcolatrice

```
num_1 = input("Inserisci un numero: ")
num_2 = input("Inserisci un altro numero: ")
risultato = int(num_1) + int(num_2)
print(risultato)
```

... int() ???

... ma attenzione!

```
num_1 = input("Inserisci un numero: ")
num_2 = input("Inserisci un altro numero: ")
risultato = float(num_1) + float(num_2)
print("Risultato = ", risultato)
```

```
num_1 = input("Inserisci un numero: ")
num_2 = input("Inserisci un altro numero: ")
risultato = float(num_1) / float(num_2)
#format_float = "{:.2f}".format(risultato)
#print(format_float)
print("{:.2f}".format(risultato))
```

Istruzioni condizionali

Istruzione **if**

```
if condizione:
     istruzione/i
```

```
i = 1
if i < 10:
    print(str(i) + " e' minore di 10")
```

- La valutazione di un'espressione restituisce `True` o `False` (1 o 0)

Istruzioni condizionali

Ramo **else**

```
if condizione:
    istruzione/i
else:
    istruzione/i
```

```
i = 100
if i < 10:
    print(str(i) + " e' minore di 10")
else:
    print(str(i) + " e' maggiore di 10")
```

Istruzioni condizionali

if **annidati**

```
i = -1
if i < 10:
    if i < 0:
        print(str(i) + " e' negativo")
    else:
        print(str(i) + " e' minore di 10")
else:
    print(str(i) + " e' maggiore di 10")
```

Istruzioni condizionali

if **annidati**

```
i = 10
if i < 10:
    if i < 0:
        print(str(i) + " e' negativo")
    else:
        print(str(i) + " e' minore di 10")
else:
    if i == 10:
        print(str(i) + " e' uguale a 10")
    else:
        print(str(i) + " e' maggiore di 10")
```

Istruzioni condizionali

Esercizio

Scrivere un programma che permetta di valutare se un numero intero (letto da tastiera) è pari o dispari

```
print("Inserisci un numero: ")
num = int(input())

if num % 2 == 0:
    print("Il numero inserito e' pari")
else:
    print("Il numero inserito e' dispari")
```

Istruzioni condizionali

elif

```
if condizione:
    istruzione/i
elif condizione:
    istruzione/i
else:
    istruzione/i
```

Esercizio

Creare un programma che presi in ingresso il peso (in Kg) e l'altezza (in m) di una persona, calcoli il BMI.

$$BMI = peso/(altezza^2)$$

Il programma dovrà successivamente assegnare una delle seguenti tre classi:

- sottopeso se BMI <= 20
- normopeso se 20<BMI<=30
- sovrappeso se BMI >30

```
peso = float(input("Inserisci il peso in Kg: "))
altezza = float(input("Inserisci l'altezza in m: "))

bmi = peso / altezza**2

print("Il bmi vale: {:.2f}".format(bmi))

if bmi <= 20:
    print("Sei sottopeso")
elif 20 < bmi <= 30:
    print("Sei normopeso")
else:
    print("Sei sovrappeso")
```

Calcolatrive v.2

```
num_1 = float(input("Inserisci un numero: "))
op = input("Inserisci l'operazione: ")
num_2 = float(input("Inserisci un altro numero: "))

if op == "+":
    print("Risultato = ", num_1 + num_2)
elif op == "-":
    print("Risultato = ", num_1 - num_2)
elif op == "*":
    print("Risultato = ", num_1 * num_2)
elif op == "/":
    print("Risultato = ", num_1 / num_2)
else:
    print("Operazione non valida!")
```

Istruzioni iterative

Ripetere un'operazione un certo numero di volte

```
print(1)
print(2)
print(3)
```

```
for i in (1, 2, 3):
    print(i)
```

```
for valore in sequenza:
    istruzione/i
```

- ad ogni iterazione `i` assume i valori indicati nelle `()` e viene eseguita l'istruzione che segue i `:`
- l'operatore `in` assume `True` o `False`

Istruzioni iterative

```
somma = 0
for i in (0, 1, 2):
    somma = somma + i
print(somma)
```

```
somma = 0
for i in range(3):
    somma = somma + i
print(somma)
```

Istruzioni iterative

```
somma = 0
for i in range(3):
    n = int(input("Inserisci un numero: "))
    somma = somma + n
print("La somma vale: " + str(somma))
```

Istruzioni iterative

```
while espressione:
    istruzione/i
```

```
somma = 0
i=1
numero = 1

while numero != 0 and i <= 10:
    numero = input("Ins. valore: ")
    numero = int(numero)
    somma = somma + numero
    i = i + 1
print("Somma = ", somma)
```

Istruzioni iterative: media

```
somma = 0
i=0
numero = 1
while numero != 0:
    numero = input("Ins. valore: ")
    numero = int(numero)
    somma = somma + numero
    i = i+1
print("Somma = ", somma)
#print("i = ", i)
if somma != 0:
    media = somma / (i - 1)
    print("Media = ", media)
```

Esercizio

Scrivere un programma che calcoli la media di N numeri. N è definito dall'utente.

```
print("Quanti numeri vuoi inserire: ")
n = int(input())
print("Inserisci i " + str(n) + " numeri: ")
somma = 0
for i in range(n):
    numero = int(input("Inserisci un numero: "))
    somma = somma + numero
media = somma / n
print("La media vale: " + str(media))
```

Immutable / Mutable

int, float e stringhe sono *immutable*

```
x = 10
y = x
y = 20
print(x) # stampa 10
```

Le liste sono *mutable*

```
x = [1, 2, 3]
y = x
print(y)
x.append(4)
print(y)
```

Collezioni

Oggetti di tipo sequenza: stringhe, tuple, liste e dizionari.

Tuple

Collezioni ordinate e non modificabili

- possono contenere duplicati
- possono contenere valori eterogenei

`x = (1, 2, 3)` o `x = 1, 2, 3`

Per verificare il tipo: `print(type(x))`

```
x = 1,
print(x)
print(type(x))
y = 1
print(y)
print(type(y))
```

Tuple e subscription

```
x = (1, 2, 3)
print(x[0])
```

Attenzione!

```
x = (1, 2, 3)
print(x[3])
```

```
x = (1, 2, 3)
x[1] = 10
print(x[0])
```

Tuple

Invece...

```
x = (1, 2, 3)
print(x)
x = (10, 20, 30)
print(x)
```

Tuple e slicing

```
x = (1, 2, 3, 4, 5, 6)
print(x[1:3])
print(x[2:])
print(x[:3])
```

Tuple e `in`

```
x = (1, 2, 3, 4, 5, 6)
for i in x:
    print(i)
```

Tuple e metodi

`count` e `index`

```
x = (10, 20, 10, 40, 10, 60)
print(x.count(10))
```

```
x = (10, 20, 10, 40, 10, 60)
print(x.index(40))
```

Liste

Collezioni ordinate e modificabili

- possono contenere valori duplicati
- possono contenere valori eterogenei

```
x = [10, 20, 10, 40, 10, 60]
```

```
x = [10, 20, 10, 40, 10, 60]
print(x)
print(type(x))
```

Liste e substription

```
x = [10, 20, 10, 40, 10, 60]
x[0] = 11
print(x)
```

```
x = [10, 20, 10, 40, 10, 60]
x.append(100)
print(x)
```

Liste e inizializzazione

```
x = []
for i in range(5):
    print("Inserisci numero: ", end='')
    x.append(int(input()))
print(x)
```

```
x.insert(0, 99)
print(x)
```

```
x.sort()
print(x)
```

Esercizio

Scrivere un programma che memorizzi N voti e determini la media, il voto maggiore e il voto minore

```
n = int(input("Quanti voti vuoi inserire? "))
print(n)
voti = []
for i in range(n):
    print("Inserisci voto: ")
    voti.append(int(input()))
massimo = voti[0]
minimo = voti[0]
somma = 0
for x in voti:
    somma = somma + x
    if x > massimo:
        massimo = x
    if x < minimo:
        minimo = x
media = somma / n
print("Massimo: ", massimo)
print("Minimo: ", minimo)
print("Media: ", "{:.2f}".format(media))
```

Liste - attenzione

```
x = [1, 3, 5, 2]
print("x: ", x)
y = x
z = x[:]
x.sort()
print("x: ", x)
print("y: ", y)
print("z: ", z)
```

Liste - attenzione

```
x = [1, 3, 5, 2]
print("x: ", x)
y = x.copy()
z = x[:]
x.sort()
print("x: ", x)
print("y: ", y)
print("z: ", z)
```

Le funzioni

Una *funzione* è costituita da insieme di istruzioni utili ad eseguire una specifica attività (sottoprogramma)
La funzione viene *(ri)chiamata* ogni volta che si vuole eseguire tale attività
L'uso delle funzioni è particolarmente utile per organizzare il codice
Keyword `def`

Le funzioni: primo esempio

Assegnazione di un *nome* alla funzione

`def nome_funzione():`

(**intestazione della funzione**)

Definzione:

```
def saluta():
    print("Ciao") #corpo della funzione
```

Chiamata:

```
saluta() #chiamata
```

Le funzioni: i parametri

```
def saluta(nome):
    print("Ciao " + nome)


saluta("Matteo")
saluta("Maria")
```

```
def saluta(nome1, nome2):
    print("Ciao " + nome1 + " e " + nome2)


saluta("Matteo", "Maria")
```

Le funzioni: i parametri

```
def somma(n1, n2):
    print("La somma vale: ", n1 + n2)


somma(2, 3)
```

Le funzioni: i parametri

```
def saluta(x):
    print("Ciao " + x)


nome = input("Inserisci un nome: ")
saluta(nome)
```

Le funzioni: i parametri

```
def somma(n1, n2):
    print("La somma vale: ", n1 + n2)


num1 = int(input("Inserisci un numero: "))
num2 = int(input("Inserisci un altro numero: "))
somma(num1, num2)
```

Le funzioni: return

Permette di restituire informazioni al programma principale

Keyword `return`

```
def cubo(n):
    return n * n * n


risultato = cubo(2)
print(risultato)
```

Le funzioni: return

```
def cubo(n):
    return n * n * n


num = int(input("Inserisci un numero: "))
risultato = cubo(num)
print("Il cubo di " + str(num) + " vale: " + str(risultato))
```

Esercizio

Creare un programma che calcoli la potenza di un numero `n` dato in ingresso l'esponente `e`

Quanto vale `n^e` ?

Nota: esiste la funzione pow() ma in questo caso non vogliamo usarla!

Soluzione

```
def eleva(n, e):
    ris = 1
    for i in range(e):
        ris = ris * n
    return ris


risultato = eleva(2, 3)
print("La potenza vale: ", risultato)
```

Soluzione

```
def eleva(n, e):
    """
    Funzione che calcola il cubo di n
    usando e come esponente
    """
    ris = 1
    for i in range(e):
        ris = ris * n
    return ris


base = int(input("Inserisci la base: "))
esponente = int(input("Inserisci l'esponente: "))
risultato = eleva(base, esponente)
print(str(base) + "^" + str(esponente) + " = " + str(risultato))
```

`""" docstring """` descrive il comportamento della funzione

Funzioni e visibilità

Concetto di **visibilità** (o *scope*)

```
def my_f():
    x = 0
    print("x in my_f vale: ", x)


x = 1
print("x in main vale: ", x)
my_f()
print("x in main vale: ", x)
```

Funzioni e visibilità

Concetto di **visibilità** (o *scope*)

Attenzione!

```
def my_f():
    x = 0
    print("x in my_f vale: ", x)


print("x in main vale: ", x)
```

Funzioni e visibilità

Concetto di **visibilità** (o *scope*)

Variabili globali

```
def my_f():
    print("x in my_f vale: ", x)


x = 1
my_f()
```

Funzioni e visibilità

Concetto di **visibilità** (o *scope*)

Variabili globali e immutabili

```
def my_f():
    x = 0
    print("x in my_f vale: ", x)


x = 1
my_f() #non modifica x globale
print("x in main vale: ", x)
```

Funzioni e visibilità

Concetto di **visibilità** (o *scope*)

Variabili globali e immutabili

```
def my_f():
    global x
    x = 0
    print("x in my_f vale: ", x)


x = 1
my_f() #modifica x globale
print("x in main vale: ", x)
```

Meglio evitare l'uso di variabili globali e usare invece il passaggio dei parametri

Domande?

www.menti.com