

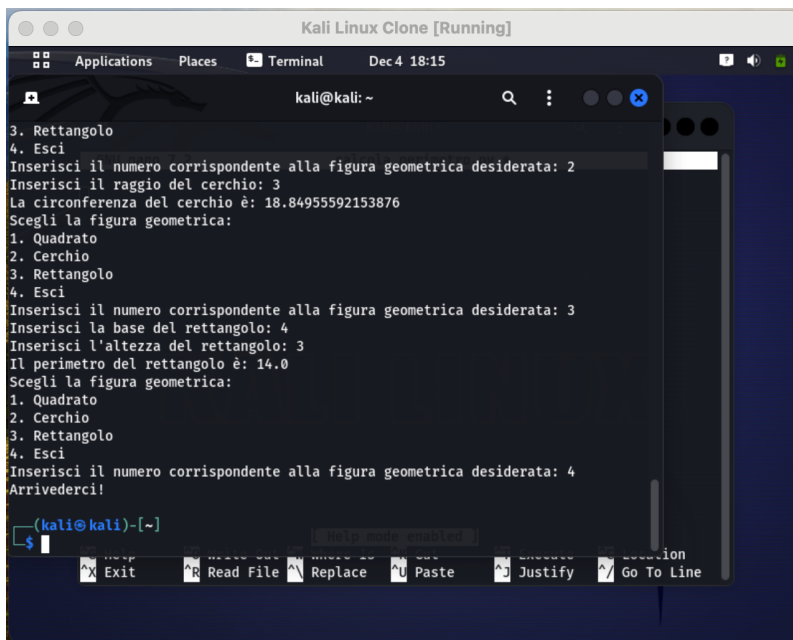
S3/L1

Traccia:

Si scriva un programma in Python che in base alla scelta dell'utente permetta di calcolare il perimetro di diverse figure geometriche (scegliete pure quelle che volete voi). Per la risoluzione dell'esercizio abbiamo scelto:

- Quadrato (perimetro = lato*4)
- Cerchio (circonferenza = $2 \cdot \pi \cdot r$)
- Rettangolo (perimetro = base*2 + altezza*2)

ESERCIZIO SVOLTO:



```
Kali Linux Clone [Running]
Applications Places Terminal Dec 4 18:15
kali@kali: ~
3. Rettangolo
4. Esci
Inserisci il numero corrispondente alla figura geometrica desiderata: 2
Inserisci il raggio del cerchio: 3
La circonferenza del cerchio è: 18.84955592153876
Scegli la figura geometrica:
1. Quadrato
2. Cerchio
3. Rettangolo
4. Esci
Inserisci il numero corrispondente alla figura geometrica desiderata: 3
Inserisci la base del rettangolo: 4
Inserisci l'altezza del rettangolo: 3
Il perimetro del rettangolo è: 14.0
Scegli la figura geometrica:
1. Quadrato
2. Cerchio
3. Rettangolo
4. Esci
Inserisci il numero corrispondente alla figura geometrica desiderata: 4
Arrivederci!
(kali@kali)-[~]
$
```

1. Ho aperto un terminale su Kali Linux ed ho inserito i comandi **"nano mio_programma.py"**.
2. Ho modificato il nome del file con **calcolo_perimetro.py**
3. Dopo di che ho inserito i seguenti comandi:

```

1 | import math
2 |
3 | def calcola_perimetro_quadrato(lato):
4 |     return lato * 4
5 |
6 | def calcola_perimetro_cerchio(raggio):
7 |     return 2 * math.pi * raggio
8 |
9 | def calcola_perimetro Rettangolo(base, altezza):
10 |     return 2 * (base + altezza)
11 |
12 | while True:
13 |     print("Scegli la figura geometrica:")
14 |     print("1. Quadrato")
15 |     print("2. Cerchio")
16 |     print("3. Rettangolo")
17 |     print("4. Esci")
18 |
19 |     scelta = input("Inserisci il numero corrispondente alla figura
geometrica desiderata: ")
20 |
21 |     if scelta == '4':
22 |         print("Arrivederci!")
23 |         break
24 |
25 |     try:
26 |         scelta = int(scelta)
27 |     except ValueError:
28 |         print("Inserisci un numero valido.")
29 |         continue
30 |
31 |     if scelta == 1:
32 |         lato = float(input("Inserisci il lato del quadrato: "))
33 |         perimetro = calcola_perimetro_quadrato(lato)
34 |         print(f"Il perimetro del quadrato è: {perimetro}")
35 |     elif scelta == 2:
36 |         raggio = float(input("Inserisci il raggio del cerchio: "))
37 |         perimetro = calcola_perimetro_cerchio(raggio)
38 |         print(f"La circonferenza del cerchio è: {perimetro}")
39 |     elif scelta == 3:

```

Commentato [lm1]: Questa istruzione importa il modulo **math** in Python, che fornisce funzioni matematiche avanzate

Commentato [lm2]: Ho utilizzato **math.pi** per calcolare la circonferenza di un cerchio.

Commentato [lm3]: Queste istruzioni definiscono tre funzioni che accettano i parametri necessari per calcolare il perimetro di un quadrato, di un cerchio e di un rettangolo. Ogni funzione restituisce il risultato del calcolo del perimetro utilizzando la logica matematica di base per ciascuna figura geometrica.

Commentato [lm4]: Questo ciclo while è il menu principale del programma. Continua ad eseguire fino a quando l'utente sceglie di uscire (opzione 4). All'interno del ciclo, vengono stampate le opzioni del menu e l'utente può inserire la scelta.

Commentato [lm5]: In questo blocco, provo a convertire la scelta dell'utente in un intero. Se l'utente inserisce qualcosa che non può essere convertito in un numero intero, viene generata un'eccezione (**ValueError**) e stampato un messaggio di errore. Il **continue** fa ripartire il ciclo **while** senza eseguire il codice successivo se c'è un errore.

```

40 | base = float(input("Inserisci la base del rettangolo: "))
41 | altezza = float(input("Inserisci l'altezza del rettangolo: "))
42 | perimetro = calcola_perimetro_rettangolo(base, altezza)
43 | print(f"Il perimetro del rettangolo è: {perimetro}")
44 | else:
45 |     print("Scelta non valida. Riprova.")

```

Commentato [lm6]: All'interno di ciascun blocco **if**, il programma richiede all'utente di inserire i dati necessari (come lato, raggio, base e altezza) e quindi calcola il perimetro utilizzando le funzioni definite all'inizio.

Commentato [lm7]: In base alla scelta dell'utente, il programma esegue il blocco di codice corrispondente. Se l'utente sceglie l'opzione 4, il programma stampa un messaggio di arrivederci e esce dal ciclo con l'istruzione **break**.

Caratteristiche di Python:

1. Python ha una sintassi chiara e leggibile, il che rende più facile scrivere e comprendere il codice. L'indentazione è parte integrante della sintassi, incoraggiando una formattazione coerente.
2. In Python, non è necessario dichiarare il tipo di variabile in anticipo. Questo rende il codice più flessibile e semplifica la gestione delle variabili.
3. L'uso del blocco **try-except** consente di gestire elegantemente situazioni in cui l'utente inserisce input non valido.
4. Python dispone di numerose librerie che semplificano molte operazioni, come **math** per le operazioni matematiche.
5. L'interazione con l'utente è semplice grazie alla funzione **input**, rendendo facile acquisire input da tastiera.
6. Python supporta l'approccio orientato agli oggetti, ma non è obbligatorio. Nel tuo esercizio, non è stato necessario utilizzare classi, ma Python le supporta bene se necessario.

Difficoltà riscontrate:

1. Per coloro che non sono abituati all'indentazione come parte integrante della sintassi, può essere inizialmente difficile gestire la formattazione del codice.
2. La conversione dell'input utente da stringa a intero è necessaria per effettuare confronti numerici. In casi di input non valido, è necessario gestire le eccezioni.

Confronto con Linguaggio C: Python vs C

1. Python ha una sintassi più chiara e leggibile rispetto a C. L'assenza di punti e virgola e parentesi graffe semplifica notevolmente la scrittura del codice.
2. Python è un linguaggio ad alto livello con una gestione dinamica delle variabili, a differenza di C, che richiede la dichiarazione esplicita del tipo di variabile.
3. Python semplifica la gestione delle eccezioni rispetto a C, dove spesso è necessario gestire manualmente gli errori attraverso il controllo dei flussi di esecuzione.
4. L'interazione con l'utente in Python è più intuitiva grazie alla funzione **input**, mentre in C richiederebbe la libreria **stdio.h** e funzioni come **scanf**.

5. In C, è necessario gestire manualmente la memoria, mentre Python ha un garbage collector che gestisce automaticamente la liberazione della memoria.

In generale, Python è spesso considerato più facile da imparare e leggere rispetto a C, grazie alla sua sintassi chiara e alle strutture dati ad alto livello. Tuttavia, il contesto in cui si sceglie l'uno o l'altro dipende dai requisiti specifici del progetto e dalle preferenze personali. Python è spesso scelto per la sua facilità di sviluppo, mentre C è spesso scelto per prestazioni e controllo più granulare sulla memoria.