Es. 1 (introduzione a Python, le basi)

February 12, 2024

1 LE BASI DI PYTHON - IL MIO PRIMO PROGRAMMA

Questo breve codice utilizza il comando print per mostrare il messaggio "Hello, World!" a schermo. In Python, il print è il comando utilizzato per visualizzare testo o risultati durante l'esecuzione di un programma. Nel caso specifico, il programma stampa semplicemente un saluto di base. L'Hello World è il programma esempio "base" di un qualsiasi linguaggio di programmazione; infatti, come in questo caso, è il primo codice che si apprende.

[2]:
```
#Comprendere il comando print
print("Hello, World!")#il comando print stampa un testo indicato nelle␣
↪virgolette, quello spazio è inteso come una stringa di testo
```

```
Hello, World!
```

Questo codice utilizza il comando print per mostrare il contenuto della variabile nome. Il programma assegna la stringa "Matteo" alla variabile nome, la stringa non è altro che una frase di testo normale in linguaggio "umano" come è per l'appunto la parola "Matteo" che è una parola del linguaggio "umano" che viene assegnata come dato ad una variabile ed è una stringa. Successivamente, il programma utilizza il comando print per visualizzare il messaggio "Il nome esempio è:" seguito dal contenuto della variabile nome. Questo significa che verrà stampato a schermo "Il nome esempio è: Matteo", ovviamente la parola "Matteo" è a capo perchè i due print sono in due righe diverse nel codice e questo fa sì che il risultato finale venga "printato" in due righe separate. In sintesi, il comando print può essere utilizzato anche per visualizzare il contenuto di variabili, come il nome nella variabile "nome", o di qualsiasi altro dato.

[1]:
```
#Comprendere il comando print e comprendere come definire una variabile con una␣
↪stringa
nome="Matteo"
print("Il nome esempio è:")
print(nome)
```

```
Il nome esempio è:
Matteo
```

2 LE VARIABILI E L'INPUT UTENTE

Questo codice utilizza il comando input per acquisire dati dall'utente. Ecco una breve spiegazione di come funziona: Il programma per prima cosa chiede all'utente di inserire il proprio nome utilizzando

il comando input, l'input viene quindi memorizzato come un dato nella variabile nome. Successivamente, viene richiesto all'utente di inserire il cognome utilizzando l'altro input, quest'ultimo a quel punto viene sempre memorizzato come un dato nella variabile cognome. Infine, il programma stampa un messaggio di benvenuto utilizzando il comando print("Benvenuto su Jupyter Notebook", nome, cognome), dove nome e cognome sono i dati inseriti dall'utente poichè sono le variabili e quindi il programma stamperà direttamente i loro valori, cioè i dati che aveva immesso prima l'utente nell'input. Questo codice è un esempio semplice ma utile di come interagire con l'utente attraverso l'input e di come salvare i dati corretti nelle variabili.

[1]:
```
#Comprendere il comando input
nome=input("Inserisci il tuo nome: ")#l'input "prende" un dato dall'utente e lo␣
 ↪salva temporaneamente nella RAM (Random Access Memory)
cognome=input("Inserisci il tuo cognome: ")#le variabili si possono indicare␣
 ↪come si vogliono però è meglio, per se stessi e per gli altri, che␣
 ↪visualizzeranno il codice usare dei nomi sensati per ogni occasione
print("Benvenuto su Jupyter Notebook",nome,cognome)#dentro il print ci sono due␣
 ↪input dettati dall'utente che specificano i dati
```

```
Inserisci il tuo nome: Matteo
Inserisci il tuo cognome: Magrino
Benvenuto su Jupyter Notebook Matteo Magrino
```

Il codice qui sotto chiede all'utente di inserire il nome completo della via e l'indirizzo di casa e successivamente stampa il messaggio confermando la via inserita. Questo codice è identico strutturalmente a quello di sopra perchè chiede sempre un informazione all'utente via input, solo che questo è un esempio diverso

CURIOSITÀ: la via 20 W 34th St., New York, NY 10001, Stati Uniti è quella dove risiede una delle due facciate laterali lunghe del famossisimo grattacielo "Empire State Building". Link alla pagina di Google Maps: https://bit.ly/googlemapsempirestatebuilding

[1]:
```
indirizzoviadicasa=input("Inserisci il nome completo della via, nonchè␣
 ↪l'indirizzo, di casa tua: ")
print("Hai inserito la via:",indirizzoviadicasa)
```

```
Inserisci il nome completo della via, nonchè l'indirizzo, di casa tua: 20 W 34th
St., New York, NY 10001, Stati Uniti
Hai inserito la via: 20 W 34th St., New York, NY 10001, Stati Uniti
```

3 CONCATENAZIONI E MOLTIPLICAZIONI CON LE STRINGHE DI TESTO

Il codice "Matteo" + "Magrino" esegue la concatenazione di due stringhe, "Matteo" e "Magrino", producendo così una nuova stringa risultante dalla fusione delle due. La concatenazione delle stringhe è un'operazione molto comune in programmazione, utile per unire più stringhe in una sola e fare così un output unico.

[1]:
```
"Matteo"+"Magrino"
```

[1]:
```
'MatteoMagrino'
```

Il codice qui sotto è identico a quello precedente ma non è presente il simbolo "+", infatti la concatenazione delle stringhe si può creare anche senza il simbolo +, questo perchè in Python se le due stringhe si trovano una accanto all'altra senza un operatore tra di esse, vengono automaticamente concatenate. Questo è noto come "concatenazione implicita delle stringhe".

[7]:
```
"Matteo""Magrino"
```

[7]:
```
'MatteoMagrino'
```

In Python, moltiplicare una stringa per un numero intero produce una nuova stringa che consiste nella stringa originale ripetuta per un certo numero di volte. Quindi, l'espressione "Matteo" * 30 restituirà come output una stringa contenente il valore della stringa "Matteo" ripetuto 30 volte. Questo approccio è utile quando si desidera creare rapidamente una stringa ripetuta più volte senza doverla scrivere esplicitamente più volte nel codice, rendendo il codice più conciso, efficiente e veloce da scrivere. Ad esempio, se volessimo creare una stringa che contenga il nome "Matteo" ripetuto 30 volte, sarebbe più efficiente e conciso scrivere "Matteo" * 30 piuttosto che scrivere manualmente "Matteo" 30 volte, inoltre il codice occuperebbe più memoria. Questo approccio è particolarmente utile quando si desidera creare stringhe di grandi dimensioni o ripetere un certo pattern (ripetività) all'interno di una stringa.

[13]:
```
"Matteo"*30
```

[13]:
```
'MatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoM
atteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMat
teoMatteoMatteoMatteo'
```

Questo codice è un riassunto di quelli precedenti riguardanti le stringhe di testo. L'espressione "Matteo" * 20 + "Magrino" * 50 combina due operazioni: la moltiplicazione di una stringa per un numero intero e la concatenazione di stringhe. Nel dettaglio, "Matteo" * 20 crea una nuova stringa in cui la parola "Matteo" viene ripetuta per 20 volte, mentre "Magrino" * 50 produce una stringa in cui la parola "Magrino" viene ripetuto per 50 volte. Infine, entrambe le stringhe risultanti vengono concatenate insieme utilizzando l'operatore +, producendo come output finale una nuova stringa che inizia con "Matteo" ripetuto 20 volte seguito poi da "Magrino" ripetuto 50 volte. In sintesi, l'output di questa espressione sarà una lunga stringa che contiene la ripetizione di "Matteo" per 20 volte seguita dalla ripetizione di "Magrino" per 50 volte. Un output a dir poco particolare...

[17]:
```
"Matteo"*20+"Magrino"*50
```

[17]:
```
'MatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoMatteoM
atteoMatteoMatteoMatteoMatteoMatteoMagrinoMagrinoMagrinoMagrinoMagrinoMagr
inoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrino
MagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMag
rinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrin
oMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrinoMagrino'
```

4 LA CONCATENAZIONE DI VARIABILI

In questo codice viene eseguita la concatenazione di variabili per creare una stringa più complessa e con tutte le informazioni necessarie all'utente. La concatenazione di variabili consiste nell'unire più variabili, in questo caso definite come stringhe di testo, per formare un'unica nuova stringa finale. Nel caso specifico, le variabili "nome", "cognome" ed "età" contengono rispettivamente il nome, il cognome e l'età di una persona esempio. Queste variabili vengono concatenate insieme utilizzando l'operatore di concatenazione "+" e del testo aggiuntivo per formare un messaggio più completo e più facile alla lettura da parte dell'utente.

[24]:
```
nome="Matteo"
cognome="Magrino"
età="15"
messaggiodastampare="Ciao a tutti mi chiamo "+nome+" "+cognome+", ho "+età+"␣
 ↪anni e sono un programmatore in erba di Python e altri linguaggi di␣
 ↪programmazione"
print(messaggiodastampare)
```

```
Ciao a tutti mi chiamo Matteo Magrino, ho 15 anni e sono un programmatore in
erba di Python e altri linguaggi di programmazione
```

5 IL CICLO FOR

Il programma sottostante acquisisce il nome e il cognome dell'utente come input, così come il numero di volte che l'utente desidera ripetere un messaggio gentile. Utilizza poi un ciclo for per iterare attraverso il numero di volte specificato dall'utente, gestito dal contatore "i" (che sta per iteratore ma si può chiamare come si vuole). In ogni iterazione del ciclo, il programma stampa un messaggio gentile che include il nome e il cognome inseriti dall'utente. La parte chiave del programma è ovviamente il ciclo for, che utilizza un contatore o iteratore (i due nomi sono sinonimi) "i" per iterare attraverso un numero di volte pari a quello specificato dalla variabile numeroripetizionimessaggio. In ogni iterazione, il programma stampa un messaggio gentile che include il nome e il cognome inseriti dall'utente. Il contatore "i" è un vero e proprio iteratore, che tiene traccia dello stato attuale del ciclo e consente di eseguire azioni specifiche per ciascuna iterazione. La parte del range(numeroripetizionimessaggio) fornisce una sequenza di numeri da 0 a numeroripetizionimessaggio-1, definendo così il numero di iterazioni.

[1]:
```
#Comprendere il ciclo for
nome=input("Inserisci il tuo nome: ")
cognome=input("Inserisci il tuo cognome: ")
numeroripetizionimessaggio=int(input("Inserisci il numero di volte che desideri␣
 ↪che un messaggio gentile si ripeta: "))#int è l'acronimo di integer, cioè di␣
 ↪intero e vuole dire che la variabile in questo caso è di tipo intero e può␣
 ↪"leggere" solo numeri interi (cioè tutti quelli senza virgola) o altro
#For è un contatore o iteratore perchè tiene traccia dello stato attuale del␣
 ↪ciclo (per spiegazione più dettagliata su come funziona leggere il riquadro␣
 ↪sopra)
for i in range(numeroripetizionimessaggio):#il numero di volte che viene␣
 ↪ripetuto questo messaggio dipende dall'input messo precedentemente dall'utente
```

```
    print("Che bel nome e cognome che hai",nome,cognome,"è davvero molto bello")
```

```
Inserisci il tuo nome: Matteo
Inserisci il tuo cognome: Magrino
Inserisci il numero di volte che desideri che un messaggio gentile si ripeta: 12
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
Che bel nome e cognome che hai Matteo Magrino è davvero molto bello
```

6 ESEMPI DI CALCOLATRICE

Il codice inizia con un messaggio di benvenuto, invitando l'utente a utilizzare la calcolatrice per effettuare addizioni. Successivamente, attraverso il comando input, si richiede all'utente di inserire due numeri, che vengono poi convertiti in formato float per consentire la lettura da parte del programma e l'immisione di numeri decimali da parte dell'utente. L'operazione di addizione tra i due numeri viene eseguita utilizzando l'operatore + (più), il risultato viene memorizzato nella variabile addizione, e infine, il risultato viene visualizzato con un messaggio appropriato. **N.B.: la virgola nei numeri decimali in Python quando vengono immessi come input, come nella maggior parte dei linguaggi, DEVE sempre essere inserita con un punto e non con una virgola questo perchè per convenzione si è scelto di assegnare il punto come separatore decimale**.

[3]:
```
#Comprendere come eseguire le addizioni
print("Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le␣
 ↪addizioni!")
numero1=float(input("Inserisci il primo numero: "))#float indica che la␣
 ↪variabile può leggere i numeri reali, cioè tutti quei numeri inclusi quelli␣
 ↪con la virgola o senza
numero2=float(input("Inserisci il secondo numero: "))
addizione=numero1+numero2
print("La somma dei due numeri è pari a:",float(addizione))
```

```
Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le addizioni!
Inserisci il primo numero: 5.7
Inserisci il secondo numero: 3.2
La somma dei due numeri è pari a: 8.9
```

Il codice qui sotto svolge l'operazione di sottrazione tra due numeri (sempre float) inseriti dall'utente. Funziona allo stesso modo ed infatti ha una identica struttura tranne per il fatto che c'è il simbolo

- (meno) che fa eseguire corretamente l'operazione in questione.

[2]:
```
#Comprendere come eseguire le sottrazioni
print("Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le␣
 ↪sottrazioni!")
numero1=float(input("Inserisci il primo numero: "))
numero2=float(input("Inserisci il secondo numero: "))
sottrazione=numero1-numero2
print("La sottrazione dei due numeri è pari a:",float(sottrazione))
```

```
Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le sottrazioni!
Inserisci il primo numero: 3.4
Inserisci il secondo numero: 8.2
La sottrazione dei due numeri è pari a: -4.799999999999999
```

Il codice qui sotto svolge l'operazione di moltiplicazione tra due numeri (sempre float) inseriti dall'utente. Funziona allo stesso modo ed infatti ha una identica struttura tranne per il fatto che c'è il simbolo * (asterisco) che in programmazione, non solo in Python ma anche nella maggiorparte dei linguaggi, vuole significare il simbolo per ($\times$) che fa eseguire corretamente l'operazione in questione.

[3]:
```
#Comprendere come eseguire le moltiplicazioni
print("Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le␣
 ↪moltiplicazioni!")
numero1=float(input("Inserisci il primo numero: "))
numero2=float(input("Inserisci il secondo numero: "))
moltiplicazione=numero1*numero2#il simbolo * vuol dire moltiplicazione
print("La moltiplicazione dei due numeri è pari a:",float(moltiplicazione))
```

```
Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le
moltiplicazioni!
Inserisci il primo numero: 3.9
Inserisci il secondo numero: 9.1
La moltiplicazione dei due numeri è pari a: 35.489999999999995
```

Il codice qui sotto svolge l'operazione di divisione tra due numeri (sempre float) inseriti dall'utente. Funziona allo stesso modo ed infatti ha una identica struttura tranne per il fatto che c'è il simbolo / (barra o slash in inglese) che in programmazione, non solo in Python ma anche nella maggiorparte dei linguaggi, vuole significare il simbolo della divisione (cioè il diviso) (: oppure il $\div$) che fa eseguire corretamente l'operazione in questione. Anche se si può dire che in algebra o semplicemente in matematica il simbolo / viene usato per scrivere le frazioni, che sono comunque delle divisoni a tutti gli effetti.

[1]:
```
#comprendere come eseguire le divisioni
print("Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le␣
 ↪divisioni!")
numero1=float(input("Inserisci il primo numero: "))
numero2=float(input("Inserisci il secondo numero: "))
divisione=numero1/numero2#il simbolo / vuol dire divisione
print("La divisione dei due numeri è pari a:",float(divisione))
```

```
Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le divisioni!
Inserisci il primo numero: 5.7
Inserisci il secondo numero: 2.6
La divisione dei due numeri è pari a: 2.1923076923076925
```

Il codice qui sotto svolge l'operazione di potenza tra due numeri (la base come float e l'esponente come int) inseriti dall'utente. Funziona allo stesso modo ed infatti ha una identica struttura tranne per il fatto che c'è il simbolo ** (doppio asterisco, che letteralmente vorrebbe dire doppio ×) che in programmazione, non solo in Python ma anche nella maggiorparte dei linguaggi, vuole significare il simbolo della potenza (xˆy, come il ** se fosse il ˆ) che fa eseguire corretamente l'operazione in questione.

[5]:
```
#Comprendere come eseguire le potenze
print("Ciao, attraverso questa calcolatrice puoi calcolare il valore di una␣
 ↪potenza!")
base=float(input("Inserisci la base della potenza: "))#la base è un float,␣
 ↪questo perchè si può fare la potenza di un numero decimale
esponente=int(input("Inserisci l'esponente della potenza: "))#l'esponente è un␣
 ↪int invece, questo perchè per fare un esempio più facile è meglio usare l'int␣
 ↪in modo che il risultato sia più semplice
potenza=base**esponente#il simbolo ** vuol dire potenza
print("Il valore della potenza è pari a:",float(potenza))
```

```
Ciao, attraverso questa calcolatrice puoi calcolare il valore di una potenza!
Inserisci la base della potenza: 5.7
Inserisci l'esponente della potenza: 3
Il valore della potenza è pari a: 185.193
```

Questo codice riassume tutti i precedenti in un'unico codice. Il codice infatti rappresenta una calcolatrice che consente all'utente di eseguire operazioni di addizione, sottrazione, moltiplicazione, divisione e calcolo delle potenze. Questa versatilità è ottenuta attraverso l'utilizzo di condizioni (che in ordine gerarchico sono: if, elif ed else) che dirigono il flusso del programma in base all'operazione scelta dall'utente. Inizialmente, il programma richiede all'utente di inserire il tipo di operazione che desidera effettuare sottoforma di segno, nonché poi i due numeri coinvolti. L'input dell'utente viene successivamente convertito in numeri in virgola decimale (i cosidetti float) per consentire l'elaborazione di numeri decimali. La parte principale del codice è pero costituita dalle condizioni if, elif ed else (in scala gerarchica). Ciascuna condizione verifica se l'operatore scritto nell'input corrisponde a uno specifico caso dichiarato in precedenza (addizione, sottrazione, moltiplicazione, divisione, potenza) e, in caso affermativo, esegue il blocco di codice associato a quella condizione, cioè il blocco di codice scritto sotto alla condizione finchè non incomincia una nuova condizione perchè poi a quel punto il codice salta quel pezzo delle condizioni e va alla parte dopo. Se l'operazione inserita non corrisponde a nessun caso noto, il blocco nell'else fornisce un messaggio di errore. Inoltre l'else è sempre l'ultima condizione mentre la prima è sempre l'if e quelle in mezzo sono sempre l'elif. Questo approccio rende il codice estremamente flessibile, consentendo allo sviluppatore di eseguire diverse operazioni senza dover scrivere del codice separato per ciascuna di esse, utilizzando per l'appunto le condizioni. Le condizioni giocano un ruolo fondamentale nell'indirizzare il flusso del programma in modo dinamico, in base alle scelte dell'utente, offrendo così in questo caso una calcolatrice completa e versatile.

[1]:
```
#Esempio di calcolatrice finale con tutte cinque le operazioni
print("Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le␣
 ↪addizioni, sottrazioni, moltiplicazioni e divisioni, inoltre puoi anche␣
 ↪calcolare il valore di una potenza!")
operazione=input("Adesso scegli l'operazione che fa al caso tuo attraverso il␣
 ↪correlato simbolo (+, -, *, /, **): ")
numero1=float(input("Inserisci il primo numero o la base della potenza: "))
numero2=float(input("Inserisci il secondo numero o l'esponente della potenza: "))
if operazione=="+":#l'if indica il primo caso possibile
    risultato=numero1+numero2
elif operazione=="-":##l'elif è l'ibrido tra if & else e indica i casi di mezzo
    risultato=numero1-numero2
elif operazione=="*":
    risultato=numero1*numero2
elif operazione=="/":
    risultato=numero1/numero2
elif operazione=="**":
    risultato=numero1**numero2
else:#l'else indica l'ultimo caso possibile
    risultato="operatore non riconosciuto. Scegli uno dei seguenti operatori␣
 ↪riavviando prima il programma: +, -, *, /, **"
print("Il risultato finale del calcolo è:",risultato)
```

```
Ciao, attraverso questa calcolatrice puoi eseguire calcoli con le addizioni,
sottrazioni, moltiplicazioni e divisioni, inoltre puoi anche calcolare il valore
di una potenza!
Adesso scegli l'operazione che fa al caso tuo attraverso il correlato simbolo
(+, -, *, /, **): non lo so
Inserisci il primo numero o la base della potenza: 5
Inserisci il secondo numero o l'esponente della potenza: 5
Il risultato finale del calcolo è: operatore non riconosciuto. Scegli uno dei
seguenti operatori riavviando prima il programma: +, -, *, /, **
```

7 IL CICLO FOR CON N

Il programma in questione è un semplice "generatore" di numeri che parte e arriva fino a un numero desiderato dall'utente. Inizia chiedendo all'utente due numeri positivi: il primo (numerodaverificare) indica fino a quale numero si vuole generare la lista della sequenza di numeri, mentre il secondo (numeroiniziale) rappresenta da quale numero positivo si desidera iniziare la numerazione. Successivamente, attraverso un ciclo for, il programma itera attraverso una sequenza di numeri a partire da "numeroiniziale" fino a "numerodaverificare" incluso. Durante ogni iterazione del ciclo, il numero corrente viene stampato a schermo dopo una stringa esplicativa ed è così per ogni volta che si ripete questo ciclo. Questo processo continua fino a quando il ciclo raggiunge il valore scelto dall'utente della variabile "numerodaverificare". Quindi alla fine si può dire che l'uso di range(numeroiniziale, numerodaverificare + 1) assicura che il valore finale "numeroiniziale" sia incluso nella sequenza, poiché senza il +1, la sequenza generata terminerebbe a "numeroiniziale" - 1, cioè al numero precedente da quello scelto dall'utente, quindi escludendo il valore desiderato "numerodaverificare" dalla

lista finale che viene stampata. In questo modo, il +1 garantisce che la numerazione vada da "numeroiniziale" fino a "numerodaverificare" scrivendo tutti i numeri della lista, inclusi entrambi i "poli" decisi come input dall'utente. Questo tipo di programma è utile quando si desidera generare e visualizzare una sequenza ordinata di numeri in base, però, alle preferenze dell'utente.

[5]:
```
#Comprendere il ciclo for con n
print("Ciao, attraverso questo programma puoi verificare la numerazione␣
 ↪ordinaria dei numeri dal e fino al punto che desideri")
numerodaverificare=int(input("Quindi, inserisci il numero positivo che desideri␣
 ↪verificare: "))
numeroiniziale=int(input("Poi, inserisci da quale numero positivo desideri che␣
 ↪la numerazione inizii: "))
print("La numerazione ordinaria dei numeri dal e fino al punto che desideri è:")
for numero in range(numeroiniziale,numerodaverificare+1):#p è un valore fisso␣
 ↪determinato dall'utente mentre +1 indica che a n verrà aggiunto il valore di 1.
 ↪ N.B.: la variabile numero è diversa dalle altre due infatti rappresenta␣
 ↪ciascun numero nella numerazione ordinaria tra numeroiniziale e␣
 ↪numerodaverificare, cioè assume successivamente ogni valore compreso tra␣
 ↪numeroiniziale e numerodaverificare e non ce mai bisogno di inizzalizzarla in␣
 ↪Python
    print(numero)
```

```
Ciao, attraverso questo programma puoi verificare la numerazione ordinaria dei
numeri dal e fino al punto che desideri
Quindi, inserisci il numero positivo che desideri verificare: 33
Poi, inserisci da quale numero positivo desideri che la numerazione inizii: 12
La numerazione ordinaria dei numeri dal e fino al punto che desideri è:
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

33

Il programma sottostante calcola la somma cumulativa dei primi n numeri interi positivi chiedendo il numero desiderato all'utente come input. La somma cumulativa è un concetto matematico che rappresenta la somma progressiva di una sequenza di numeri, aggiungendo ad ogni passo il numero corrente all'accumulatore. In questo contesto, il programma calcola la somma cumulativa dei primi n numeri interi positivi, fornendo un risultato che riflette l'incremento graduale della somma durante l'iterazione. Ad esempio, se l'utente inserisce n=5, la somma cumulativa sarà calcolata come 1+2+3+4+5, che darà come risultato 15. Il programma inizia chiedendo all'utente di inserire come input un numero intero positivo (variabile n) di cui desidera calcolare la somma cumulativa. Successivamente, il programma utilizza un ciclo for per iterare attraverso i numeri da 1 a n inclusi e quindi dal numero 1 fino al numero scelto dall'utente. Durante ogni iterazione, il programma aggiunge il numero corrente alla variabile addizione, che funge da accumulatore per la somma cumulativa, finchè il processo di iterazione non sarà arrivato al numero scelto dall'utente. L'operatore += usato in questo codice ad ogni ciclo di iterazione esegue contemporaneamente l'addizione e la riassegnazione del risultato alla variabile addizione, cioè per prima cosa fa il calcolo dell'addizione che deve svolgere e poi riassegna il valore del risultato di quel calcolo alla variabile addizione e questo permette di far due azione che dovrebbero essere assegnate con due comandi diversi in uno singolo in modo da risparmiare memoria. Infine, il programma stampa il risultato, indicando la somma dei primi n numeri interi e sottolineando che questo valore rappresenta la somma cumulativa. Questo codice fornisce un'implementazione semplice della somma cumulativa.

[3]:
```
#Comprendere il ciclo for con n facendo la somma cumulativa
print("Ciao, attraverso questo programma puoi calcolare la somma cumulativa di␣
 ↪qualsiasi numero che desideri sapere")
n=int(input("Quindi, inserisci il numero positivo che desideri calcolare: "))
addizione=0
for numero in range(1,n+1):#1 è un valore fisso mentre +1 indica che a n verrà␣
 ↪aggiunto il valore di 1
    addizione+=numero#il simbolo += svolge due compiti in uno: fa la somma e␣
 ↪riassegna il valore della variabile
print("La somma dei primi",n,"numeri interi è",addizione)
print("Perciò la somma cumulativa di",n,"è",addizione)
```

```
Ciao, attraverso questo programma puoi calcolare la somma cumulativa di
qualsiasi numero che desideri sapere
Quindi, inserisci il numero positivo che desideri calcolare: 12
La somma dei primi 12 numeri interi è 78
Perciò la somma cumulativa di 12 è 78
```

8 CALCOLARE LA MEDIA DI UNA LISTA DI NUMERI

Questo programma consente all'utente di calcolare la media di una lista di numeri scelta dall'utente stesso. La media dei numeri è ottenuta sommando tutti i numeri insieme e dividendo il risultato per il numero totale di numeri. Inizialmente, il programma chiede all'utente quanti numeri desidera inserire nella lista. Utilizzando poi un ciclo for, il programma acquisisce ogni numero specifico della lista inserito dall'utente e li aggiunge alla lista numeri utilizzando il metodo append, che per l'appunto "appende" ognuno dei numeri scritti all'utente e salvati nella variabile "numero" alla lista

vera e propria del programma. Successivamente, viene calcolata la media dei numeri presenti nella lista utilizzando i comandi sum e len. Il comando sum(numeri) restituisce la somma di tutti i numeri nella lista, mentre len(numeri) restituisce la lunghezza della lista, cioè il numero totale di numeri inseriti. Dividendo la somma per la lunghezza, si ottiene così la media dei numeri. Infine il programma stampa la lista finale, fornendo così all'utente la media di tutti i numeri inseriti da lui. Questo codice fornisce un modo efficiente per calcolare la media di una lista di numeri inseriti dall'utente.

[1]:
```
#Comprendere come usare i comandi sum e len facendo la media di una lista di␣
↪numeri
numeri=[]#l'append serve a fare una lista da poter richiamare
print("Ciao, attraverso questo programma puoi calcolare la media di una lista di␣
↪numeri che desideri sapere")
n=int(input("Quanti numeri desideri inserire in totale? "))
for i in range(n):
    numero=float(input("Inserisci il numero di cui desideri fare la media: "))
    numeri.append(numero)#così si crea la lista che varia dalla variabile␣
↪numero, "appendendo" il numero scritto dall'utente prima alla lista vera e␣
↪propria
mediadeinumeri=sum(numeri)/len(numeri)#sum=somma mentre len=lunghezza
print("La media di tutti i numeri inseriti è:",mediadeinumeri)
```

```
Ciao, attraverso questo programma puoi calcolare la media di una lista di numeri
che desideri sapere
Quanti numeri desideri inserire in totale? 5
Inserisci il numero di cui desideri fare la media: 12
Inserisci il numero di cui desideri fare la media: 33
Inserisci il numero di cui desideri fare la media: 57
Inserisci il numero di cui desideri fare la media: 127
Inserisci il numero di cui desideri fare la media: 73
La media di tutti i numeri inseriti è: 60.4
```

9 CALCOLARE IL QUADRATO DEI NUMERI

Il programma consente all'utente di calcolare il quadrato di un numero specificato sempre dall'utente. Il quadrato di un numero è ottenuto moltiplicando il numero per se stesso, per l'appunto facendo la potenza alla seconda del numero in questione. Inizialmente, il programma richiede all'utente di inserire come input un numero intero (variabile n) di cui desidera calcolare il suo quadrato. Successivamente, attraverso un ciclo for, il programma itera attraverso i numeri da 1 a n inclusi. Per ogni numero da 1 a n, il ciclo for calcola il quadrato utilizzando l'operatore di potenza ** e poi stampa il risultato dopo una stringa di testo esplicativa. Il risultato finale di output è una lista di quadrati dei primi n numeri, con ogni riga che mostra all'utente il numero originale e il suo quadrato corrispondente. Questo codice fornisce un modo semplice per calcolare il quadrato di un numero.

[1]:
```
#Comprendere come eseguire le potenze facendo il quadrato dei numeri
print("Ciao, attraverso questo programma puoi calcolare il quadrato di un numero␣
↪che desideri sapere")
```

```
n=int(input("Quindi, inserisci il numero di cui desideri sapere il quadrato: "))
print("I quadrati dei primi",n,"numeri sono: ")
for numero in range(1,n+1):
    quadratodelnumero=numero**2#il simbolo di potenza si fa con il doppio **
    print("Il quadrato di",numero,"è",quadratodelnumero)
```

```
Ciao, attraverso questo programma puoi calcolare il quadrato di un numero che
desideri sapere
Quindi, inserisci il numero di cui desideri sapere il quadrato: 12
I quadrati dei primi 12 numeri sono:
Il quadrato di 1 è 1
Il quadrato di 2 è 4
Il quadrato di 3 è 9
Il quadrato di 4 è 16
Il quadrato di 5 è 25
Il quadrato di 6 è 36
Il quadrato di 7 è 49
Il quadrato di 8 è 64
Il quadrato di 9 è 81
Il quadrato di 10 è 100
Il quadrato di 11 è 121
Il quadrato di 12 è 144
```

10 LA VERIFICA DI PARITÀ O DISPARITÀ DI UN NUMERO

Il programma consente all'utente di verificare se un numero inserito come input è pari o dispari. Inizialmente, chiede all'utente di inserire un numero intero come input della variabile numero. Successivamente, utilizza l'operatore modulo "%" (simbolo di percentuale) per verificare se il numero è divisibile per 2 o meno. Se il resto della divisione di numero per 2 è uguale a 0, il programma stampa che il numero è pari. In caso contrario, se il resto è diverso da 0, il programma stampa che il numero è dispari. L'operatore modulo restituisce il resto della divisione tra i due operandi, cioè in questo caso il numero definito dall'utente e il 2. Nello specifico nella condizione "if numero%2==0:", il programma verifica se il numero è divisibile per 2 (cioè se il resto è 0), il che indica che il numero è pari. L'else ovviamente, se il programma si trova nella altra condizione, stampa che è dispari. Questo codice fornisce una semplice verifica della parità o disparità di un numero.

[2]:
```
#Comprendere l'utillizo del simbolo di %
print("Ciao, attraverso questo programma puoi verificare se un numero è pari o␣
 ↪dispari")
numero=int(input("Quindi, inserisci il numero che desideri verificare: "))
if numero%2==0:#il simbolo di % vuol dire se è divisibile o meno un numero, ==0␣
 ↪vuol dire se in quel caso è con il resto di 0
    print("Il numero",numero,"è un numero pari")
else:
    print("Il numero",numero,"è un numero dispari")
```

```
Ciao, attraverso questo programma puoi verificare se un numero è pari o dispari
Quindi, inserisci il numero che desideri verificare: 12
```

```
Il numero 12 è un numero pari
```

11 CALCOLARE IL FATTORIALE

Il programma qui sotto consente all'utente di calcolare il fattoriale di un numero specificato dall'utente stesso. Il fattoriale di un numero è ottenuto moltiplicando tutti i numeri interi positivi fino a quel numero, il fattoriale di 5 ad esempio è calcolato moltiplicando tutti i numeri interi positivi da 1 a 5, cioè: 1 * 2 * 3 * 4 * 5, che è uguale a 120. Inizialmente, il programma richiede all'utente come input di inserire un numero positivo (variabile n) di cui desidera calcolare il fattoriale. Successivamente, attraverso un ciclo for, il programma itera attraverso i numeri da 1 a n inclusi. Per ogni numero, il ciclo for moltiplica il valore corrente della variabile fattoriale per il numero stesso e riassegna il risultato alla variabile fattoriale. L'operatore = *è un modo conciso e semplice per eseguire moltiplicazioni e riassegnare il risultato alla stessa variabile, praticamente fa due cose in uno: svolge la moltiplicazione e salva il risultato nella stessa variabile senza l'uso di due comandi separati. Ad esempio, fattoriale*=numero è equivalente a fattoriale=fattoriale*numero solo che per quest'ultimo il programma userà più memoria. Alla fine del ciclo, dopo una stringa esplicativa, il programma stampa il risultato che rappresenta il fattoriale del numero inserito all'inizio dall'utente.

[1]:
```
#Comprendere l'utillizo del simbolo di *=
print("Ciao, attraverso questo programma puoi calcolare il fattoriale di␣
 ↪qualsiasi numero che desideri sapere")
n=int(input("Quindi, inserisci il numero positivo che desideri calcolare: "))
fattoriale=1
for numero in range(1,n+1):
    fattoriale*=numero#il simbolo *= svolge due compiti in uno: fa la␣
 ↪moltiplicazione e riassegna il valore della variabile
print("Il fattoriale del numero",n,"è",fattoriale)
```

```
Ciao, attraverso questo programma puoi calcolare il fattoriale di qualsiasi
numero che desideri sapere
Quindi, inserisci il numero positivo che desideri calcolare: 12
Il fattoriale del numero 12 è 479001600
```

12 IL GIOCO DELL'INDOVINELLO

Il programma in questione è un semplice gioco in cui l'utente, o in questo caso il giocatore, deve indovinare un numero generato casualmente compreso tra 1 e 100. Inizialmente, il programma utilizza la libreria random per generare un numero segreto (salvato nella variabile numerodaindovinare) mediante la funzione randint(1, 100), che restituisce un numero intero casuale compreso tra 1 e 100 (inclusi). Successivamente, il programma inizia un ciclo "while True" che si ripeterà finché l'utente non indovina il numero segreto. Il ciclo "while True" non è altro che un ciclo che dura fino all'"infinito", cioè fino a quando l'utente non si sarà imbattuto nell'unica condizione del programma che attraverso il comando "break" riuscirà ad interrompere questo ciclo continuo e far così terminare il programma; se durante lo sviluppo del programma ci si dimentica di inserire questo comando il programma potrebbe realmente continuare "per sempre", anche se questo sarebbe scientificamente impossibile. Durante ogni iterazione del ciclo, il programma chiede all'utente di inserire un tenta-

tivo numerico (salvato nella variabile tentativo). Il programma tiene traccia del numero di tentativi effetuati con la variabile "tentativi". Se l'utente riesce ad indovinare il numero segreto, il programma stampa un messaggio di congratulazioni insieme al numero segreto stampato a schermo e al numero di tentativi effetuati, quindi a quel punto quest'ultimo esce dal ciclo con l'istruzione "break". Se il tentativo dell'utente è sbagliato, il programma fornisce un suggerimento sulla direzione corretta per aiutare il giocatore (in particolare il programma specifica se il numero è più grande o più piccolo, questo grazie a delle condizioni specifiche inserite nel programma).

[3]:
```
#Comprendere l'utillizzo della libreria random
import random
numerodaindovinare=random.randint(1,100)#random.randint vuol dire che il numero␣
↪da generare è compreso da 1 a 100 ed è intero perchè è int
tentativi=0
print("Ciao, questo gioco prevede che tu indovini un numero compreso da 1 a 100")
while True:
    tentativo=int(input("Adesso inserisci il numero che ritieni sia quello␣
↪giusto: "))
    tentativi += 1
    if tentativo==numerodaindovinare:
        print("Complimenti! Sei riuscito a indovinare il numero incognito che␣
↪è",numerodaindovinare,"in",tentativi,"tentativi")
        break#fa fermare "improvvisamente" il programma
    elif tentativo<numerodaindovinare:
        print("Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero␣
↪è più grande")
    else:
        print("Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero␣
↪è più piccolo")
```

```
Ciao, questo gioco prevede che tu indovini un numero compreso da 1 a 100
Adesso inserisci il numero che ritieni sia quello giusto: 10
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 90
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più piccolo
Adesso inserisci il numero che ritieni sia quello giusto: 20
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 80
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più piccolo
Adesso inserisci il numero che ritieni sia quello giusto: 30
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 70
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più piccolo
Adesso inserisci il numero che ritieni sia quello giusto: 40
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 60
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 61
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
```

```
Adesso inserisci il numero che ritieni sia quello giusto: 62
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 63
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 64
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 65
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 67
Ci sei quasi, ti stai avvicinando! Ti do un consiglio: il numero è più grande
Adesso inserisci il numero che ritieni sia quello giusto: 68
Complimenti! Sei riuscito a indovinare il numero incognito che è 68 in 15
tentativi
```

Qui sotto è presente un breve codice che mostra come generare un numero causale intero (int), questa parte di codice è anche presente nel codice soprastante ma ovviamente con variabili diverse.

[19]:
```
#Comprendere l'utillizzo della libreria random per generare numeri casuali interi
import random
print("Un numero di esempio intero (int) generato casualmente da un range da 1 a␣
 ↪100 è: ")
numerointerocasuale=random.randint(1,100)#random.randint vuol dire che il numero␣
 ↪da generare è compreso da 1 a 100 ed è intero perchè è int
print(numerointerocasuale)
```

```
Un numero di esempio intero (int) generato casualmente da un range da 1 a 100 è:
27
```

Invece qui sotto è presente il metodo per farlo con i numeri decimali (float).

[22]:
```
#Comprendere l'utillizzo della libreria random per generare numeri casuali␣
 ↪decimali
import random
print("Un numero di esempio decimale (float) generato casualmente da un range da␣
 ↪0,12 a 99,7 è: ")
numerofloatcasuale=random.uniform(0.12, 99.7)#random.uniform vuol dire che il␣
 ↪numero da generare è compreso da 0,12 a 99,07 ed è decimale perchè è float
print(numerofloatcasuale)
```

```
Un numero di esempio decimale (float) generato casualmente da un range da 0,12 a
99,7 è:
86.19652177433156
```

13 IL GIOCO DEL MORRA CINESE

Nel codice del gioco della morra cinese, viene sfruttata la libreria random di Python per creare un gioco interattivo e coinvolgente. Inizialmente, il programma genera casualmente la scelta da parte del computer tra carta, forbici e sasso, consentendo così una migliore dinamicità e casualità nel gioco. Successivamente, l'utente viene coinvolto con un'interfaccia di input, in cui è richiesto di

selezionare (cioè scrivere nella apposita casella di testo) la propria mossa preferita tra le opzioni disponibili: carta, forbici o sasso. Il programma fornisce poi all'utente la mossa scelta dal computer stampandola a schermo appena l'utente avrà scritto la sua mossa, creando così un'atmosfera di suspense. Le regole del gioco sono implementate attraverso delle condizioni logiche che confrontano le mosse dell'utente e del computer con delle scelte prefissate. Gli operatori logici and e or vengono utilizzati per definire le situazioni in cui l'utente può vincere, perdere o ottenere una parità contro il computer e vengono usate per poter specificare più cose contemporaneamente senza dover creare tantissime condizioni nel codice. Inoltre, il codice gestisce situazioni in cui l'utente inserisce una mossa non riconosciuta rispetto all'elenco delle opzioni scritte sopra, fornendo un messaggio di avviso e suggerendo di scegliere tra le opzioni valide riavviando però prima il programma. Un elemento interessante del codice è l'inclusione di un link esterno, in questo caso un link di Wikipedia, che invita l'utente a consultare ulteriori informazioni sul regolamento e le strategie del gioco della morra cinese. Complessivamente, il codice combina efficacemente l'input utente, logica condizionale e interazione dinamica per creare un'esperienza di gioco coinvolgente. Questo codice riprende quasi tutte le competenze raccolte finora in questa esercitazione.

[27]:
```
#Riassunto di tutte le competenze imparate finora in questa esercitazione +␣
↪comprendere l'utilizzo dei comandi not in, and e or
import random
mosse=["carta","forbici","sasso"]
mossadelcomputer=random.choice(mosse)#random.choice vuol dire che il programma␣
↪sceglierà casualmente un elemento tra quelli della lista o stringa
print("Ciao, questo gioco è la morra cinese quindi devi provare ad indovinare la␣
↪mossa giusta e battere il computer che è l'avversario")
print("Puoi trovare tutto il regolamento e le strategie del gioco a questo link:␣
↪https://it.wikipedia.org/wiki/Morra_cinese")
sceltadelgiocatore=input("Quindi adesso scegli la tua mossa scrivendo una delle␣
↪seguenti opzioni: carta, forbice o sasso: ")
if sceltadelgiocatore not in mosse:#il comando not in viene utilizzato per␣
↪verificare se un elemento NON è presente in una lista o stringa
    print("Mossa non riconosciuta. Scegli una delle seguenti mosse riavviando␣
↪prima il programma: carta, forbice o sasso")
    print("Impossibile perciò decretare un vincitore")
else:#all'interno di un else, o in una qualsiasi altra condizione, ci possono␣
↪essere altre condizioni che si possono verificare, con altre condizioni a loro␣
↪volta all'interno e così via dicendo
    print("Il computer ha scelto la seguente mossa:",mossadelcomputer)
    if sceltadelgiocatore==mossadelcomputer:
        print("Parità!")
    elif (sceltadelgiocatore=="carta" and␣
↪mossadelcomputer=="sasso")or(sceltadelgiocatore=="forbici" and␣
↪mossadelcomputer=="carta")or(sceltadelgiocatore=="sasso" and␣
↪mossadelcomputer=="forbici"):#l'and serve a comparare più cose␣
↪contemporaneamente mentre l'or funge come un oppure
        print("Complimenti! Hai vinto!")
    else:
        print("Peccato! Hai perso!")
```

```
Ciao, questo gioco è la morra cinese quindi devi provare ad indovinare la mossa
giusta e battere il computer che è l'avversario
Puoi trovare tutto il regolamento e le strategie del gioco a questo link:
https://it.wikipedia.org/wiki/Morra_cinese
Quindi adesso scegli la tua mossa scrivendo una delle seguenti opzioni: carta,
forbice o sasso: forbici
Il computer ha scelto la seguente mossa: carta
Complimenti! Hai vinto!
```