**LA RIPETIZIONE INDEFINITA: CICLO WHILE**

# 

[INTRODUZIONE](#_heading=h.dk3j1ecl5xyc)

[COSA FA IL CODICE? [5 minuti]](#_heading=h.n9wugsqac7dh)

[COSA FA IL CODICE? [5 minuti]](#_heading=h.r4c0kk30s7ln)

[MODIFICA IL CODICE? [5 minuti]](#_heading=h.7ta4acm5jyym)

[TROVA L’ERRORE](#_heading=h.8bkxfi7ulycp)

[TERMINI CHIAVE DELLA LEZIONE](#_heading=h.vfgtfqxfn3f2)

[ESERCIZI](#_heading=h.wgqpwgmhil5i)

# 

# 

# INTRODUZIONE

Nella Lezione 11 abbiamo affrontato il costrutto di iterazione for per poter ripetere un certo numero di volte delle istruzioni o per poter iterare sugli elementi di un elenco, lista o sequenza.

Nella programmazione esiste un altro tipo di iterazione, chiamato **“Ciclo indefinito While”**, e si caratterizza dal fatto che esso serve per ripetere un blocco di istruzioni fintanto che una **condizione resta vera**.

# COSA FA IL CODICE? [5 minuti]

**Lavoro in coppia**

Osservate il seguente codice:

n = 0  
**while** n<=10:  
 print(n)  
 n+=1

| Secondo voi, cosa viene visualizzato dal codice? |
| --- |
| Stamperà e aggiungerà 1 ad n finchè n è maggiore o uguale di 10 |
| Copiate e incollate il codice in VSCode (salvate con nome *165\_While.py)* e controllate cosa fa. Fa quello che vi aspettavate? Provate a dare una spiegazione in caso contrario. |
| Si |

# COSA FA IL CODICE? [5 minuti]

**Lavoro in coppia**

Osservate il seguente codice:

lista\_numeri = []  
risposta = ""  
**while** risposta != "no":  
 n = int(input("Inserisci numero: "))  
 lista\_numeri.append(n)  
 risposta = input("Vuoi inserire altri numeri? ")  
print(lista\_numeri)

| Secondo voi, quanti numeri verranno inseriti all’interno della lista lista\_numeri? |
| --- |
|  |
| Ora copiate e incollate il codice in VSCode (in fondo al file *166\_WhileLista.py*) e controllate cosa fa. Fa quello che vi aspettavate? Provate a dare una spiegazione in caso contrario. |
|  |

# 

# MODIFICA IL CODICE? [5 minuti]

**Lavoro in coppia**

Salva con nome il file precedente e chiamalo *166\_WhileLista\_2condizioni.py* e modifica il codice in modo tale che il ciclo termini se la risposta dell’utente è “no” oppure si è raggiunta una lunghezza della lista di 5 elementi.

# 

# TROVA L’ERRORE

**Lavoro in coppia**

Nel seguente codice è presente un errore: trovatelo e scrivete brevemente la motivazione.

i = 0

lista = []

**while** i <= 10:

print(i\*5)

lista.append(i)

print(lista)

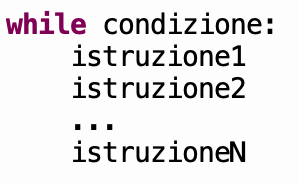
| Secondo voi, qual è l’errore? Scrivete una breve motivazione. |
| --- |
| i vale 0 quindi non può cambiare il valore moltiplicandola per 5 |

# 

# TERMINI CHIAVE DELLA LEZIONE

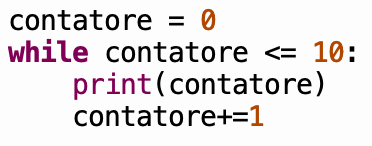
Il ciclo *while* ci permette di eseguire un blocco di codice finché una determinata condizione è e resta *True*, la parola **while** può essere infatti tradotta in italiano come **finché** o **mentre.** Quindi finché una condizione è vera il ciclo viene ripetuto, quando questa diventa falsa il ciclo viene interrotto.

La **sintassi** è la seguente:



La condizione viene controllata all’inizio e ad ogni ciclo prima di eseguire nuovamente le istruzioni.

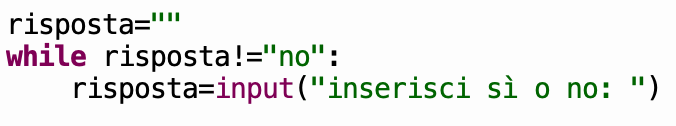
**Esempi di utilizzo:**

1. *while* con **variabile di ciclo** (contatore):  
   

La variabile contatore, dopo essere stata inizializzata a 0, viene incrementata ad ogni ciclo e quando supera il valore 10, il ciclo termina.

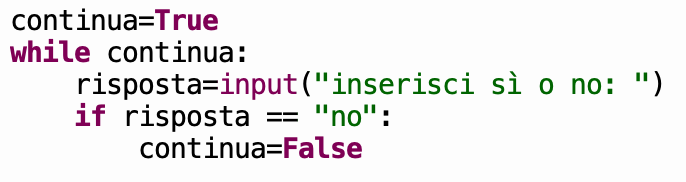
In questo caso è più comodo usare il ciclo *for.*

1. *while* con **cambio di valore** di una **variabile**

  
Il valore della variabile risposta cambia ad ogni ciclo.

Il ciclo viene ripetuto finché l’utente inserisce un valore diverso da “no”, quando invece l’utente inserisce “no” il ciclo termina.

1. *while* con **cambio di valore** di una **variabile booleana**

  
Viene utilizzata una variabile booleana (continua) il cui valore cambia al verificarsi di una condizione. La condizione del ciclo controlla che il valore della variabile continua sia a *True*, se diventa *False*, il ciclo termina.

# 

# ESERCIZI

**Lavoro in autonomia**

1. *[167\_Quadrati.py] Visualizzare i quadrati dei numeri partendo da 1 e fermandosi quando l'utente risponde "no" alla domanda "vuoi continuare?"*
2. [*168\_SommaComplessiva.py*] L'utente inserisce ripetutamente dei numeri dei quali si vuole calcolare la somma complessiva. L'inserimento dei numeri termina quando l'utente inserisce il numero 0.
3. [*169\_Indovinello.py*] Creare un in dovinello per l'utente: generare un numero da 1 a 100 in modo casuale e chiedere all'utente di indovinare il numero inserito. Il programma dice di volta in volta se il numero inserito è corretto, maggiore o minore del numero da indovinare.
4. [*170\_IntevalloMaxMin.py*] L'utente inserisce due numeri compresi tra 0 e 100 (massimo e minimo). Il programma genera casualmente dei numeri tra 0 e 100, fino a quando sono stati generati 10 numeri che sono compresi nell’intervallo massimo e minimo indicato. Il programma elenca i numeri generati e quanti numeri sono stati generati prima di soddisfare la condizione.
5. [*171\_InserisciAsterischi.py*] Il programma chiede all'utente di inserire ripetutamente un carattere, finché l'utente non ha inserito 10 volte il carattere \*. Il programma comunica quanti caratteri sono stati inseriti in totale.
6. [*172\_Calcolatrice.py*] Creare una semplice calcolatrice che fa la somma, la sottrazione, la moltiplicazione e la divisione. Chiedere all'utente due numeri e un'operazione da svolgere. Successivamente mostrare il risultato. Continuare a chiedere l'inserimento dei numeri finché alla domanda "Vuoi continuare? " l'utente non risponde no.
7. [*173\_InserimentoPassword.py*] Talvolta, capita di inserire la propria password in un sito web, sbagliarla più volte e non poter riprovare l’accesso prima di un certo tempo, oppure di dover chiedere l’intervento dell’amministratore del sito per sbloccare il proprio account. Scriviamo un programma che implementi l’accesso a un sistema protetto da password. Il programma chiede una password e ripete la richiesta fino a quando l’utente non dà la risposta corretta, oppure dà tre risposte sbagliate.
8. [*174\_ConteggioPari.py*] Il programma chiede all'utente un numero intero e poi genera numeri casuali tra 40 e 90 fermandosi quando ha generato un numero di **elementi** **pari** uguale al numero inserito dall’utente. Al termine il programma visualizza un messaggio finale che comunica quanti elementi sono stati generati in totale.
9. [*175\_FattoriPrimi.py*] Scomporre un numero in fattori primi, visualizzando tutti i suoi fattori.
10. [*176\_ListaPari.py*] Crea una lista chiamata *listanum*, che contiene 20 elementi generati in modo random tra 1 e 100. Chiedi all'utente un numero N. Copia i primi N numeri pari dalla *listanum* e mettili in una seconda lista chiamata *listapari*
11. [*177\_TrovaNumero.py*] Crea *listanum* come nell’esercizio precedente. Stabilire quanti elementi di *listanum* ci sono prima di incontrare il numero indicato in input dall'utente.

Ad esempio nella lista [3,2,5,7,6] l'utente chiede di cercare il numero 7 l'output del programma sarà 3 perché 3 sono gli elementi della lista che si trovano prima del 7.

*Ricorda che il numero potrebbe non essere presente nella lista! (in tal caso si dovrà dire il numero di elementi contenuti nella lista)*

1. [*178\_SuperoSomma.py*] Crea *listanum* come nell’esercizio precedente. Dato in input un numero, stabilire quanti elementi della lista sono necessari per avere come somma almeno il valore indicato dal numero.

Ad esempio nella lista [3,2,5,7,6] se l'utente inserisce il numero 4 l'output sarà 2 perché è necessario considerare i primi due elementi della lista per ottenere come somma almeno 4.