LICEO RUSSELL · INFORMATICA · CLASSE SECONDA

Verifica di informatica

Simulazione 1 — Produzione e lettura di codice con liste in Python

0 1 0 1 1 0 0 1

DURATA	PUNTEGGIO	ESERCIZIO	SNIPPET
90 minuti	110 punti	80 punti	30 punti

Istruzioni generali

- La prova è divisa in due parti: **produzione del codice** (80 punti) e **lettura del codice** (30 punti).
- Sono valutati la correttezza, la gestione dei casi limite e la chiarezza del codice.
- Anche l'utilizzo di vscode e la correttezza nel nominare i file costituiscono parte della valutazione.
- Non è consentito consultare appunti, libri o Internet.

Istruzioni per la consegna

- Aprite il **file explorer** (icona cartella sulla barra degli strumenti in basso) e navigate nella cartella `Questo Pc -> temi_xx`.
- Create una nuova cartella, denominata `cognome-nome`, **tutto minuscolo**. Se avete più nomi o più cognomi, metteteli tutti, concatenati dal carattere `-` (trattino). Ad esempio:

Vostro nome	Nome cartella
Marini Mattia	`marini-mattia`
Olaizola Elisa Maitea	`olaizola-elisa-maitea`

- Dentro a questa cartella, va creato **un file per ciascun esercizio**. Il nome del file è quello indicato nell'intestazione dell'esercizio, trasformato in modo che sia **tutto minuscolo e senza spazi**: l'estensione deve essere `.py` per gli esercizi di programmazione e `.txt` per le risposte alle domande e agli snippet.

> **Esercizio 1** **Esempio esercizio** **0 punti**
>
> *Ad esempio, per questo esercizio va creato un file* `esercizio1.py`

Per questa prova nello specifico mi aspetto che consegniate i seguenti file:

Parte	File da consegnare
Produzione del codice	`esercizio1.py`, `esercizio2.py`, `esercizio3.py`, `esercizio4.py`, `esercizio5.py`
Lettura del codice	`snippet1.txt`, `snippet2.txt`, `snippet3.txt`, `snippet4.txt`, `snippet5.txt`

Riepilogo dei punteggi

Voce	Dettaglio	File risposta	Punteggio
Produzione del codice			
E1	Posizioni alternate	`esercizio1.py`	**12**
E2	Inversione in place	`esercizio2.py`	**16**
E3	Palindroma	`esercizio3.py`	**16**
E4	Secondo più grande	`esercizio4.py`	**18**
E5	Rimuovi i negativi	`esercizio5.py`	**18**
Lettura del codice			
S1	Primo frammento	`snippet1.txt`	**6**
S2	Secondo frammento	`snippet2.txt`	**6**
S3	Terzo frammento	`snippet3.txt`	**6**
S4	Quarto frammento	`snippet4.txt`	**6**
S5	Quinto frammento	`snippet5.txt`	**6**
		Totale	**110 punti**

1. Produzione del codice

Per ciascun esercizio scrivi un file separato (`esercizio1.py` … `esercizio5.py`), eseguilo e verifica il risultato con gli esempi. Il programma deve stampare **soltanto** il risultato richiesto.

Prima di consegnare

- Il programma esegue senza errori?
- Viene stampato **solo** il risultato richiesto, senza messaggi in più?
- Hai provato i casi limite: lista vuota, un solo elemento, valori duplicati, numeri negativi?
- I nomi delle variabili sono chiari e il codice è rientrato in modo corretto?

E' **fondamentale** che il programma funzioni per qualsiasi input gli venga dato. Gli esercizi che sono strutturati per funzionare solo su input specifici verranno valutati **0 punti**. Ad esempio, se venisse chiesto di stampare tutti i numeri contenuti in una lista, uno ad uno, allora il seguente esercizio varrebbe zero punti:

```
l = [1,2,3]
print(l[0])
print(l[1])
print(l[2])
```

Funzioni non consentite

- Sono vietati `sum()`, `max()`, `min()`, `sorted()`, `reversed()`, `all()`, `any()`, `list.reverse()`, `list.sort()`, `list.count()`, `list.index()`.
- Sono consentiti `len()`, `append()`, `pop()`, `remove()` e i normali cicli.

Esercizio 1 — Posizioni alternate

★☆☆☆☆ **12 punti**

Consegna questo esercizio nel file `esercizio1.py`

Consegna. Data la lista `l`, stampa gli elementi uno si e uno no, ad esempio 0, 2, 4, … e così via.

Nota. Con `[1, 2, 3, 4, 5]` l'output è `1 3 5`. Puoi stampare mentre scorri la lista.

TEMPLATE DI PARTENZA

```
l = [1, 2, 3, 4, 5]

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [1, 2, 3, 4, 5]`	`1 3 5`	Si parte dalla posizione `0` e si salta una posizione.
`l = [10, 20, 30]`	`10 30`	Gli elementi in posizione `0` e `2`.
`l = []`	nessun output	Una lista vuota non stampa nulla.

Esercizio 2 — Inversione in place

★★☆☆☆ 16 punti

Consegna questo esercizio nel file `esercizio2.py`

Consegna. Inverti la lista `l` **senza usare una lista ausiliaria**. Alla fine stampa `l`.

TEMPLATE DI PARTENZA

```
l = [1, 2, 3, 4, 5]

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [1, 2, 3, 4, 5]`	`[5, 4, 3, 2, 1]`	Gli estremi vengono scambiati fino al centro.
`l = [1, 2]`	`[2, 1]`	Un solo scambio.
`l = [7]`	`[7]`	Una lista con un solo elemento resta uguale.

Esercizio 3 — Palindroma

★★☆☆☆ 16 punti

Consegna questo esercizio nel file `esercizio3.py`

Consegna. Stabilisci se la lista `l` è **palindroma**: si legge allo stesso modo da sinistra a destra e da destra a sinistra. Stampa `True` oppure `False`.

> **Nota.** Una lista vuota e una lista con un solo elemento sono palindrome.

TEMPLATE DI PARTENZA

```
l = [1, 2, 3, 2, 1]

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [1, 2, 3, 2, 1]`	`True`	La sequenza è simmetrica.
`l = [1, 2, 3]`	`False`	Gli estremi `1` e `3` sono diversi.
`l = []`	`True`	Non c'è nessuna coppia da confrontare.

Esercizio 4 — Secondo più grande

★★★☆☆ 18 punti

Consegna questo esercizio nel file `esercizio4.py`

Consegna. Trova e stampa il secondo valore più grande della lista `l`. La lista contiene almeno due valori distinti.

Nota. Se il massimo compare più volte, considera il valore distinto successivo: in `[5, 5, 3]` il secondo più grande è `3`.

TEMPLATE DI PARTENZA

```
l = [1, 2, 3, 4, 5]

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [1, 2, 3, 4, 5]`	`4`	Il massimo è `5`, il valore successivo è `4`.
`l = [5, 5, 3]`	`3`	Il massimo compare due volte, ma il secondo valore distinto è `3`.
`l = [-3, -1, -7]`	`-3`	Il massimo è `-1`, il secondo è `-3`.

Esercizio 5 — Rimuovi i negativi

★★★☆☆ **18 punti**

Consegna questo esercizio nel file `esercizio5.py`

Consegna. Modifica `l` rimuovendo tutti i numeri negativi. Alla fine stampa `l`. Lo zero va mantenuto.

Nota. Puoi modificare la lista originale oppure costruirne una nuova, purché alla fine `l` contenga solo i valori non negativi.

TEMPLATE DI PARTENZA

```
l = [3, -2, 0, -7, 5]

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [3, -2, 0, -7, 5]`	`[3, 0, 5]`	Vengono eliminati `-2` e `-7`; lo `0` resta.
`l = [-1, -2, -3]`	`[]`	Tutti gli elementi sono negativi.
`l = [0, 1, 2]`	`[0, 1, 2]`	Non ci sono numeri negativi da rimuovere.

2. Lettura del codice

Per ciascun frammento di codice spiega a parole che cosa fa e, se il programma stampa qualcosa, riporta anche l'output. Non devi riscrivere il codice o eseguirne la traccia; va dimostrato di averne compreso l'intento, descrivendone il funzionamento.

Snippet 1 Primo frammento ★★☆☆☆ 6 punti

Consegna questo esercizio nel file `snippet1.txt`

```python
a = [2, 4, 5, 3, 8]
b = True
i = 1
while i < len(a):
    if a[i] < a[i - 1]:
        b = False
    i = i + 1
print(b)
```

Snippet 2 Secondo frammento ★★☆☆☆ 6 punti

Consegna questo esercizio nel file `snippet2.txt`

```python
a = [4, 8, 6, 10]
b = 0
for x in a:
    b = b + x
c = b / len(a)
for x in a:
    if x > c:
        print(x)
```

Snippet 3 Terzo frammento ★★★☆☆ 6 punti

Consegna questo esercizio nel file `snippet3.txt`

```python
a = [4, 9, 1, 7]
b = a[0]
c = a[0]
i = 1
while i < len(a):
    if a[i] > b:
        b = a[i]
    if a[i] < c:
        c = a[i]
    i = i + 1
print(b)
print(c)
```

Snippet 4 Quarto frammento ★★★☆☆ 6 punti

Consegna questo esercizio nel file `snippet4.txt`

```python
a = [4, 9, 1, 7]
b = 0
i = 1
while i < len(a):
    if a[i] > a[b]:
        b = i
    i = i + 1
print(b)
```

Snippet 5 Quinto frammento ★★★☆☆ 6 punti

Consegna questo esercizio nel file `snippet5.txt`

```python
a = [4, 9, 1, 9]
b = a[0]
c = 1
i = 1
while i < len(a):
    if a[i] > b:
        b = a[i]
        c = 1
    elif a[i] == b:
```

```
        c = c + 1
    i = i + 1
print(c)
```