LICEO RUSSELL · INFORMATICA · CLASSE PRIMA

Verifica di informatica

Simulazione 1 — Teoria, scrittura e lettura di codice base in Python

0 1 0 1 1 0 0 1

DURATA	PUNTEGGIO	DOMANDA	ESERCIZIO	SNIPPET
80 minuti	110 punti	20 punti	60 punti	30 punti

Istruzioni generali

- La prova è divisa in tre parti: **teoria** (20 punti), **scrittura del codice** (60 punti) e **lettura del codice** (30 punti).
- Rispondi alle domande di teoria in modo **sintetico ma completo**, usando le righe a disposizione.
- Per la parte di scrittura del codice scrivi i programmi in **Python**, uno per esercizio. Sono valutate la correttezza, la gestione dei casi limite e la chiarezza del codice.
- Non è consentito consultare appunti, libri o Internet. E' possibile usare il **cheatsheet** messo su scrittura.

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
Teoria	`domanda1.txt`, `domanda2.txt`, `domanda3.txt`
Scrittura del codice	`esercizio1.py`, `esercizio2.py`, `esercizio3.py`
Lettura del codice	`snippet1.txt`, `snippet2.txt`, `snippet3.txt`

Riepilogo dei punteggi

Voce	Dettaglio	File risposta	Punteggio
Teoria			
D1	Le componenti di un computer	`domanda1.txt`	**7**
D2	Numeri, caratteri e parole	`domanda2.txt`	**8**
D3	Istruzioni e instruction set	`domanda3.txt`	**5**
Scrittura del codice			
E1	Conta i numeri pari	`esercizio1.py`	**10**
E2	La somma da 0 a n	`esercizio2.py`	**20**
E3	Tutti dentro l'intervallo	`esercizio3.py`	**30**
Lettura del codice			
S1	Primo frammento	`snippet1.txt`	**10**
S2	Secondo frammento	`snippet2.txt`	**10**
S3	Terzo frammento	`snippet3.txt`	**10**
		Totale	**110 punti**

1. Teoria

Rispondi in modo breve e preciso, usando le righe sotto ogni domanda.

Domanda 1 — Le componenti di un computer — ★☆☆☆☆ 7 punti

Consegna questo esercizio nel file `domanda1.txt`

Descrivi brevemente quali sono le principali componenti di un personal computer e qual è lo scopo di ciascuna.

> **Nota.** Una o due frasi per componente sono sufficienti. Nomina almeno **cinque** componenti diverse.

Domanda 2 — Numeri, caratteri e parole — ★★☆☆☆ 8 punti

Consegna questo esercizio nel file `domanda2.txt`

Descrivi brevemente come vengono immagazzinati i numeri in un computer. Come si possono salvare caratteri e parole?

Domanda 3 — Istruzioni e instruction set — ★★☆☆☆ 5 punti

Consegna questo esercizio nel file `domanda3.txt`

Che cos'è un'**istruzione** di un processore? Come viene rappresentata? Cosa si intende quando di parla di **instruction set**?

> **Nota.** Accenno ad **assembly** = punti extra

2. Scrittura del codice

Per ciascun esercizio scrivi un file separato (`esercizio1.py`, `esercizio2.py`, `esercizio3.py`), eseguilo e verifica il risultato con gli esempi. Il programma deve stampare **soltanto** il risultato richiesto.

Prima di consegnare

- Il programma stampa **solo** il risultato richiesto, senza messaggi in più?
- Hai provato i casi limite: `n = 0`, lista vuota e così via?
- I nomi delle variabili sono chiari e il codice è rientrato in modo corretto?
- Ogni esercizio di programmazione è nel suo file `esercizio1.py`, `esercizio2.py`, `esercizio3.py`?

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])
```

Esercizio 1 — Conta i numeri pari

★☆☆☆☆ **10 punti**

Consegna questo esercizio nel file `esercizio1.py`

Consegna. Dato un intero `n` maggiore o uguale a `0`, conta i numeri pari fra `0` e `n`, estremi inclusi, e stampa un solo numero: il conteggio trovato.

Nota. Viene richiesto di contare *quanti numeri pari* ci sono fra 0 e n, non di stamparli. Ricorda che un numero è pari se `x % 2 == 0`.

TEMPLATE DI PARTENZA

```
n = 7

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`n = 0`	`1`	L'unico numero dell'intervallo è `0`, che è pari.
`n = 7`	`4`	I pari sono `0, 2, 4, 6`.
`n = 10`	`6`	I pari sono `0, 2, 4, 6, 8, 10`: anche `n` va contato.

Esercizio 2 — La somma da 0 a n

★★☆☆☆ **20 punti**

Consegna questo esercizio nel file `esercizio2.py`

Consegna. Dato un intero `n` maggiore o uguale a `0`, calcola e stampa la somma degli interi da `0` a `n`, estremi inclusi.

> **Nota.** `n` deve essere **incluso** nella somma: con `n = 5` la somma è `15`

TEMPLATE DI PARTENZA

```
n = 10

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`n = 0`	`0`	L'intervallo contiene solo lo `0`, che non cambia la somma.
`n = 5`	`15`	`0 + 1 + 2 + 3 + 4 + 5 = 15`.
`n = 100`	`5050`	La somma di tutti gli interi da `0` a `100`.

Esercizio 3 — Tutti dentro l'intervallo

★★★☆☆ **30 punti**

Consegna questo esercizio nel file `esercizio3.py`

Consegna. Date la lista `l` e gli interi `a` e `b`, stampa `True` se **tutti** gli elementi di `l` sono fra `a` e `b`, estremi inclusi, altrimenti `False`. Se `l` è vuota, stampa `True`. Le variabili `l`, `a` e `b` sono già nel template.

> **Nota.** **Non è detto che `a < b`.** Con `a = 10` e `b = 1` l'intervallo è `[1, 10]`; non basta controllare che `x` sia compreso fra `a` e `b`.

TEMPLATE DI PARTENZA

```
l = [3, 5, 7, 12]
a = 1
b = 10

# scrivi qui il tuo codice
```

ESEMPI DI INPUT E OUTPUT

Input	Output	Spiegazione
`l = [3, 5, 7]` `a = 1`, `b = 10`	`True`	Tutti i valori stanno fra `1` e `10`.
`l = [3, 5, 7]` `a = 10`, `b = 1`	`True`	Gli estremi sono invertiti, ma l'intervallo è sempre `[1, 10]`.
`l = [-4, 0, 2, 9]` `a = 2`, `b = -4`	`False`	L'intervallo è `[-4, 2]`, ma `9` è fuori.

3. 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 che ne sta dietro, descrivendone il funzionamento.

Snippet 1 — Primo frammento

★★☆☆☆ **10 punti**

Consegna questo esercizio nel file `snippet1.txt`

```
v = [3, 7, 1, 9]
i = len(v) - 1
while i >= 0:
    print(v[i])
    i = i - 1
```

Snippet 2 — Secondo frammento

★★☆☆☆ **10 punti**

Consegna questo esercizio nel file `snippet2.txt`

Le variabili `a` e `b` contengono due numeri. Cosa succede ai valori delle due variabili durante l'esecuzione del programma?

```
a = 5
b = 8
c = a
a = b
b = c
print(a, b)
```

Snippet 3 — Terzo frammento

★★☆☆☆ **10 punti**

Consegna questo esercizio nel file `snippet3.txt`

```
n = 10
i = 0
```

```
while i <= n:
    if i % 2 == 0:
        print(i)
    i = i + 1
```