Corso Python

Modulo 1: L'Ecosistema e le Fondamenta

Agenda del Modulo 1 (2 Ore)

1. **Intro & Setup:** Python, VS Code e strumenti.
2. **Variabili & Tipi:** Scatole ed etichette.
3. **Dati Primitivi:** Stringhe, Numeri, Booleani.
4. **Operatori:** Matematica e Logica.
5. **Laboratorio:** Scrivere il primo script.

Parte 1: Setup & Ambiente

"Hello World"

Perché Python?

- **Leggibilità:** Sintassi pulita, molto simile all'inglese.
- **Versatilità:** Web, Data Science, AI, Scripting.
- **Batteries Included:** Libreria standard ricchissima.
- **Community:** Una delle più grandi al mondo.

L'Interprete Python

Python è un linguaggio **interpretato**.
C'è un programma (l'interprete) che legge il tuo codice riga per riga e lo esegue.

Verifica installazione:
Apri il terminale e digita:

```
python --version
# oppure
python3 --version
```

Installazione Python: Windows

Passo 1: Vai su python.org/downloads e clicca su **"Download Python 3.x"** per Windows.

Passo 2: Apri il file `.exe` scaricato.

Passo 3: ⚠️ **IMPORTANTE:** Seleziona la casella **"Add python.exe to PATH"** prima di procedere!

Passo 4: Clicca su **"Install Now"** (o "Customize installation" se vuoi cambiare cartella/opzioni).

Installazione Python: Windows (verifica)

Al termine dell'installazione, apri il **Prompt dei comandi** e verifica:

```
python --version
```

oppure

```
python3 --version
```

Se vedi la versione di Python (es. `Python 3.12.1`), l'installazione è andata a buon fine! ✅

Installazione Python: Linux

Spesso Python è **già presente** su Linux, ma puoi installarlo/aggiornarlo dal gestore pacchetti.

Ubuntu/Debian:

```
sudo apt update
sudo apt install python3
```

Fedora:

```
sudo dnf install python3
```

Installazione Python: Linux (verifica)

Verifica l'installazione con:

```
python3 --version
```

Su molte distribuzioni Linux, il comando principale è `python3` (non `python`).

Installazione Python: macOS

Su macOS moderno conviene installare dal **sito ufficiale** oppure usare **Homebrew**.

Metodo 1: Sito Ufficiale

1. Vai su python.org/downloads
2. Scegli **"macOS 64-bit universal2 installer"**
3. Apri il file `.pkg` e segui i passaggi di installazione predefiniti
4. Verifica da Terminale con `python3 --version`

Installazione Python: macOS (Homebrew)

Metodo 2: Homebrew (se lo usi già)

```
brew install python
```

Verifica l'installazione:

```
python3 --version
```

Nota: Su macOS, usa sempre `python3` per evitare conflitti con la versione di sistema.

L'Editor: VS Code

Non scriviamo codice su Notepad! Usiamo un IDE.
VS Code è lo standard attuale.

Estensioni Essenziali:

1. **Python (Microsoft):** Debugging, Intellisense.
2. **Pylance:** Analisi codice performante.
3. **Material Icon Theme:** (Opzionale) Per icone file leggibili.

Jupyter Notebooks in VS Code

I **Notebook** sono documenti interattivi che combinano:

- **Codice eseguibile** (in celle separate)
- **Testo formattato** (Markdown)
- **Output e visualizzazioni** (grafici, tabelle)

Perfetti per:

- Sperimentare codice
- Data Science e analisi dati
- Documentazione interattiva

Come Usare i Notebook in VS Code

1. **Installa l'estensione** "Jupyter" da Microsoft.
2. **Crea un nuovo notebook:**
   - `Ctrl+Shift+P` → "Create: New Jupyter Notebook"
   - Oppure crea un file `.ipynb`
3. **Seleziona il Python Interpreter** (in alto a destra).
4. **Scrivi codice** nelle celle e premi `Shift+Enter` per eseguire.

Google Colab: Notebook sul Cloud

Google Colaboratory (Colab) è un servizio **gratuito** di Google.

Vantaggi:

- ✅ **Nessuna installazione:** Basta un browser.
- ✅ **GPU/TPU gratis:** Per calcoli intensivi (AI, ML).
- ✅ **Salvataggio su Google Drive:** I tuoi notebook sempre disponibili.
- ✅ **Collaborazione:** Come Google Docs, ma per codice.

Come Accedere a Google Colab

1. Vai su colab.research.google.com
2. Accedi con il tuo **account Google**
3. Scegli:
   - **Nuovo Notebook** (vuoto)
   - **Apri da Drive** (se ne hai già salvati)
   - **Carica** un file `.ipynb` dal tuo PC

Interfaccia di Google Colab

- **Celle di Codice:** `+ Codice` per aggiungerne una.
- **Celle di Testo:** `+ Testo` per Markdown.
- **Eseguire:** Clicca sul ▶️ a sinistra della cella, oppure `Shift+Enter`.
- **Runtime:** In alto, puoi cambiare tipo (Python 3, GPU, TPU).

Shortcut utili:

- `Ctrl+Enter`: Esegue la cella corrente.
- `Shift+Enter`: Esegue e passa alla cella successiva.

Esempio Pratico: Hello World su Colab

Crea una nuova cella di codice e scrivi:

```
print("Hello from Google Colab!")
```

Premi `Shift+Enter` per eseguire.

Prova anche:

```
import sys
print(f"Python version: {sys.version}")
```

Il tutto senza installare nulla sul tuo PC!

Il Primo Script: Hello World

Crea un file chiamato `main.py`.

```
print("Hello, World!")
```

Esecuzione:

1. Apri il terminale integrato in VS Code.
2. Digita `python main.py`.
3. Premi Invio.

Parte 2: Variabili e Tipi

Dove salviamo i dati?

Cos'è una Variabile?

Immagina una **scatola** con un'etichetta.
Dentro la scatola mettiamo un valore.

```
nome = "Mario"
eta = 30
```

- `nome` è l'etichetta.
- `"Mario"` è il contenuto.
- `=` è l'operatore di **assegnazione** (Metti il valore a destra nella scatola a sinistra).

Naming Convention (Regole dei Nomi)

In Python usiamo lo **snake_case**:

- ✅ `mio_nome_variabile`
- ❌ `MioNomeVariabile` (CamelCase, usato per le Classi)
- ❌ `mio-nome` (kebab-case, non valido)

Regole:

- Non iniziare con numeri (`1nome` ❌).
- Niente spazi (`mio nome` ❌).
- **Case Sensitive:** `Pippo` ≠ `pippo` .

Tipi di Dati Primitivi: Stringhe

Testo racchiuso tra virgolette (`"` o `'`).

```
messaggio = "Ciao a tutti"
carattere = 'A'
```

Concatenazione (Unire stringhe):

```
saluto = "Ciao " + "Mario"
# Risultato: "Ciao Mario"
```

f-strings (Formattazione)

Il modo moderno per inserire variabili nelle stringhe (Python 3.6+).

```
nome = "Luca"
eta = 25

# Senza f-string (scomodo)
print("Mi chiamo " + nome + " e ho " + str(eta) + " anni")

# Con f-string (consigliato)
print(f"Mi chiamo {nome} e ho {eta} anni")
```

Nota la `f` prima delle virgolette!

Metodi delle Stringhe

Le stringhe hanno **metodi** utili per manipolare il testo:

```
testo = "  Ciao Mondo!  "

# Rimuovere spazi
print(testo.strip())      # "Ciao Mondo!"

# Maiuscole/Minuscole
print(testo.upper())      # "  CIAO MONDO!  "
print(testo.lower())      # "  ciao mondo!  "

# Sostituzione
print(testo.replace("Mondo", "Python"))  # "  Ciao Python!  "
```

Metodi delle Stringhe (continua)

```
frase = "uno,due,tre,quattro"

# Dividere in lista
parole = frase.split(",")  # ["uno", "due", "tre", "quattro"]

# Verifiche
print("ciao".startswith("ci"))   # True
print("ciao".endswith("ao"))     # True
print("123".isdigit())           # True
print("abc".isalpha())           # True

# Trovare posizione
print("Python".find("th"))       # 2
print("Python".find("xyz"))      # -1 (non trovato)
```

Tipi Numerici: Int & Float

Interi (`int`): Numeri senza virgola.

```
x = 10
y = -5
```

Decimali (`float`): Numeri con la virgola (punto).

```
prezzo = 9.99
pi_greco = 3.14159
```

Booleani e None

Booleani (`bool`): Solo due valori possibili. Fondamentali per la logica.

```
is_online = True
is_admin = False
```

Attenzione alla maiuscola!

NoneType (`None`):
Rappresenta l'assenza di valore (null).

```
risultato = None
```

Parte 3: Operatori

Matematica e Logica

Operatori Matematici

Simbolo	Descrizione	Esempio
`+`	Somma	`10 + 5` → 15
`-`	Sottrazione	`10 - 2` → 8
`*`	Moltiplicazione	`3 * 4` → 12
`/`	Divisione (float)	`10 / 2` → 5.0
`//`	Divisione Intera	`10 // 3` → 3
`%`	Modulo (Resto)	`10 % 3` → 1
`**`	Potenza	`2 ** 3` → 8

Operatori di Confronto

Restituiscono sempre `True` o `False`.

```
x = 10
y = 20

print(x == y)  # Uguale a: False
print(x ! = y)  # Diverso da: True
print(x > 5)   # Maggiore: True
print(x ⩽ 10) # Minore o uguale: True
```

Operatori Logici

Servono a combinare più condizioni.

- `and` : Tutto deve essere vero.
  `True and True` → `True`
- `or` : Basta che uno sia vero.
  `True or False` → `True`
- `not` : Inverte il valore.
  `not True` → `False`

Esempio:

```
login_success = (password_corretta and utente_attivo)
```

Laboratorio 1

"Generatore di Identikit"

Esercizio: Identikit

Obiettivo: Creare uno script `identikit.py` che:

1. Definisca variabili per: `nome`, `anno_nascita`, `città`.
2. Calcoli l'età attuale sottraendo l'anno di nascita all'anno corrente.
3. Usi una **f-string** per stampare una frase completa.
   Es: "Ciao, sono Marco, ho 34 anni e vengo da Roma."

Bonus:

Usa l'operatore `%` per stampare `True` se l'età è un numero pari, `False` se dispari.

Soluzione Attesa

```
# Setup Variabili
nome = "Giulia"
anno_nascita = 1995
anno_corrente = 2026
citta = "Milano"

# Calcolo
eta = anno_corrente - anno_nascita
is_pari = (eta % 2 == 0)

# Output
print(f"Ciao, sono {nome}, ho {eta} anni e vengo da {citta}.")
print(f"La mia età è un numero pari? {is_pari}")
```

Laboratorio 2

"Esplora i Notebook Interattivi"

Esercizio: Il Tuo Primo Notebook

Obiettivo: Creare un notebook interattivo su **Google Colab** o **VS Code**.

Requisiti:

1. **Cella Markdown:** Titolo del notebook e breve descrizione.
2. **Cella Codice #1:** Calcola l'area di un rettangolo (larghezza × altezza).
3. **Cella Markdown:** Spiega la formula usata con il Markdown.
4. **Cella Codice #2:** Crea una lista di numeri e calcola la somma.

Guida Step-by-Step

1. **Apri Google Colab** o crea un `.ipynb` in VS Code.
2. **Prima cella (Markdown):**

```
# Il Mio Primo Notebook Python

Questo notebook esplora i calcoli base.
```

3. **Seconda cella (Codice):**

```
larghezza = 5
altezza = 10
area = larghezza * altezza
print(f"L'area del rettangolo è: {area}")
```

Guida Step-by-Step (continua)

4. **Terza cella (Markdown):**

```
## Formula dell'Area

L'area di un rettangolo si calcola con:
**Area = larghezza × altezza**
```

5. **Quarta cella (Codice):**

```
numeri = [10, 20, 30, 40, 50]
somma_totale = sum(numeri)
print(f"La somma è: {somma_totale}")
```

Funzionalità Avanzate dei Notebook

Markdown Avanzato:

- Liste puntate, tabelle, immagini
- Formule matematiche con LaTeX: `$E = mc^2$`

Visualizzazioni:

- Usa librerie come `matplotlib` per grafici:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [1, 4, 9])
plt.show()
```

Sfida Bonus: Visualizza i Dati

Aggiungi una cella finale con questo codice:

```
import matplotlib.pyplot as plt

# Dati di esempio
mesi = ["Gen", "Feb", "Mar", "Apr", "Mag"]
vendite = [120, 150, 180, 130, 200]

# Crea il grafico
plt.bar(mesi, vendite, color='skyblue')
plt.title("Vendite Mensili")
plt.xlabel("Mese")
plt.ylabel("Vendite")
plt.show()
```

Guarda come il notebook rende il risultato direttamente sotto la cella!

📐 PEP 8: Lo Standard Python

PEP 8 è la guida ufficiale per scrivere codice Python leggibile e professionale.

Perché è importante?

- Il codice si **legge** molto più di quanto si **scrive**
- Facilita la collaborazione in team
- Gli strumenti automatici (linter) si basano su PEP 8

Le regole principali:

- **Indentazione:** 4 spazi (mai tab!)
- **Lunghezza righe:** max 79 caratteri
- **Linee vuote:** 2 tra funzioni, 1 tra blocchi logici
- **Import:** uno per riga, all'inizio del file

📐 Convenzioni di Naming

Nomi diversi per concetti diversi:

Tipo	Convenzione	Esempio
Variabili	`snake_case`	`nome_utente`, `totale_vendite`
Funzioni	`snake_case`	`calcola_totale()`, `invia_email()`
Costanti	`UPPER_CASE`	`MAX_TENTATIVI`, `PI_GRECO`
Classi	`PascalCase`	`Studente`, `ContoBancario`
Privato	`_underscore`	`_variabile_interna`

```
# ❌ Nomi sbagliati
x = "Mario"           # Non descrittivo
nomeUtente = "Mario" # camelCase (è Java, non Python!)

# ✅ Nomi corretti
nome_utente = "Mario"    # snake_case, descrittivo
MAX_TENTATIVI = 3        # Costante in UPPER_CASE
```

📐 Spazi e Formattazione

```python
# ✅ Spazi intorno agli operatori
x = 5 + 3
risultato = a * b - c

# ✅ NO spazi nei keyword arguments
def saluta(nome, saluto="Ciao"):
    print(f"{saluto}, {nome}!")

# ✅ Spazio dopo la virgola
lista = [1, 2, 3, 4]
funzione(arg1, arg2, arg3)

# ❌ Troppi spazi
x=5+3              # Nessuno spazio
lista = [1 , 2 , 3]  # Spazio PRIMA della virgola
```

📐 Strumenti automatici

Strumenti automatici per formattazione e controllo:

- `black` → Formatta automaticamente il codice
- `flake8` / `pylint` → Segnala violazioni PEP 8
- VS Code: installa l'estensione **Pylint** o **Ruff**

⚠️ Errori Comuni dei Principianti

Errore	Problema	Soluzione
`=` vs `==`	`if x = 5`	Usa `==` per confronto
Case Sensitivity	`Print()` non funziona	È `print()` minuscolo
Virgolette miste	`"ciao'`	Usa lo stesso tipo `"ciao"`
Spazi/Tab misti	`IndentationError`	Usa solo spazi (4)

```
# ❌ SBAGLIATO
if x = 5:       # Assegnazione, non confronto!
    Print("OK")  # P maiuscola

# ✅ CORRETTO
if x == 5:      # Confronto
    print("OK")  # p minuscola
```

🎯 Progetto Cross-Modulo: Rubrica Contatti

Parte 1: Le Basi

Costruiremo una **rubrica contatti** che evolverà in ogni modulo!
Obiettivo Modulo 1: Definire le variabili per UN contatto.

```
# Il nostro primo contatto
nome = "Mario Rossi"
telefono = "333-1234567"
email = "mario.rossi@email.com"
eta = 30
# Stampa formattata
print(f"Contatto: {nome}")
# Ripeti per telefono, email e eta
```

Prossimo modulo: Gestiremo MOLTI contatti con le liste!

Riepilogo Modulo 1

Concetto	Esempio
Variabile	`nome = "Mario"`
Stringa	`"Testo"` o `'Testo'`
f-string	`f"Ciao {nome}"`
Numeri	`int: 42`, `float: 3.14`
Booleano	`True`, `False`
Operatori	`+`, `-`, `*`, `/`, `==`, `and`, `or`

Prossimo Modulo: Strutture Dati (Liste, Dizionari, Tuple, Set)