SVILUPPO APPLICAZIONI E ANALISI DATI CON PYTHON

Fondamenti di Python

Strutture dati e Liste

Docente: Marino Speciale

Indice della lezione

- Strumenti di Sviluppo Consigliati
- Cosa sono le Liste?
- Creazione e Inizializzazione
- Accesso e Modifica degli Elementi
- Metodi Principali
- Slicing e Iterazione

Strumenti di sviluppo: Panoramica

Per imparare e sviluppare in Python servono:

- **Python 3.x** (runtime)
- **IDE/Editor** (ambiente di sviluppo)
- **Package Manager** (pip, conda)
- **Virtual Environment** (isolamento dipendenze)

IDE Consigliati: PyCharm e VS Code

PyCharm

Pro: Completamento intelligente, debugging avanzato, supporto framework
Contro: Pesante, versione pro a pagamento

→ Professional developers

Visual Studio Code

Pro: Leggero, estensioni, open-source
Contro: Richiede configurazione

→ Uso generale consigliato

IDE Consigliati: Specializzati

Jupyter Lab

Uso: Data science, analisi interattiva
Free: Sì

→ Didattica

Thonny

Uso: Principianti, debugging visuale
Free: Sì

→ Learning

Installazione: Setup Base

```
# 1. Installa Python 3.x
# Da python.org oppure package manager

# 2. Crea virtual environment
python -m venv mio_env

# 3. Attiva virtual environment
# Su Windows:
mio_env\Scripts\activate
# Su macOS/Linux:
source mio_env/bin/activate

# 4. Installa packages con pip
pip install numpy pandas matplotlib
```

VS Code: Setup consigliato

Estensioni da installare:

- Python (Microsoft)
- Python Debugger (Microsoft)
- Pylance (completamento intelligente)
- Code Runner (esecuzione rapida)

PyCharm: Setup consigliato

Versione Community (Free):

- Debugging avanzato integrato
- Completamento codice eccellente
- Virtual environment manager
- Test runner integrato

Perfetto per didattica e sviluppo serio

Strumenti aggiuntivi consigliati

```
# Linting e Formattazione
pip install pylint flake8 black

# Testing
pip install pytest

# Ambiente Data Science
pip install numpy pandas matplotlib jupyter

# Jupyter per didattica interattiva
pip install jupyterlab
jupyter lab  # Avvia il server
```

Best Practices con le Liste

- Usa per iterare, non indici manuali
- Usa per aggiungere singoli elementi
- Preferisci list comprehension quando appropriato
- Evita di modificare liste durante l'iterazione
- Documenta il tipo di dati atteso nella lista

Cosa sono le Liste?

Una lista è una struttura dati **mutabile** e **ordinata** che contiene una collezione di elementi eterogenei.

```
# Creazione di liste
lista_vuota = []
lista_numeri = [1, 2, 3, 4, 5]
lista_mista = ["Python", 3.14, 42, True]
```

Le liste mantengono l'ordine degli elementi e permettono duplicati

Creazione e Inizializzazione

```
# Con square brackets
linguaggi = ["Python", "Java", "JavaScript"]

# Usando il costruttore list()
numeri = list(range(1, 6))  # [1, 2, 3, 4, 5]

# Con elementi di diversi tipi
dati = [1, "due", 3.0, True]

# Lista vuota
lista_vuota = []
```

I square brackets ([]) in Python sono un elemento sintattico fondamentale utilizzato principalmente per due scopi: la creazione di liste e l'accesso (indicizzazione e slicing) agli elementi di sequenze e collezioni.

Accesso agli Elementi (Indexing)

Python usa **indici a partire da 0**. Supporta anche indici negativi:

```
colori = ["rosso", "verde", "blu", "giallo"]
print(colori[0])    # rosso (primo elemento)
print(colori[2])    # blu (terzo elemento)
print(colori[-1])   # giallo (ultimo elemento)
print(colori[-2])   # blu (penultimo elemento)
```

Modifica degli Elementi

Le liste sono **mutabili** - puoi cambiarle:

```
numeri = [1, 2, 3, 4, 5]
numeri[0] = 10        # [10, 2, 3, 4, 5]
numeri[-1] = 50       # [10, 2, 3, 4, 50]

del numeri[1]         # Rimuove elemento: [10, 3, 4, 50]
```

Metodi Principali: Aggiunta

```
lista = [1, 2, 3]

# append() - aggiunge elemento alla fine
lista.append(4)           # [1, 2, 3, 4]

# extend() - aggiunge più elementi
lista.extend([5, 6])      # [1, 2, 3, 4, 5, 6]

# insert() - inserisce a posizione specifica
lista.insert(1, 1.5)      # [1, 1.5, 2, 3, 4, 5, 6]
```

Metodi Principali: Rimozione

```
lista = [1, 2, 3, 4, 5]

# remove() - rimuove primo elemento con valore
lista.remove(3)          # [1, 2, 4, 5]

# pop() - rimuove per indice (default ultimo)
ultimo = lista.pop()     # ultimo = 5, lista = [1, 2, 4]
elemento = lista.pop(0)  # elemento = 1, lista = [2, 4]

# clear() - svuota la lista
lista.clear()            # []
```

Metodi Principali: Ricerca e Ordinamento

```
numeri = [3, 1, 4, 1, 5]

# index() - trova posizione del valore
pos = numeri.index(4)    # pos = 2

# count() - conta occorrenze
volte = numeri.count(1)  # volte = 2

# sort() - ordina la lista (in-place)
numeri.sort()            # [1, 1, 3, 4, 5]

# sorted() - crea lista ordinata
ordinata = sorted([3, 1, 2])  # [1, 2, 3]
```

Slicing: Estrarre Sottoliste

```
numeri = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# [inizio:fine:step] - fine è esclusa
print(numeri[2:5])        # [2, 3, 4]
print(numeri[:3])         # [0, 1, 2]
print(numeri[7:])         # [7, 8, 9]
print(numeri[::2])        # [0, 2, 4, 6, 8]
print(numeri[::-1])       # [9, 8, 7, ..., 0] - invertita
```

Iterazione su Liste

```
colori = ["rosso", "verde", "blu"]

# For loop semplice
for colore in colori:
    print(colore)

# Con enumerate (indice + valore)
for i, colore in enumerate(colori):
    print(f"{i}: {colore}")

# List comprehension
quadrati = [x**2 for x in range(5)]  # [0, 1, 4, 9, 16]
```

Esempio Pratico: Gestione Voti

```
voti = [7, 8, 6, 9, 7, 5]

# Media
media = sum(voti) / len(voti)
print(f"Media: {media:.2f}")  # 7.00

# Voto massimo e minimo
print(f"Max: {max(voti)}, Min: {min(voti)}")

# Conta voti sufficienti
sufficienti = len([v for v in voti if v >= 6])
```

Riepilogo Lezione

✓ Strumenti: VS Code, PyCharm, Jupyter

✓ Liste: strutture ordinate e mutabili

✓ Indexing, slicing, metodi principali

✓ Iterazione e list comprehension

Domande?

Risorse per approfondire:

→ python.org/docs

→ realpython.com

→ official Python Tutorial