Esercizi sui Cicli in Python: Codice e Descrizione Dettagliata

Sezione 1: Ciclo `for` (Iterazione su Sequenze)

Esercizio 1: Stampa Nomi da una Lista

Python
```
nomi = ["Anna", "Marco", "Sara"]
for nome in nomi:
  print(nome)
```

Riga	Codice	Descrizione	Stato Variabili
1	`nomi = ["Anna", "Marco", "Sara"]`	**Inizializza** la lista di stringhe `nomi`.	`nomi = ["Anna", "Marco", "Sara"]`
2	`for nome in nomi:`	**Inizia il ciclo `for`**. La variabile `nome` prende il primo elemento della lista.	`nome = "Anna"`
3	`print(nome)`	**Stampa** il valore corrente di `nome`.	**Output:** Anna
2	`for nome in nomi:`	**Prossima iterazione**. `nome` prende il secondo elemento.	`nome = "Marco"`
3	`print(nome)`	**Stampa** "Marco".	**Output:** Marco
2	`for nome in nomi:`	**Prossima iterazione**. `nome` prende l'ultimo elemento.	`nome = "Sara"`
3	`print(nome)`	**Stampa** "Sara".	**Output:** Sara
2		La lista è esaurita. Il ciclo **termina**.	

Esercizio 2: Somma Elementi di una Lista

Python
```
numeri = [10, 5, 2]
somma = 0
for n in numeri:
  somma += n
print(somma)
```

Riga	Codice	Descrizione	Stato Variabili
1	`numeri = [10, 5, 2]`	**Definisce** la lista di numeri.	`numeri = [10, 5, 2]`
2	`somma = 0`	**Inizializza** la variabile per l'accumulo.	`somma = 0`
3	`for n in numeri:`	**Inizia il ciclo**. n prende il primo elemento (10).	`n = 10`
4	`somma += n`	**Aggiorna `somma`**: $0 + 10 = 10$.	`somma = 10`
3	`for n in numeri:`	**Prossima iterazione**. n prende il secondo elemento (5).	`n = 5`
4	`somma += n`	**Aggiorna `somma`**: $10 + 5 = 15$.	`somma = 15`
3	`for n in numeri:`	**Prossima iterazione**. n prende l'ultimo elemento (2).	`n = 2`
4	`somma += n`	**Aggiorna `somma`**: $15 + 2 = 17$.	`somma = 17`
3		La lista è esaurita. Il ciclo termina.	
5	`print(somma)`	**Stampa** il risultato finale.	**Output:** 17

Esercizio 3: Stampa con `range()`

Python

```
for i in range(1, 4):
  print(i)
```

Riga	Codice	Descrizione	Stato Variabili
1	`for i in range(1, 4):`	**Inizia il ciclo**. `range(1, 4)` genera i numeri $1, 2, 3$. i prende il primo valore (1).	`i = 1`
2	`print(i)`	**Stampa** il valore corrente di i.	**Output:** 1
1	`for i in range(1, 4):`	**Prossima iterazione**. i prende il valore successivo (2).	`i = 2`
2	`print(i)`	**Stampa** 2.	**Output:** 2
1	`for i in range(1, 4):`	**Prossima iterazione**. i prende l'ultimo valore (3).	`i = 3`
2	`print(i)`	**Stampa** 3.	**Output:** 3
1		Non ci sono più valori (range si ferma prima di 4). Il ciclo termina.	

Esercizio 4: Iterazione con `enumerate()` (Indice e Valore)

Python
```
elementi = ("A", "B")
for indice, val in enumerate(elementi):
  print(f"Posizione {indice}: {val}")
```

Riga	Codice	Descrizione	Stato Variabili
1	`elementi = ("A", "B")`	**Definisce** la tupla.	`elementi = ("A", "B")`
2	`for indice, val in enumerate(elementi):`	**Inizia il ciclo**. `enumerate` fornisce la prima coppia (indice, valore).	`indice = 0`, `val = "A"`

Riga	Codice	Descrizione	Stato Variabili
3	`print(f"Posizione {indice}: {val}")`	**Stampa** la posizione e il valore.	**Output:** Posizione 0: A
2	`for indice, val in enumerate(elementi):`	**Prossima iterazione**. `enumerate` fornisce la seconda coppia.	`indice = 1, val = "B"`
3	`print(f"Posizione {indice}: {val}")`	**Stampa** la posizione e il valore.	**Output:** Posizione 1: B
2		L'iterazione è completa. Il ciclo **termina**.	

Esercizio 5: Numeri Pari con `range()` e `step`

Python
```
for num in range(2, 7, 2):
  print(num)
```

Riga	Codice	Descrizione	Stato Variabili
1	`for num in range(2, 7, 2):`	**Inizia il ciclo**. `range(2, 7, 2)` genera numeri a partire da 2, fermandosi prima di 7, con un passo di 2.	`num = 2`
2	`print(num)`	**Stampa** 2.	**Output:** 2
1	`for num in range(2, 7, 2):`	**Prossima iterazione**. $2 + 2 = 4$.	`num = 4`
2	`print(num)`	**Stampa** 4.	**Output:** 4
1	`for num in range(2, 7, 2):`	**Prossima iterazione**. $4 + 2 = 6$.	`num = 6`
2	`print(num)`	**Stampa** 6.	**Output:** 6

Riga	Codice	Descrizione	Stato Variabili
1		Il prossimo numero sarebbe $6 + 2 = 8$, che è ≥ 7. Il ciclo **termina**.	

Sezione 2: Ciclo `while` (Iterazione Condizionale)

Esercizio 6: Contatore Base `while`

Python
```
i = 1
while i <= 3:
  print(f"Conto: {i}")
  i += 1
```

Riga	Codice	Descrizione	Stato Variabili
1	`i = 1`	**Inizializza** la variabile contatore.	`i = 1`
2	`while i <= 3:`	**Verifica condizione**: $1 \le 3$ è **Vero**. Entra nel ciclo.	
3	`print(f"Conto: {i}")`	**Stampa** il valore corrente di i.	**Output:** Conto: 1
4	`i += 1`	**Incrementa** il contatore: $1 + 1 = 2$.	`i = 2`
2	`while i <= 3:`	**Verifica condizione**: $2 \le 3$ è **Vero**.	
3	`print(f"Conto: {i}")`	**Stampa** 2.	**Output:** Conto: 2
4	`i += 1`	**Incrementa** il contatore: $2 + 1 = 3$.	`i = 3`
2	`while i <= 3:`	**Verifica condizione**: $3 \le 3$ è **Vero**.	

Riga	Codice	Descrizione	Stato Variabili
3	`print(f"Conto: {i}")`	**Stampa** 3.	**Output:** Conto: 3
4	`i += 1`	**Incrementa** il contatore: $3 + 1 = 4$.	`i = 4`
2	`while i <= 3:`	**Verifica condizione**: $4 \le 3$ è **Falso**. Il ciclo **termina**.	

Esercizio 7: Input Condizionale con `break`

Python

```
while True:
  comando = input("Digita 'stop' per uscire: ")
  if comando == "stop":
    break
  print("Continuo...")
print("Uscito dal loop.")
```

Riga	Codice	Descrizione	Stato Variabili
1	`while True:`	**Inizia un ciclo infinito** (la condizione è sempre vera).	
2	`comando = input("Digita 'stop' per uscire: ")`	**Chiede input**. Supponiamo l'utente digiti "vai".	`comando = "vai"`
3	`if comando == "stop":`	**Verifica condizione di uscita**: "vai" == "stop" è **Falso**.	
5	`print("Continuo...")`	**Stampa** il messaggio.	**Output:** Continuo...
1	`while True:`	**Ritorna** all'inizio del loop.	
2	`comando = input("Digita 'stop' per uscire: ")`	**Chiede input**. Supponiamo l'utente digiti "stop".	`comando = "stop"`

Riga	Codice	Descrizione	Stato Variabili
3	`if comando == "stop":`	**Verifica condizione**: "stop" == "stop" è **Vero**.	
4	`break`	**Interrompe** immediatamente il ciclo `while`.	
6	`print("Uscito dal loop.")`	**Esegue** l'istruzione successiva al ciclo.	**Output:** Uscito dal loop.

Esercizio 8: Potenza di 3

Python
```
p = 1
while p < 30:
  p *= 3
  print(p)
```

Riga	Codice	Descrizione	Stato Variabili
1	`p = 1`	**Inizializza** la variabile.	`p = 1`
2	`while p < 30:`	**Verifica condizione**: $1 < 30$ è **Vero**.	
3	`p *= 3`	**Moltiplica** `p` per 3: $1 \times 3 = 3$.	`p = 3`
4	`print(p)`	**Stampa** 3.	**Output:** 3
2	`while p < 30:`	**Verifica condizione**: $3 < 30$ è **Vero**.	
3	`p *= 3`	**Moltiplica** `p` per 3: $3 \times 3 = 9$.	`p = 9`
4	`print(p)`	**Stampa** 9.	**Output:** 9
2	`while p < 30:`	**Verifica condizione**: $9 < 30$ è **Vero**.	

Riga	Codice	Descrizione	Stato Variabili
3	`p *= 3`	**Moltiplica** `p` per 3: $9 \times 3 = 27$.	`p = 27`
4	`print(p)`	**Stampa** 27.	**Output:** 27
2	`while p < 30:`	**Verifica condizione**: $27 < 30$ è **Vero**.	
3	`p *= 3`	**Moltiplica** `p` per 3: $27 \times 3 = 81$.	`p = 81`
4	`print(p)`	**Stampa** 81.	**Output:** 81
2	`while p < 30:`	**Verifica condizione**: $81 < 30$ è **Falso**. Il ciclo **termina**.	

Esercizio 9: Calcolo Fattoriale con `while`

Python
```
n = 4
risultato = 1
temp_n = n

while temp_n > 0:
  risultato *= temp_n
  temp_n -= 1

print(f"{n}! = {risultato}")
```

Riga	Codice	Descrizione	Stato Variabili
1-3	`n = 4; risultato = 1; temp_n = n`	**Inizializza** il numero, il risultato e una copia per il ciclo.	`n=4, ris=1, temp_n=4`
5	`while temp_n > 0:`	**Verifica condizione**: $4 > 0$ è **Vero**.	
6	`risultato *= temp_n`	**Moltiplica**: $1 \times 4 = 4$.	`ris=4`
7	`temp_n -= 1`	**Decrementa** il contatore: $4 - 1 = 3$.	`temp_n=3`

Riga	Codice	Descrizione	Stato Variabili
5	`while temp_n > 0:`	**Verifica condizione**: $3 > 0$ è **Vero**.	
6	`risultato *= temp_n`	**Moltiplica**: $4 \times 3 = 12$.	`ris=12`
7	`temp_n -= 1`	**Decrementa**: $3 - 1 = 2$.	`temp_n=2`
5	$\dots$	$\dots$	$\dots$
5	`while temp_n > 0:`	Ultima iterazione: $1 > 0$ è **Vero**.	`ris=24, temp_n=1`
6	`risultato *= temp_n`	**Moltiplica**: $24 \times 1 = 24$.	`ris=24`
7	`temp_n -= 1`	**Decrementa**: $1 - 1 = 0$.	`temp_n=0`
5	`while temp_n > 0:`	**Verifica condizione**: $0 > 0$ è **Falso**. Il ciclo **termina**.	
9	`print(f"{n}! = {risultato}")`	**Stampa** il risultato finale.	**Output:** 4! = 24

Esercizio 10: Inversione di un Numero con `while`

Python
```
numero = 543
invertito = 0

while numero > 0:
  cifra = numero % 10
  invertito = invertito * 10 + cifra
  numero //= 10

print(invertito)
```

Riga	Codice	Descrizione	Stato Variabili
1-2	`numero = 543; invertito = 0`	**Inizializza** il numero sorgente e l'accumulatore.	`num=543, inv=0`
4	`while numero > 0:`	**Iterazione 1**: $543 > 0$ è **Vero**.	
5	`cifra = numero % 10`	**Estrae l'ultima cifra**: $543 \bmod 10 = 3$.	`cifra=3`
6	`invertito = invertito * 10 + cifra`	**Accumula**: $0 \times 10 + 3 = 3$.	`inv=3`
7	`numero //= 10`	**Rimuove l'ultima cifra**: $543 // 10 = 54$.	`num=54`
4	`while numero > 0:`	**Iterazione 2**: $54 > 0$ è **Vero**.	
5	`cifra = numero % 10`	**Estrae l'ultima cifra**: $54 \bmod 10 = 4$.	`cifra=4`
6	`invertito = invertito * 10 + cifra`	**Accumula**: $3 \times 10 + 4 = 34$.	`inv=34`
7	`numero //= 10`	**Rimuove l'ultima cifra**: $54 // 10 = 5$.	`num=5`
4	`while numero > 0:`	**Iterazione 3**: $5 > 0$ è **Vero**.	
5	`cifra = numero % 10`	**Estrae l'ultima cifra**: $5 \bmod 10 = 5$.	`cifra=5`
6	`invertito = invertito * 10 + cifra`	**Accumula**: $34 \times 10 + 5 = 345$.	`inv=345`
7	`numero //= 10`	**Rimuove l'ultima cifra**: $5 // 10 = 0$.	`num=0`

Riga	Codice	Descrizione	Stato Variabili
4	`while numero > 0:`	**Verifica condizione**: $0 > 0$ è **Falso**. Il ciclo **termina**.	
9	`print(invertito)`	**Stampa** il risultato invertito.	**Output:** 345

Sezione 3: Esercizi Misti e Avanzati (`break`, `continue`, Dizionari, Annidati)

Esercizio 11: `for` su Stringa e `continue`

Python
```
testo = "prova-salto-prova"
for char in testo:
  if char == '-':
    continue
  print(char, end="")
```

Riga	Codice	Descrizione	Stato Variabili
1	`testo = "prova-salto-prova"`	**Definisce** la stringa.	
2	`for char in testo:`	**Inizia il ciclo**. `char` prende 'p', poi 'r', 'o', 'v', 'a'.	`char='a'`
3	`if char == '-':`	Per 'a', è **Falso**.	
5	`print(char, end="")`	Stampa 'prova'.	**Output:** prova
2	`for char in testo:`	`char` prende '-'.	`char='-'`
3	`if char == '-':`	Condizione **Vera**.	
4	`continue`	**Salta** il resto del codice nel ciclo e passa alla prossima iterazione (senza stampare '-').	

Riga	Codice	Descrizione	Stato Variabili
2	`for char in testo:`	`char` prende 's', 'a', 'l', 't', 'o'.	`char='o'`
5	`print(char, end="")`	Stampa 'salto'.	**Output:** provasalto
2	`for char in testo:`	$\dots$	$\dots$
5		Stampa l'ultima 'prova'. Il ciclo **termina**.	**Output:** provasalto-prova

---

Esercizio 12: `for` con `break`

Python
```
numeri = [10, 20, 5, 30, 40]
for n in numeri:
  if n > 25:
    print(f"Trovato: {n}. Interruzione.")
    break
  print(f"Processo: {n}")
```

Riga	Codice	Descrizione	Stato Variabili
1	`numeri = [10, 20, 5, 30, 40]`	**Definisce** la lista.	
2	`for n in numeri:`	n prende 10.	`n=10`
3	`if n > 25:`	$10 > 25$ è **Falso**.	
5	`print(f"Processo: {n}")`	Stampa 10.	**Output:** Processo: 10
2	`for n in numeri:`	n prende 20, poi 5. Il processo continua.	`n=5`

Riga	Codice	Descrizione	Stato Variabili
5	$\dots$	Stampa 20 e 5.	**Output:** Processo: 20, Processo: 5
2	`for n in numeri:`	n prende 30.	`n=30`
3	`if n > 25:`	$30 > 25$ è **Vero**.	
4	`print(f"Trovato: {n}. Interruzione.")`	Stampa il messaggio.	**Output:** Trovato: 30. Interruzione.
5	`break`	**Interrompe** immediatamente il ciclo `for`. Non si raggiungerà mai 40.	

Esercizio 13: `while` con Blocco `else` (Successo)

Python
```
tentativi = 0
while tentativi < 3:
  print(f"Tentativo {tentativi + 1}")
  tentativi += 1
else:
  print("Loop completato senza interruzioni.")
```

Riga	Codice	Descrizione	Stato Variabili
1	`tentativi = 0`	**Inizializza** il contatore.	`tentativi=0`
2	`while tentativi < 3:`	Condizione **Vera** (ciclo da 0, 1, 2).	
3-4	`print(...); tentativi += 1`	Esegue le iterazioni 0, 1, 2.	`tentativi` raggiunge 3
2	`while tentativi < 3:`	Condizione **Falsa** ($3 < 3$). Il loop termina **normalmente**.	

Riga	Codice	Descrizione	Stato Variabili
5	`else:`	Il blocco `else` viene eseguito perché il ciclo non è stato interrotto da un `break`.	
6	`print("Loop completato senza interruzioni.")`	**Stampa** il messaggio.	**Output:** Loop completato...

Esercizio 14: Loop Infinito Controllato da Flag

Python
```
attivo = True
cont = 0
while attivo:
  cont += 1
  if cont == 2:
    attivo = False # Imposta la condizione di uscita per il prossimo ciclo
  print(f"Ciclo n. {cont}")
```

Riga	Codice	Descrizione	Stato Variabili
1-2	`attivo = True; cont = 0`	**Inizializza** il flag e il contatore.	`attivo=True, cont=0`
3	`while attivo:`	**Iterazione 1**. Condizione `True`.	
4	`cont += 1`	$cont$ diventa 1.	`cont=1`
5	`if cont == 2:`	$1 == 2$ è **Falso**.	
7	`print(f"Ciclo n. {cont}")`	Stampa 1.	**Output:** Ciclo n. 1
3	`while attivo:`	**Iterazione 2**. Condizione `True`.	
4	`cont += 1`	$cont$ diventa 2.	`cont=2`

Riga	Codice	Descrizione	Stato Variabili
5	`if cont == 2:`	$2 == 2$ è **Vero**.	
6	`attivo = False`	**Imposta il flag** a `False`.	`attivo=False`
7	`print(f"Ciclo n. {cont}")`	Stampa 2.	**Output:** Ciclo n. 2
3	`while attivo:`	**Verifica condizione**: `attivo` è **False**. Il ciclo **termina**.	

Esercizio 15: Iterazione su Chiavi Dizionario

Python

```
config = {"user": "admin", "mode": "test", "timeout": 60}
for chiave in config:
  print(f"Parametro: {chiave}")
```

Riga	Codice	Descrizione	Stato Variabili
1	`config = { ... }`	**Definisce** il dizionario.	
2	`for chiave in config:`	**Inizia il ciclo**. Di default, itera sulle chiavi. `chiave` prende "user".	`chiave="user"`
3	`print(f"Parametro: {chiave}")`	Stampa "user".	**Output:** Parametro: user
2	`for chiave in config:`	`chiave` prende "mode".	`chiave="mode"`
3	`print(f"Parametro: {chiave}")`	Stampa "mode".	**Output:** Parametro: mode
2	$\dots$	$\dots$	$\dots$

Riga	Codice	Descrizione	Stato Variabili
3		Stampa "timeout". Il ciclo **termina**.	

Esercizio 16: Iterazione su Chiavi e Valori Dizionario (`items()`)

Python
```
punteggi = {"A": 10, "B": 20}
for k, v in punteggi.items():
  print(f"Chiave {k} ha valore {v}")
```

Riga	Codice	Descrizione	Stato Variabili
1	`punteggi = {"A": 10, "B": 20}`	**Definisce** il dizionario.	
2	`for k, v in punteggi.items():`	**Inizia il ciclo**. `items()` fornisce la prima coppia (chiave, valore).	`k="A", v=10`
3	`print(f"Chiave {k} ha valore {v}")`	Stampa A e 10.	**Output:** Chiave A ha valore 10
2	`for k, v in punteggi.items():`	**Prossima iterazione**. Fornisce la seconda coppia.	`k="B", v=20`
3	`print(f"Chiave {k} ha valore {v}")`	Stampa B e 20.	**Output:** Chiave B ha valore 20
2		Il ciclo **termina**.	

Esercizio 17: Ciclo Annidato Semplice

Python
```
for r in range(1, 3):
  for c in range(1, 3):
    print(f"({r},{c})", end=" ")
  print() # Va a capo
```

Riga	Codice	Descrizione	Stato Variabili
1	`for r in range(1, 3):`	Ciclo esterno (Righe). r prende 1.	`r=1`
2	`for c in range(1, 3):`	Ciclo interno (Colonne). c prende 1.	`c=1`
3	`print(f"({r},{c})", end=" ")`	Stampa (1, 1) sulla riga corrente.	**Output:** (1,1)
2	`for c in range(1, 3):`	Ciclo interno. c prende 2.	`c=2`
3	`print(f"({r},{c})", end=" ")`	Stampa (1, 2) sulla riga corrente.	**Output:** (1,1) (1,2)
2		Ciclo interno termina.	
4	`print()`	Stampa un carattere di nuova riga.	**Output:** (nuova riga)
1	`for r in range(1, 3):`	Ciclo esterno. r prende 2.	`r=2`
2	`for c in range(1, 3):`	Ciclo interno. c prende 1.	`c=1`
3-4	$\dots$	Stampa (2, 1) (2, 2) e va a capo.	**Output:** (2,1) (2,2)
1		Il ciclo esterno termina.	

Esercizio 18: Generazione Lista al Quadrato

Python
```
lista_orig = [2, 3, 5]
lista_quad = []

for n in lista_orig:
  q = n * n
  lista_quad.append(q)
```

```
print(lista_quad)
```

Riga	Codice	Descrizione	Stato Variabili
1-2	`lista_orig = [2, 3, 5];` `lista_quad = []`	**Inizializza** le due liste.	`lista_quad=[]`
4	`for n in lista_orig:`	n prende 2.	`n=2`
5	`q = n * n`	Calcola il quadrato: $2 \times 2 = 4$.	`q=4`
6	`lista_quad.append(q)`	**Aggiunge** 4 a `lista_quad`.	`lista_quad=[4]`
4	`for n in lista_orig:`	n prende 3.	`n=3`
5	`q = n * n`	Calcola il quadrato: $3 \times 3 = 9$.	`q=9`
6	`lista_quad.append(q)`	**Aggiunge** 9.	`lista_quad=[4, 9]`
4	`for n in lista_orig:`	n prende 5.	`n=5`
5-6	$\dots$	Aggiunge 25.	`lista_quad=[4, 9, 25]`
8	`print(lista_quad)`	**Stampa** il risultato.	**Output:** [4, 9, 25]

Esercizio 19: Fibonacci con `while` (Stampa)

Python
```
a, b = 0, 1
count = 0
while count < 6:
  print(a, end=" ")
  a, b = b, a + b
  count += 1
```

Riga	Codice	Descrizione	Stato Variabili
1-2	`a, b = 0, 1; count = 0`	**Inizializza** i termini Fibonacci e il contatore.	`a=0, b=1, count=0`
3	`while count < 6:`	Iterazione 1. $0 < 6$ **Vero**.	
4	`print(a, end=" ")`	**Stampa** 0.	**Output:** 0
5	`a, b = b, a + b`	**Aggiorna**: a diventa 1, b diventa $0+1=1$.	`a=1, b=1`
6	`count += 1`	$count$ diventa 1.	`count=1`
3	`while count < 6:`	Iterazione 2. $1 < 6$ **Vero**.	
4	`print(a, end=" ")`	**Stampa** 1.	**Output:** 0 1
5	`a, b = b, a + b`	**Aggiorna**: a diventa 1, b diventa $1+1=2$.	`a=1, b=2`
6	$\dots$	Il ciclo continua per altre 4 volte, stampando 1, 2, 3, 5.	`count` fino a 6
3	`while count < 6:`	$6 < 6$ **Falso**. Il ciclo **termina**.	

Esercizio 20: Stampa al Contrario con `range()` e `step` negativo

Python
```
for i in range(5, 0, -1):
  print(i)
```

Riga	Codice	Descrizione	Stato Variabili
1	`for i in range(5, 0, -1):`	**Inizia il ciclo**. `range(start=5, stop=0, step=-1)` genera $5, 4, 3, 2, 1$. i prende 5.	`i = 5`
2	`print(i)`	**Stampa** 5.	**Output:** 5
1	`for i in range(5, 0, -1):`	**Prossima iterazione**. $5 - 1 = 4$.	`i = 4`
2	`print(i)`	**Stampa** 4.	**Output:** 4
1	$\dots$	$\dots$	$\dots$
1	`for i in range(5, 0, -1):`	Ultima iterazione. i prende 1.	`i = 1`
2	`print(i)`	**Stampa** 1.	**Output:** 1
1		Il prossimo passo sarebbe $1 - 1 = 0$, che è uguale a `stop`. Il ciclo **termina**.	