PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PYTHON

Prof. Ada Bagozi
ada.bagozi@unibs.it

Cos'è il Python?

- ✓ *Python* è un linguaggio di programmazione ad alto livello, interpretato e dinamico
- ✓ Creato da Guido van Rossum e rilasciato nel 1991
- ✓ Alcune caratteristiche "tecniche"
  - ✓ è un linguaggio di scripting
  - ✓ è interpretato (non compilato)
  - ✓ è multipiattaforma
  - ✓ è tipizzato dinamicamente (non fortemente tipizzato)
  - ✓ supporta la programmazione ad oggetti, funzionale e imperativa
- ✓ È *Open Source e ha una vasta community*

Linguaggi compilati

- ✓ Il programmatore scrive il codice sorgente in un file di testo
- ✓ Il compilatore traduce il codice sorgente e produce un codice eseguibile (linguaggio macchina)
- ✓ L'esecutore carica il codice eseguibile nella memoria del computer e lo esegue
- ✓ Es. C, C++

Linguaggi interpretati

- ✓ Il programmatore scrive il codice sorgente in un file di testo
- ✓ L' interprete traduce (ad ogni esecuzione) il codice sorgente, trasformandolo in istruzioni del linguaggio macchina che vengono eseguite
- ✓ Es. python, php

Why Python?

- ✓ Progettato per essere semplice da leggere e scrivere
- ✓ Utilizzato per:
  - ✓ Sviluppo web (Django, Flask)
  - ✓ Data Science e Machine Learning
  - ✓ Automazione, scripting, giochi, applicazioni desktop
- ✓ Ampiamente adottato nella ricerca, nell'industria e nella didattica

Installazione

- ✓ Installazione tramite link ufficiale
  https://www.python.org/doc/versions
- ✓ tramite installer integrato:
  - ✓ Linux: apt get

```
sudo apt update
sudo apt install python3 python3-pip
```

  - ✓ Mac: homebrew

```
brew install python
```

    python3 e pip3 vengono installati automaticamente
- ✓ Verifica

```
python --version
pip --version
```

```
python3 --version
pip3 --version
```

- ✓ Cos'è pip?
  pip è il gestore di pacchetti per Python.
  Ti consente di installare librerie aggiuntive come requests, pandas, flask, ecc.

```
pip install nome-pacchetto
```

Strumenti utili

jupyter

Google colab

ANACONDA®

Jupyter

https://jupyter.org/

- ✓ Applicazione web interattiva per scrivere ed eseguire codice Python
- ✓ Perfetta per data science, analisi interattiva e prototipazione
- ✓ Ogni cella può contenere codice, output, testi Markdown o immagini
- ✓ Si può installare con pip o viene incluso con Anaconda

Google Colab

https://colab.google/

- ✓ Versione cloud gratuita di Jupyter Notebook offerta da Google
- ✓ Non richiede installazione né configurazione
- ✓ Supporta l'uso di CPU, GPU e TPU gratuite
- ✓ Ideale per studenti e progetti collaborativi

Anaconda

https://www.anaconda.com/docs/getting-started/anaconda/install

- ✓ Distribuzione Python pensata per Data Science
- ✓ Include Python, Jupyter Notebook, Spyder, pandas, numpy e altri strumenti scientifici
- ✓ Contiene conda, un gestore di pacchetti e ambienti virtuali
- ✓ Ottimo per chi lavora su progetti complessi o in ambito accademico/professionale

Sintassi di base

- ✓ Il codice Python non ha bisogno di tag di apertura/chiusura
- ✓ Le istruzioni non richiedono il punto e virgola (ma è accettato)
- ✓ L'indentazione è obbligatoria e definisce i blocchi di codice
- ✓ I commenti possono essere:
  - ✓ Su una riga usando #
  - ✓ Su più righe: usando triple virgolette ''' commento ''' o """ commento """

```
age = 17;
if age < 18:
  print("Minorenne")
  print("Non puoi proseguire sul sito")
else:
  print("Adulto")
```

```
# questo è un commento su una riga

'''
questo è un commento
su più righe
'''

"""
anche questo è un commento
su più righe
"""
```

Variabili

- ✓ Nome di una variabile (identificatore):
  - ✓ Inizia con una lettera (a-z, A-Z) o un underscore _
  - ✓ Formato da lettere, cifre e underscore '_'
  - ✓ Lunghezza illimitata
  - ✓ Non può iniziare con una cifra
  - ✓ Nessun simbolo speciale come $, -, @, ecc.
  - ✓ Case sensitive: `nome ≠ Nome ≠ NOME`

- ✓ Una variabile viene creata nel momento in cui viene assegnata la prima volta

```
quantity = 0
costo = 0.00
...
quantity = q
```

Tipi di dato

Tipo	Descrizione	Esempio
int	Numeri interi	x = 5
float	Numeri decimali	pi = 3.14
str	Stringhe di testo	nome = ”Ares"
bool	Valori logici: True o False	attivo = True
list	Liste di valori ordinati	numeri = [1, 2, 3]
dict	Dizionari (coppie chiave:valore)	persona = {"nome": "Ada"}
tuple	Sequenze immutabili	coordinate = (4, 5)
set	Insiemi non ordinati senza duplicati	colori = {"rosso", "blu"}

Esistono altri tipi “speciali”:

- ✓ `None`: variabili cui non è assegnato un valore o sono state assegnate con NULL
- ✓ Oggetti/risorse: istanze di classi, connessioni a file o database

Tipizzazione delle variabili in Python

- ✓ Le variabili sono molto “elastiche” nell’assegnamento del tipo di dati

```
quantity = 0
quantity = 'Hello'
```

- ✓ Casting

```
x = 10
costo = float(x)  # 10.0
```

Tipizzazione debole e liste (I)

- ✓ Le liste sono **dinamiche e eterogenee**, cioè possono contenere elementi di tipo diverso.

Dichiarazione di una lista

```
numeri = [1, 2, 3, 4, 5]
frutti = ["mela", "banana", "ciliegia"]
misto = [1, "due", 3.0, True]
```

Accesso e modifica

```
print(frutti[1]) # banana
frutti[0] = "pera" # sostituisce mela con pera
```

Liste multidimensionali (array 2D)

```
matrice = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
print(matrice[1][2]) # 6
```

Tipizzazione debole e liste (II)

- ✓ Le liste sono **dinamiche e eterogenee**, cioè possono contenere elementi di tipo diverso.

  Funzioni e metodi utili

```
len(frutti) # lunghezza
frutti.append("kiwi") # aggiunge elemento
frutti.pop() # rimuove ultimo elemento
"mela" in frutti # verifica presenza
```

Altre tipologie di dati

- ✓ Dizionari

```
persona = {"nome": "Ada"}
print(persona["nome"]) # stampa "Ada"
print(persona.get("nome")) # stampa "Ada"
persona["cognome"] = "Lovelace" # aggiunge chiave
print(persona) # stampa {"nome": "Ada", "cognome": "Lovelace"}
```

- ✓ Coordinate

```
coordinate = (4, 5)
print(coordinate) # stampa (4, 5)
print(coordinate[0]) # stampa 4
coordinate[0] = 10 # errore, le tuple sono immutabili
```

- ✓ Set

```
colori = {"rosso", "blu"}
print(colori) # stampa {"rosso", "blu"}
colori.add("verde") # aggiunge "verde"
print(colori) # stampa {"rosso", "blu", "verde"}
print("rosso" in colori) # True
colori.add("rosso") # non aggiunge duplicato
print(colori[0]) # errore, gli insiemi non supportano l'indicizzazione
print(len(colori)) # stampa 3
colori.pop() # rimuove un elemento casuale
print(colori['rosso']) # errore, gli insiemi non supportano le chiavi
for colore in colori:
print(colore) # stampa ogni colore nell'insieme
```

Variabili predefinite

Python non dispone di "superglobali" come PHP (es. $_SERVER), ma fornisce moduli e funzioni per accedere a dati simili

```
import os
print(os.environ) # stampa le variabili d'ambiente
```

Per accedere a informazioni HTTP in un contesto web, si utilizzano framework come Flask o Django. Esempio con Flask:

```
from flask import request

@app.route("/")
def index():
user_agent = request.headers.get('User-Agent')
return f"Il tuo browser è: {user_agent}"
```

Non esiste un equivalente diretto a $_SERVER["HTTP_HOST"] o $_SERVER["PHP_SELF"], ma strumenti analoghi sono forniti dai web framework.

Lavorare con le variabili

In Python puoi inserire facilmente variabili all'interno di stringhe:

```
''' Metodo 1: f-string (consigliato) '''
qbanane = 3
print(f"{qbanane} banane")

''' Metodo 2: concatenazione '''
qbanane = 3
print(str(qbanane) + " banane")

''' Metodo 3: format() '''
qbanane = 3
print("{} banane".format(qbanane))

''' Esempi '''
nome = "Ada"
cognome = "Lovelace"

print(f"Ciao {nome}") # stampa Ciao Ada
print('Ciao {nome}') # stampa letteralmente Ciao {nome}
print(f'Ciao {nome}') # stampa Ciao Ada
print("Ciao {} {}".format(nome, cognome)) # stampa "Ciao Ada Lovelace"
print("Ciao {1} {0}".format(nome, cognome)) # stampa "Ciao Lovelace Ada"
print("Ciao {0} {1}".format(nome, cognome)) # stampa "Ciao Ada Lovelace"
```

Costanti

- ✓ In Python non esistono costanti vere, ma per convenzione si usano nomi in MAIUSCOLO

```
PI = 3.14159
MAX_UTENTI = 100
```

- ✓ Non esiste un meccanismo che impedisce la modifica

```
MAX_UTENTI = 200 # possibile, ma sconsigliato!
```

- ✓ Le costanti non sono vincolate, ma il maiuscolo segnala al programmatore che non dovrebbe modificarle

Alternative alle Costanti

Enum

- ✓ Utile per gruppi di costanti semantiche
- ✓ È possibile confrontare valori in modo chiaro:
  if oggi == Giorno.LUNEDI

```python
from enum import Enum

class Giorno(Enum):
LUNEDI = 1
MARTEDI = 2
MERCOLEDI = 3

# Uso
oggi = Giorno.LUNEDI
print(oggi) # Giorno.LUNEDI
print(oggi.name) # LUNEDI
print(oggi.value) # 1
```

dataclass

- ✓ Frozen=True rende l'istanza immutabile (come una costante)
- ✓ Utile per impostazioni condivise o valori globali

```python
from dataclasses import dataclass

@dataclass(frozen=True)
class Config:
DB_HOST: str = "localhost"
DB_PORT: int = 5432
DEBUG: bool = False

config = Config()
print(config.DB_HOST) # localhost
```

Operatori (I)

Operatori **aritmetici**

Operatore	Significato	Esempio	Risultato
+	Addizione	5 + 3	8
-	Sottrazione	5 - 2	3
*	Moltiplicazione	4 * 2	8
/	Divisione	10 / 4	2.5
//	Divisione intera	10 // 4	2
%	Modulo (resto)	10 % 4	2
**	Potenza	2 ** 3	8

- ✓ Gli operatori funzionano su int e float
- ✓ L'operatore divisione intera (/) restituisce sempre un float

Operatori (II)

Operatori **su stringhe**

- ✓ Operatore +
- ✓ f-string (più leggibile)
- ✓ join() per concatenare una lista di stringhe
- ✓ Non si può concatenare una stringa con un intero direttamente

```
nome = "Ada"
saluto = "Ciao, " + nome + "!"
print(saluto) # Ciao, Ada!


nome = "Ada"
print(f"Ciao, {nome}!")



parole = ["Python", "è", "facile"]
print(" ".join(parole)) # Python è facile









eta = 30
print("Hai " + eta + " anni") # Errore!
print("Hai " + str(eta) + " anni") # Corretto
```

Operatori (III)

Operatore di **assegnamento**

<variabile> = valore

```
b = 6
a = 5
b = 6 + a # stampa 11
```

```
b = 6 + (a = 5) # errore non è possibile assignare a una variabile in un'espressione
```

Operatori di assegnamento combinato

Operatore	Esempio	Equivalente a
+=	x += 1	x = x + 1
-=	x -= 2	x = x - 2
*=	x *= 3	x = x * 3
/=	x /= 4	x = x / 4
//=	x //= 2	x = x // 2
%=	x %= 5	x = x % 5
**=	x **= 2	x = x ** 2

Operatori (IV)

Python **non support** operatore di incremento (++) e decremento (--)

Operatori di **confronto**

Operatore	Significato	Esempio	Risultato
==	Uguale a	5 == 5	True
!=	Diverso da	5 != 3	True
>	Maggiore di	7 > 4	True
<	Minore di	2 < 1	False
>=	Maggiore o uguale	5 >= 5	True
<=	Minore o uguale	3 <= 2	False

Operatori (V)

Operatore **identità**

Operatore	Significato	Esempio	Risultato
is	Stesso oggetto in memoria (identità)	a is b	True / False
is not	Oggetti diversi	a is not b	True / False

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print(a == c) # True → stessi valori
print(a is c) # False → oggetti diversi
print(a is b) # True → stesso oggetto


if variabile is None:
   print("Variabile non inizializzata")
```

Operatori (VI)

Operatori logici

Operatore	Significato	Esempio	Risultato
and	E logico	True and False	False
or	O logico	True or False	True
not	Negazione logica	not True	False

```
eta = 20
studente = True

if eta > 18 and studente:
  print("Studente adulto")

if not studente:
  print("Non sei uno studente")
```

Operatori (VII)

Operatore **condizionale** (ternario)

(<condizione> **if** <valore se vera> **else** <valore se falsa> **)**

```
# valore_se_vero if condizione else valore_se_falso
eta = 20
messaggio = "Adulto" if eta >= 18 else "Minorenne”

print(messaggio) # Adulto

logged_in = True
print("Accesso consentito" if logged_in else "Accesso negato")
```

Type juggling

- ✓ Alcune conversioni di tipo avvengono automaticamente in base agli operatori utilizzati nelle espressioni
- ✓ La conversione non modifica il tipo degli operandi, che rimangono inalterati
- ✓ Python è più rigido di PHP nel type juggling: non tenta conversioni per operazioni ambigue

```php
<?php

$numero = 5;
$testo = "10 banane";
$somma = $numero + $testo; // PHP converte automaticamente "10 banane" in 10
echo $somma; // 15
```

```python
numero = 5
testo = "10 banane"
somma = numero + testo # Errore! non puoi sommare un numero e una stringa

# Conversione esplicita
somma = numero + int(testo.split()[0]) # 5 + 10

print(somma) # 15
```

Funzioni di accesso al tipo

PHP	Python equivalente	Cosa fa
is_boolean($a)	isinstance(a, bool)	Verifica se la variabile contiene un valore booleano
is_integer($a)	isinstance(a, int)	Verifica se la variabile contiene un numero intero
is_float($a) / is_double($a)	isinstance(a, float)	Verifica se la variabile contiene un numero reale
is_array($a)	isinstance(a, list)	Verifica se la variabile è una lista
is_resource($a)	nessun equivalente diretto	In Python si usano oggetti o gestori di contesto
is_null($a)	a is None	Verifica se la variabile contiene il valore None
is_numeric($a)	a.isnumeric() (solo per str)	Verifica se una stringa è convertibile in numero
gettype($a)	type(a).__name__	Restituisce il nome del tipo della variabile
—	callable(a)	Verifica se la variabile è una funzione o può essere chiamata

L’istruzione if


```
if<condizione>:
  <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se <condizione> è vera

```
quantita_totale = 0

if quantita_totale == 0:
  print("Non hai ordinato alcun articolo!")


costo_totale = 32000

if costo_totale >= 30000:
  print("Hai diritto ad uno sconto del 10%")
```

L'istruzione else


```
else:
  <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se la condizione del precedente `if` è falsa

```
qpomodori = 2
qbanane = 3
qmele = 4
quantita_totale = qpomodori + qbanane + qmele

if quantita_totale == 0:
  print("Non hai ordinato alcun articolo!<br />")
else:
  print("Ecco la lista degli articoli:<br />")
  print("<ul>")
  print(f"<li>{qpomodori} pomodori</li>")
  print(f"<li>{qbanane} banane</li>")
  print(f"<li>{qmele} mele</li>")
  print("</ul>")
```

L'istruzione elseif

È usato quando si hanno più di due rami decisionali:

```
elif <condizione>:
    <istruzioni>
```

```
if costo_totale <= 10000:
  sconto = 0.0
elif costo_totale <= 20000:
  sconto = 0.10
elif costo_totale <= 40000:
  sconto = 0.20
elif costo_totale <= 80000:
  sconto = 0.25
else:
  sconto = 0.30
```

Viene eseguito solo il blocco di istruzioni corrispondente alla prima condizione vera

Se nessuna condizione vera, viene eseguito il blocco della `else` (se specificata)

L'istruzione switch (I)

In Python non esiste l'istruzione **`switch`** come in PHP, C o JavaScript

```
switch ($nome)
{
      case 'Luca':
      case 'Giorgio':
      case 'Franco':
      echo "Ciao, vecchio amico!"; break;
      case 'Mario': echo "Ciao, Mario!"; break;
      default: print "Benvenuto, chiunque tu sia";
}
```

Uso di if-elif-else

```
nome = "Luca"
if nome in ("Luca", "Giorgio", "Franco"):
  print("Ciao, vecchio amico!")
elif nome == "Mario":
  print("Ciao, Mario!")
else:
  print("Benvenuto, chiunque tu sia")
```

L’istruzione switch (II)

Con dizionario di risposte

```
nome = "Luca"

messaggi = {
   "Luca": "Ciao Luca, vecchio amico!",
   "Giorgio": "Ciao Giorgio, vecchio amico!",
   "Franco": "Ciao Franco, vecchio amico!",
   "Mario": "Ciao, Mario!"
}

print(messaggi.get(nome, "Benvenuto, chiunque tu sia"))
```

Python 3.10+ – Con match (pattern matching)

```
match nome:
  case "Giorgio" | "Franco":
    print("Ciao, vecchio amico!")
  case "Luca":
    print("Ciao, Luca!")
  case "Mario":
    print("Ciao, Mario!")
  case _:
    print("Benvenuto, chiunque tu sia")
```

Ciclo while


```
while <condizione>:
  <istruzioni>
```

Il blocco <istruzioni> viene eseguito fino a quando <condizione> è vera

```
x = 0
while x < 5:
  print(x)
  x += 1
```

In Python tutti i cicli condizionati si fanno con while, **non esiste** il do...while

Ciclo for


```
for variable in range(start, stop, step):
  <istruzioni>
```

```python
for i in range(5):
  print(i) # stampa 0, 1, 2, 3, 4


for i in range(1, 10, 2):
  print(i) # stampa 1, 3, 5, 7, 9


nomi = ["Luca", "Sara", "Giorgio"]
for nome in nomi:
  print(f"Ciao, {nome}")
```

Funzioni


```
def nome_funzione(parametri):
  # corpo della funzione
  valore = 5
  return valore
```

Argomenti variabili: *args e **kwargs

```
def somma(*numeri):
  return sum(numeri)

print(somma(1, 2, 3)) # 6
```

```
def stampa_info(**info):
  for chiave, valore in info.items():
    print(f"{chiave}: {valore}")

stampa_info(nome="Luca", eta=30)
```

Funzioni anonime - lambda


```
quadrato = lambda x: x * x
print(quadrato(4)) # 16


numeri = [1, 2, 3, 4]
doppio = list(map(lambda x: x * 2, numeri))
print(doppio) # [2, 4, 6, 8]


numeri = [1, 2, 3, 4, 5, 6]
pari = list(filter(lambda x: x % 2 == 0, numeri))
print(pari) # [2, 4, 6]


parole = ["banana", "kiwi", "mela"]
ordinate = sorted(parole, key=lambda x: len(x))
print(ordinate) # ['kiwi', 'mela', 'banana']
```

PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PYTHON

Prof. Ada Bagozi
ada.bagozi@unibs.it