SuperDuper Py 01

Ciao youtubers e benvenuti in SuperDuper PY una serie di tutorial pensati per chi è all'inizio con Python e sta cercando un qualcosa che non sia il classico tutorial tipo addominali in 7 minuti o che spieghi ogni singolo comando senza dare esempi o idee.

Python è un linguaggio di programmazione versatile e potente, ed è un ottimo punto di partenza per chiunque voglia imparare a programmare. In questo tutorial, esploreremo i concetti fondamentali di Python, compresi gli oggetti, le variabili, le espressioni e gli operatori.

Potrebbe essere il caso che Python non sia ancora installato sul vostro computer. Per risolvere questo problema, dovete seguire questi passaggi per installare Python:

Scaricare Python: Andate sul sito ufficiale di Python all'indirizzo https://www.python.org/

Scegliere la Versione: Vi verranno presentate due versioni principali, Python 2 e Python 3. Assicuratevi di scaricare Python 3, poiché Python 2 è obsoleto. Selezionate la versione più recente di Python 3 disponibile.

Scegliere il Sistema Operativo: Scegliete la versione di Python compatibile con il vostro sistema operativo. Se avete un sistema Windows, sarà disponibile una versione specifica per Windows. Se avete un sistema macOS, è disponibile un installer per macOS. Per Linux, Python è spesso preinstallato o può essere installato tramite il vostro gestore di pacchetti.

Scaricare l'Installer: Fate clic sul link per scaricare l'installer Python per il vostro sistema operativo.

Eseguire l'Installer: Una volta completato il download, eseguite l'installer. Assicuratevi di spuntare l'opzione "Aggiungi Python al PATH" durante l'installazione in modo che Python possa essere eseguito da qualsiasi posizione nel vostro computer.

Completare l'installazione: Seguite le istruzioni dell'installer per completare l'installazione di Python sul vostro computer. Dovreste vedere un messaggio di conferma una volta terminata l'installazione.

Una volta installato Python, potete aprire il terminale, o prompt dei comandi, io ora sono su windows e basta scrivere cmd nella barra di ricerca. digitare "Python" e premere invio. Vedrete apparire quello che viene chiamato

REPL, un ambiente che vi permette di eseguire il codice Python riga per riga, proprio come si faceva con i computer negli anni '80.

```
a = 10
b = 1
print(a+b)
```

Per uscire dal REPL possiamo scrivere exit() o premere CTRL-D e se digitate `python --version` o `python3 --version`, se è installato correttamente vi stamperà la versione corrente.

Sappiamo tutti più o meno cos'è un computer. Un computer è una macchina elettronica capace di eseguire una vasta gamma di operazioni, ma alla base di tutto ciò ci sono operazioni estremamente semplici. Queste operazioni elementari, o **elementi primitivi**, sono la chiave per capire come il computer elabora le informazioni.

Gli elementi primitivi sono i blocchi fondamentali su cui si basa la programmazione. Nella computer Science, o programmazione questi elementi primitivi sono noti anche come oggetti o tipi, e al momento quello che ci interessa sapere è che ne esistono quattro:

- Integers (interi): Rappresentano numeri interi, ad esempio, `5` o `-3`.
- Floats (virgola): Rappresentano numeri decimali, ad esempio, `3.14` o `2.0`.
- Booleans: Rappresentano i valori di verità `True` e `False`.
- Strings (stringhe): Rappresentano sequenze di caratteri, ad esempio, `"Hello, World!"`.

Per scoprire il tipo di un oggetto, puoi utilizzare la funzione `type()`. Ad esempio:

```
x = 5
print(type(x))  # Stampa <class 'int'>
```

I tipi o elementi primitivi possono essere usati per eseguire operazioni come:

- `+` (addizione): Somma due numeri.
- `-` (sottrazione): Sottrae un numero da un altro.
- `*` (moltiplicazione): Moltiplica due numeri.
- `/` (divisione): Divide un numero per un altro.
- `%` (modulo): Restituisce il resto della divisione tra due numeri.
- `**` (elevamento a potenza): Eleva un numero a una potenza.

```
a = 10
b = 3

sum_result = a + b  # 13
subtraction_result = a - b  # 7
multiplication_result = a * b  # 30
division_result = a / b  # 3.333...
modulo_result = a % b  # 1
power_result = a ** b  # 1000

# Operazioni tra tipi diversi
a = 10
b = 3.5

sum_result = a + b  # 13.5 (int + float)
subtraction_result = a - b  # 6.5 (int - float)
multiplication_result = a * b  # 35.0 (int * float)
division_result = a / b  # 2.857... (int / float)

# Conversione di tipi
x = 5.7
y = int(x)  # Converte float in int, y diventa 5

# Concatenazione di stringhe
string1 = "Hello"
string2 = " World!"
concatenated_string = string1 + string2  # "Hello World!"

# Moltiplicazione di stringhe
repeated_string = string1 * 3  # "HelloHelloHello"
```

Alcune di queste operazioni sono possibili perchè come vedremo sono definite e quindi python sa ad esempio come interpretare `string1 * 3`, ma non sa come interpretare `string1 * string1`!

Possiamo memorizzare i dati come ad esempio 3 all'interno di una struttura chiamate variabile. Non dobbiamo specificare se quella determinata variabile contiene un numero intero o con la virgola in Python e è perfettamente corretto cambiare il tipo di primitivo associato ad una variabile senza avere un errore.

```
name = "Alice"
type(name)
name = 25
type(name)
```

Ora puoi utilizzare queste variabili in espressioni e operazioni. Le variabili sono un modo utile per memorizzare dati e riferirsi ad essi con facilità.

Per visualizzare i risultati dei tuoi programmi, puoi utilizzare la funzione `print()`. Questa funzione consente di stampare testo e variabili sulla console. Ad esempio:

```
name = "Bob"
print("Hello, " + name + "!")
```

Questo stamperebbe "Hello, Bob!" sulla console.

In Python ci sono vari modi per formattare il testo e uno dei modi che trovo più intuitivo è il metodo dell'"f-string" che consiste nel mettere una f seguita da due apici includendo il contenuto di una variabile fra parentesi graffe, ad esempio:

print(f"Hello: {name}")

Per fare la parentesi graffa potete premere shift + alt grid + "[" per l'apertura della parentesi graffa, e shift+alt grid + "]" per chiuderla.

Usando questo sistema possiamo creare frasi che contengono molte variabili in modo molto semplice.

A questo punto creiamo una cartella io lo faccio sul desktop, ma voi potete farlo dove ritenete più opportuno. Dentro creiamo un'altra cartella che si chiamerà tut01 e creiamo un file di testo rinominandolo ex01.py

scriviamo dentro: print("hello world!")

Se apriamo il terminale di nuovo e scriviamo python ex01.py il file viene eseguito.

Ora che hai imparato i concetti di base di Python, perché non provi a scrivere il tuo primo script Python? scrivi un programma che calcola l'area di un cerchio. Metti in pausa il video e scrivi il programma in tranquillità ti ricordo che la formula per trovare l'area è 3.14*(raggio^2).

```
# Calcola l'area di un cerchio
pi = 3.14159
radius = 2.2
area = pi * (radius ** 2)
print(f"L'area del cerchio è: {area}")
```

A questo punto possiamo eseguire questo script e osservare che stampa l'area.

Avete imparato i concetti base di python, ora vi introduco il concetto di programmazione. Abbiamo visto che imparando la sintassi di python si possono scrivere programmi e che questi programmi risolvono problemi. I

problemi più difficili da risolvere però sono quelli dovuti al fatto che prima o poi dovrà essere usato da un utente.

Nella realtà quello che fa un programmatore è chiedere all'utente il raggio, questo però introduce una piccola difficoltà che ci farà usare tutte le conoscenze che abbiamo sviluppato finora.

```
# Calcola l'area di un cerchio
pi = 3.14159
radius = input("inserisci il raggio")
area = pi * (radius ** 2)
print("L'area del cerchio è:", area)
```

Come vedete il programma non funziona.

```
Traceback (most recent call last):
  File "/home/tedk/PycharmProjects/chatGptOpenSourceGUI/scratch/tutFoundamentals_01.py", line 6, in <module>
    area = pi * (radius ** 2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Proviamo a risolvere il problema introducendo il comando if.

Quando usiamo un comando tipo if quello che facciamo è una comparazione, quindi controlliamo se qualcosa è vero o falso, se è maggiore o minore di un certo numero o di una certa variabile e la sintassi per usarlo è if <comparazione>: segue una tabulazione che indica a python che i comandi successivi vengono eseguiti nel corpo dell'if.

In italiano suona un po' come se gli dicessi se radius è una stringa non fare niente, se è un numero fai l'operazione. In Python si scrive:

```
pi = 3.14159
radius = input("inserisci il raggio")
if type(radius) == int:
   area = pi * (radius ** 2)
   print(f"L'area del cerchio è:{area}" )
else:
   print(f"sei sicuro che: {radius} - sia un numero?")
```

Ora, se provo a far girare il programma mi posso facilmente accorgere che se scrivo hello world il programma non va in crash, ma fa la stessa cosa anche se scrivo 2.

scriviamo: A me risulta un {type(radius)}

Come vedete input, trasforma quello che abbiamo digitato in una stringa e la parte grossa che fa il programmatore è usare dei meccanismi che impediscano che queste cose accadano ed importante tanto quanto sapere come usare le variabili, perchè trovare errori nel codice e sapere cos'è che non funziona è una delle parti della programmazione che porta via più tempo!

Si può trasformare l'input in un numero in modo facile, scrivendo int(radius) ma se l'utente scrive Hello World il programma restituisce un errore.

Soluzioni?

Il modo migliore quindi è usare isdigit().

Si chiamano metodi e piano piano imparerete che ce ne sono un sacco e ci possono dare una mano a eseguire operazioni noiose sulle stringhe come ad esempio sapere se è un numero o meno. Se ad esempio ho una variabile string = "QUESTO E' UN TESTO MAIUSCOLO" posso scrivere print(string.lower()) e lo fa diventare questo e' un testo maiuscolo.

```
pi = 3.14159
radius = input("Inserisci il raggio: ")

if radius.isdigit():
   radius = float(radius)  # Converte la stringa in un intero
   area = pi * (radius ** 2)
   print("L'area del cerchio è:", area)
else:
   print(f"Sei sicuro che '{radius}' sia un numero?")
```

Un altro sistema, un pò più semplice per risolvere questo problema è usare try/except Exception.

Il senso di questo comando è questo, noi diciamo a Python guarda, prova a fare questa cosa, se non funziona fai questa.

```
pi = 3.14159
radius = input("Inserisci il raggio: ")

try:
   radius = float(radius)  # Prova a convertire la stringa in un intero
   area = pi * (radius ** 2)
   print("L'area del cerchio è:", area)
except Exception:
   print(f"Sei sicuro che '{radius}' sia un numero?")
```

Un'altra cosa di cui dobbiamo parlare è questa. Prendiamo una calcolatrice. Avrebbe la stessa utilità se ad esempio dopo ogni operazione che fate, si spegnesse e per fare un'altra operazione doveste accenderla di nuovo?

Un altro concetto chiave della programmazione sono i loop. Un loop è qualcosa che si ripete fino a quando non succede qualcosa come ad esempio ho finito i cerchi e non mi serve più calcolare l'area.

Per creare un loop posso utilizzare il comando while che significa letteralmente fino a quando e di solito lo utilizziamo accoppiato con una variabile a cui associamo un valore vero o falso.

Se scrivo while True: e una lista di operazioni il programma continua a girare all'infinito. Ma se creo una variabile isWannaContinue = True, mentre controllo che il programma ha scritto un numero, posso controllare ad esempio che se ha scritto exit interrompo il loop facendo diventare isWannaContinue = False.

```
pi = 3.14159
isWannaContinue = True

while isWannaContinue:
    radius = input("Inserisci il raggio: (o 'exit' per uscire):
")
    if radius == "exit":
        isWannaContinue = False
    else:
        try:
            radius = float(radius)
            area = pi * (radius ** 2)
            print("L'area del cerchio è:", area)
        except Exception:
            print(f"Sei sicuro che '{radius}' sia un numero?")
            continue
exit()
```

Come vedete il ciclo while continua fino a che l'utente non scrive exit, solo a quel punto esegue l'istruzione successiva. Se l'operazione è valida esegue l'operazione, nel caso in cui ci sia un errore, stampa la nostra frase di controllo e quindi fa ripartire il ciclo con continue.

Non vi preoccupate se ancora non è tutto chiaro. L'unico modo è scrivere, scrivere, scrivere e mettervi alla prova. Provate ad esempio a creare un programma che calcola l'area di un rettangolo, il volume di un cubo o che calcoli il teorema di pitagora!

Bene siamo giunti al termine di questo tutorial. Oltre a ricordarvi che in descrizione trovate il link dove trovare i codici che abbiamo usato nel tutorial, se non lo avete fatto vi invito a iscrivervi al canale e a mettere un like.

GoodByte! Alla prossima