Python Dev Course

Email di *Giacomo Usai*: giacomo.usai.iic97@gmail.com

Tutorial su come installare Python:
https://www.youtube.com/watch?v=Kn1HF3oD19c

Sito ufficiale Python dove è possibile scaricare l'ultima versione sul proprio pc:
https://www.python.org/downloads/

Sito ufficiale JetBrains dove scaricare la versione Community di PyCharm per poter sviluppare il proprio codice:
https://www.jetbrains.com/pycharm/download/#section=windows

Nota. PyCharm per funzionare deve avere una versione di Python installata sul PC.

Risorse utili dove esercitarsi: i seguenti siti in inglese sono molto validi e contengono diverse categorie a seconda degli argomenti che caratterizzano il linguaggio. In ognuno sono presenti esercizi con le soluzioni.
https://www.w3schools.com/python/
https://www.geeksforgeeks.org/python-programming-language/

Consiglio di scaricare questo libro molto utile, anche un pò divertente, per tutti gli studenti appassionati di informatica.
https://drive.google.com/file/d/1tMllvuRregMCwwYb-c25DlhJ7FxsPaM8/view
Vengono proposti quesiti di varia natura e analizzate le interessanti soluzioni che si possono raggiungere grazie all'informatica.
Per qualsiasi approfondimento rimango a disposizione!

Lezione 1

1. `print("Hello, World!")`
2. indentations, cioè "*identazione*" del codice usando il tasto TAB della tastiera
3. Comment: su una riga (#) o multiriga (""")
4. Variabili
    a. string, int
    b. caso "-"
    c. multi assegnazione su una riga
5. keyword **type**()
    a. x = 5
    b. x = "Hello World"
    c. x = 20.5

d. x = ["apple", "banana", "cherry"]

e. x = True

f. print(type(x))

6. Definire una funzione:

    a. `def my_function(x):`

Nella Prima lezione abbiamo introdotto il linguaggio Python. Come installarlo sul proprio PC e come scaricare l'ambiente di sviluppo per poter scrivere del codice (PyCharm).
Abbiamo visto la funzione ***print*** che consente di stampare a video il contenuto "di qualcosa": una stringa, un numero o sostanzialmente qualsiasi informazione siamo interessati, basta appunto specificare cosa vogliamo che ***print*** stampi passandogli questa informazione all'interno delle parentesi tonde.
Questo meccanismo di mettere all'interno delle parentesi tonde il nome di una variabile è molto utile e verrà usato parecchio.
Infine abbiamo brevemente visto come definire una ***funzione***: cioè una procedura in grado di prendere informazioni in input, elaborarle, e fornire un risultato in output. Questo risultato dipende da come e cosa vogliamo che la funzione restituisca. Un esempio è la funzione concatenazione di due stringhe: quindi una funzione che prende in input 2 stringhe e restituisce la loro unione (es. input="Ci", "ao"; avremo come output="Ciao", cioè l'unione della prima stringa con la seconda).
In Python scriveremo:

```
def concatenazionestringhe(x, y):
    print(x + y)
```

Come possiamo notare, l'operatore "+" ci consente di fare la *concatenazione* (cioè unire) due stringhe.
Nel caso in cui invece di due stringhe vengono forniti due numeri interi (es. 2 e 3), verrà eseguita la loro somma. Quindi lo stesso

operatore “+” si comporta in modo diverso a seconda dell’input che Noi diamo: nel caso di due numeri farà la somma, nel caso di due stringhe farà la concatenazione/unione delle due stringhe.

Compito 1

- Collegarsi al sito https://www.w3schools.com/python/exercise.asp?filename=exercise_syntax1
  e svolgere esercizi su *Syntax*, *Comments*, *Variables*, *Data Types* e *Numbers*.
- scrivere una funzione che prende come argomento 3 numeri e stampa la loro somma a video.
- scrivere una funzione che prende come argomento 3 stringhe e stampa le due stringhe concatenata tra loro.

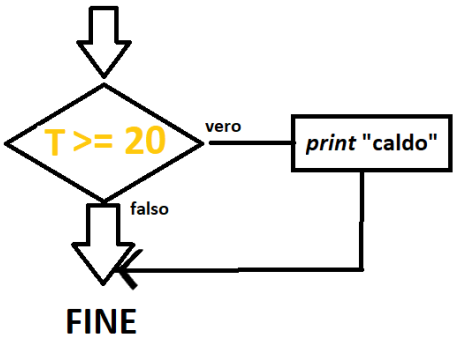
Lezione 2

7. ripasso lezione precedente e compiti

8. Definire una **funzione**:
    a. a cosa serve? quando è conveniente crearne una?
    b. funzioni primitive di Python
9. Python *Keywords*: cioè le parole speciali. Abbiamo visto **def** (per creare una funzione), **IF**, **True**… e ne vedremo molte altre. (Qui trovi l'elenco completo https://www.w3schools.com/python/python_ref_keywords.asp)
10. *Namespaces* and *Scope* in Python
    a. posso chiamare due variabili con lo stesso nome?
    b. posso chiamare una variabile con lo stesso nome di una keyboard? Cioè creare una variabile che si chiama ad esempio *True*?
11. cos'è il **MAIN()**? Perché è importante?
12. controllo: **IF**
    a. https://www.w3schools.com/python/python_conditions.asp
13. cicli: **FOR**
    a. https://www.w3schools.com/python/python_for_loops.asp
14. **input** da console utente

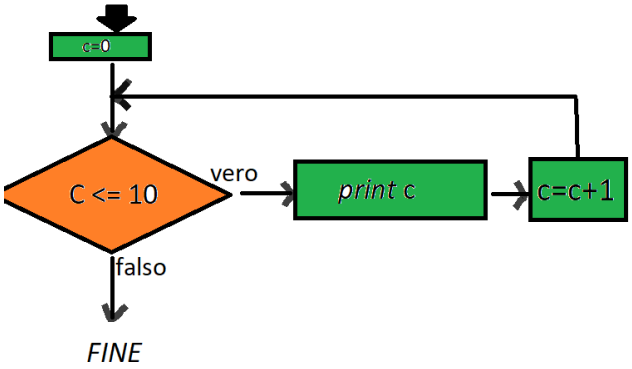
Riassunto Seconda lezione:
il **main()** è una parte presente in tutti i programmi: le parentesi dopo il main cosa indicano? Parentesi tonde, indicano una *funzione*.
Un programma può contenere una o più funzioni, una delle quali deve essere **main**. Perchè è obbligatorio avere una funzione **main**? Perchè ogni programma comincia eseguendo la funzione **main**, è pertanto il nostro punto di partenza, un riferimento universale.

Principio di funzionamento del controllo **IF**: l'immagine seguente mostra sia il flow chart (diagramma di flusso) sia relativo codice python.



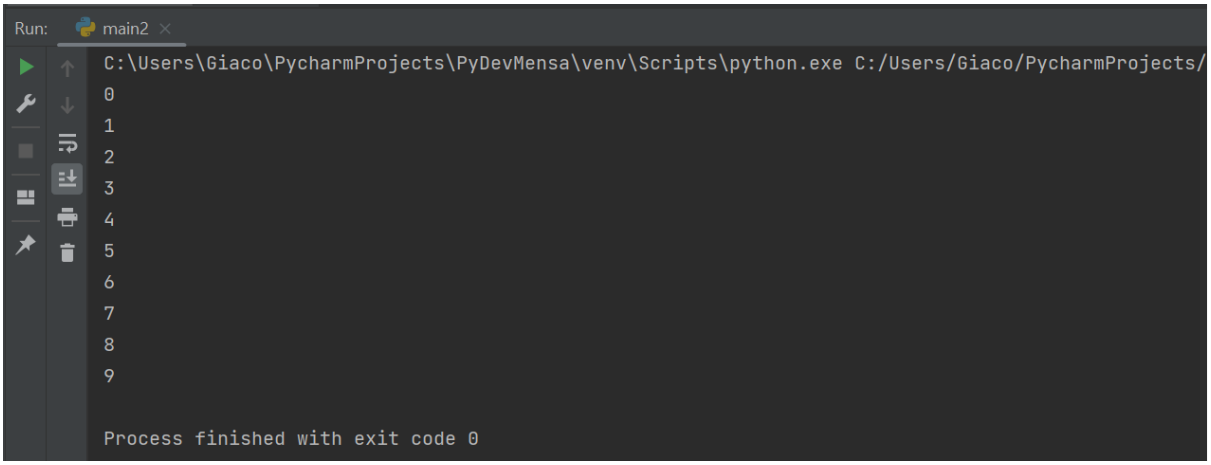

```
if (T >= 20):
    print("caldo")
```

Principio di funzionamento del comando **FOR:** l'immagine mostra sia il flow chart (diagramma di flusso) sia relativo codice python.




```
for c in range(10):
    print(c)
```

Eseguendo questo codice python, si ottiene una stampa in sequenza dei numeri da 0 a 9, come mostrato in figura:



Compito 2

- Abbiamo visto il ciclo FOR per stampare una sequenza di numeri da 0 a 9.
    - Sapresti modificarlo in modo che stampi partendo da 5?
    - Sapresti modificarlo in modo che stampi da 1 a 100?
- In matematica, il fattoriale di un numero è il prodotto di tutti i suoi predecessori fino ad 1: quindi ad esempio il fattoriale di 5 si indica con 5! e sarà 5! = 5*4*3*2*1 = 120.

  Completare il seguente codice python, sostituendo i “???”, in modo che esegua il fattoriale di 5:

```
def funzione_fattoriale():
    fattoriale = 1
    for c in range(2, ???):
        print(c)
        fattoriale = c * ???
    print(fattoriale)
```

Lezione 3

15. ripasso lezione precedente e compiti
16. implementazione della funzione fattoriale di un numero preso da tastiera
    a. usando la funzione *input*()
17. *Serie di Fibonacci* e ciclo **while**:
18. **Sequenze** (https://www.geeksforgeeks.org/python-lists/?ref=lbp)
    a. esempio: A = [23, 302, 9]
    b. cosa sono?
    c. come si accede ad un elemento?
19. Funzione **len**() per conoscere la lunghezza della sequenza
20. Funzione **append**() per inserire in coda alla sequenza
21. Funzione **remove**() per rimuovere un elemento (uno o più) dalla sequenza

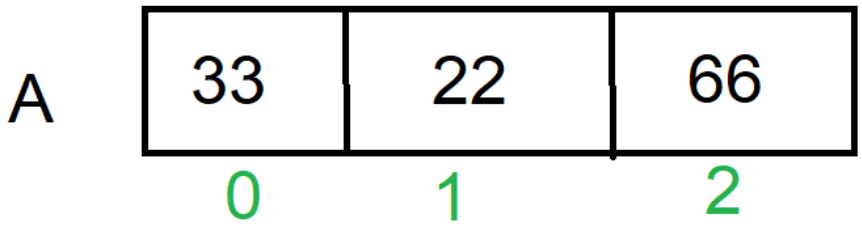
Riassunto Terza lezione:

Si noti che l'indice iniziale è zero (0) e poi così via 1,2!!

Quindi se definisco una sequenza di 3 numeri:

A = [33, 22, 66]

La seguente immagine aiuta a comprendere la logica con cui si accede ad una Sequenza in Python:



A[0] = 33
A[1] = 22
A[2] = 66

La seguente tabella riassume le principali operazioni che possiamo effettuare su una sequenza (in questo caso chiamata "s").

Operation	Result	Notes
`s[i] = x`	item *i* of *s* is replaced by *x*	
`s[i:j] = t`	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*	
`del s[i:j]`	same as `s[i:j] = []`	
`s[i:j:k] = t`	the elements of `s[i:j:k]` are replaced by those of *t*	(1)
`del s[i:j:k]`	removes the elements of `s[i:j:k]` from the list	
`s.append(x)`	appends *x* to the end of the sequence (same as `s[len(s):len(s)] = [x]`)	
`s.clear()`	removes all items from *s* (same as `del s[:]`)	(5)
`s.copy()`	creates a shallow copy of *s* (same as `s[:]`)	(5)
`s.extend(t)` or `s += t`	extends *s* with the contents of *t* (for the most part the same as `s[len(s):len(s)] = t`)	
`s *= n`	updates *s* with its contents repeated *n* times	(6)
`s.insert(i, x)`	inserts *x* into *s* at the index given by *i* (same as `s[i:i] = [x]`)	
`s.pop()` or `s.pop(i)`	retrieves the item at *i* and also removes it from *s*	(2)

Inoltre abbiamo visto come calcolare la Serie di Fibonacci e il ciclo **while**. Il seguente codice è quello implementato nell'esempio pratico e come possiamo vedere riceve un numero n che determina quanto andrà avanti in ciclo while: cioè fintanto che la nostra variabile a è minore di n. Fintanto che la condizione è vera il programma eseguirà le operazioni all'interno del ciclo while, infine quando la condizione non sarà più vera, quindi avremo a>=n, esco dal ciclo **while** e stampo il risultato.

```
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
fib(1000)
```

Compito 3

- Definire una funzione che somma tutti gli elementi della seguente sequenza:
  - A = [10, 5, 2, 3, 50, 20, 10]
- Esercitarsi sul sito: https://www.w3schools.com/python/exercise.asp?filename=exercise_lists1
  - sono presenti anche le soluzioni!
- *Difficile*: Definire una funzione che riceve in input 3 numeri inseriti da tastiera dall'utente e li salva all'interno di una sequenza.
  - aiuto 1: per ricevere il numero da tastiera usare il codice visto a lezione
    ```
    a = int(input('inserisci numero: '))
    ```
  - aiuto 2: per salvare il numero (a) all'interno della sequenza bisogna crearla prima di richiedere l'input, e usare la funzione "append()" che inserisce in fondo alla sequenza.
  - aiuto 3: è utile usare un ciclo **for** per ripetere 3 volte la stessa procedura: cioè chiedo il numero in input all'utente e poi lo salvo nella sequenza, ripeto per 3 volte.
    Prendi spunto dal seguente codice, è un ciclo **for** che stampa per 3 volte la lettera "s":
    ```
    for i in range(3):
        print("s")
    ```

Lezione 4

22. ripasso sequenze

23. funzione **append**
24. funzione **pop**
25. funzione **remove**
26. format
    a. per formattare una stringa in output (console): di fatto sostituisce le graffe con la variabili che passiamo alla funzione format:

```
a=3223
print("{0} = {1}".format("ciao", a))
```

risultato: stampa “ciao = 3223”

Riassunto Quarta lezione:

Con il seguente codice che definisce una sequenza, controllo se il numero 3 è all’interno…usando la keyword **in**.

```
myList = [1,2,3,4,5]
if 3 in myList:
    print("3 is present")
```

Dopo aver visto come definire una sequenza, in questo caso chiamata A (lettera maiuscola!), abbiamo introdotto le funzioni:

- **pop**
    - Remove and return item at index (default last)
    - Rimuove e ritorna l’elemento all’indice specificato (di default rimuove dalla coda/fondo della lista)
- **append**
    - Append object to the end of the list.
    - Aggiunge elemento in coda alla lista.
- **remove**
    - Remove first occurrence of value.
      Raises ValueError if the value is not present.
      Rimuove prima occorrenza del valore specificato. Genera errore se il valore non è presente all’interno della

sequenza (e quindi non trova nessun elemento specificato da rimuovere).

```
A = [10, 5, 2, 3, 50, 20, 10]
A.pop()
A.append(30)
print(A)
```

Compito 4

- Buon Natale!!!

```
# Function to draw a Christmas tree with a given height
def draw_tree(height):
  # Loop through each row of the tree
  for i in range(1, height + 1):
    # Print the spaces before the asterisks on each row
    for j in range(height - i):
      print(" ", end="")

    # Print the asterisks on each row
    for j in range(2 * i - 1):
      print("*", end="")

    # Move to the next line
    print()

# Call the function to draw a tree with a height of 5
draw_tree(5)
```

Lezione 5 - 2023

27. concetto di programmazione ad oggetti: classe, oggetto, attributi di un oggetto.
28. creazione **classe** persona
    a. modifica attributi e chiamate a metodi della classe
29. I **files** in Python
30. funzione **open()**

Riassunto Quinta lezione:
In Python circa ogni cosa è un oggetto, con le sue proprietà ed i suoi metodi. Una classe può essere vista come un "costruttore di oggetti", cioè un modo che consente allo sviluppatore di creare un nuovo oggetto potendo scegliere i suoi attributi (ad esempio, nel nostro codice di esempio visto a lezione, nome ed età per la classe Persona).

Per operare su un file è necessario conoscere la funzione **open()**.
Esistono 4 modalità con cui si può aprire un file:

1. "r" - Lettura
2. "a" - Append
3. "w" - Scrittura - Apre il file in lettura, crea il file se non esiste.
4. "x" - Creazione nuovo file

```
f = open("giacomo.txt", "x") # crea file
```

Lezione 6 - 2023

31. ripasso funzione **open()** per gestire i file
32. nuova keyword per gestire file: **with-as**

Riassunto Sesta lezione:

```
with open('C:/Users/Giaco/Documenti/ddd.txt', "w", encoding="utf-8") as f:
    f.write("this is a test")
```

il codice sopra elencato consente di fare…che cosa?

open()? write()? with?

Risposta: consente di aprire il file "ddd.txt" in scrittura perchè viene specificata la "w" nella open(). Dopo averlo aperto in scrittura, usando la write() scriviamo al suo interno "this is a test". Se il file aveva già del contenuto (testo) al suo interno questo viene sovrascritto con il nostro nuovo testo perché stiamo facendo una write. Se avessimo voluto scrivere partendo dalla fine, senza cancellare il contenuto del file, dovevamo fare una append("a")!!

Infine non dobbiamo preoccuparci di fare la close() del nostro file, che ci consente di chiuderlo, dato che usando la keyword **with-as** sarà proprio questa ad eseguire la chiusura terminato il blocco di operazioni che abbiamo scritto.

Compito 6

Abbiamo visto molte operazioni per gestire il file. Questo compito richiede di sviluppare del codice che fa le seguenti operazioni in sequenza (una dopo l'altra):

1. *crea* un file chiamato "leggimi.txt",
2. *apre* il file in *scrittura* e *scrive* al suo interno i tuoi dati personali "nome, cognome e data nascita"
3. dopo aver scritto, fa una ***print***() per stampare a video il contenuto del file
4. infine *chiude* ed *elimina* il file creato, quindi il file verrà cancellato dal disco.

funzioni utili viste a lezione:

- **open**(),
- **write**(),
- **close**() o usare il **with-as**
- **import os**
  os.remove("nomefile.txt")

Lezione 9 - 2023

Lezioni sulle basi di dati

Una base di dati (in inglese, ***database***): dal punto di vista logico, è un insieme di
tabelle (formate da una o più colonne), di solito "collegate" in qualche modo tra loro.

- **Tabella** = è un ***insieme di righe*** (n-uple o tuple, dall'inglese tuple) dello stesso formato (istanze di uno stesso tipo, strutturato in campi);

si noti il termine "*insieme*": sotto l'aspetto semantico (cioè del significato), non sono importanti né l'ordine delle righe né eventuali ripetizioni ossia duplicati di una
stessa riga (due o più righe identiche equivalgono a una sola, il significato è esattamente lo stesso).

SQL (**Structured Query Language**) è un linguaggio standardizzato per database, progettato per le seguenti operazioni:

- creare e modificare schemi di database;
- inserire, modificare e gestire dati memorizzati;
- interrogare i dati memorizzati: **query**.

https://www.w3schools.com/sql/

SQL è un linguaggio per interrogare e gestire basi di dati mediante l'utilizzo di costrutti di programmazione denominati query. Con SQL si leggono, modificano, cancellano dati e si esercitano funzioni gestionali ed amministrative sul sistema dei database.