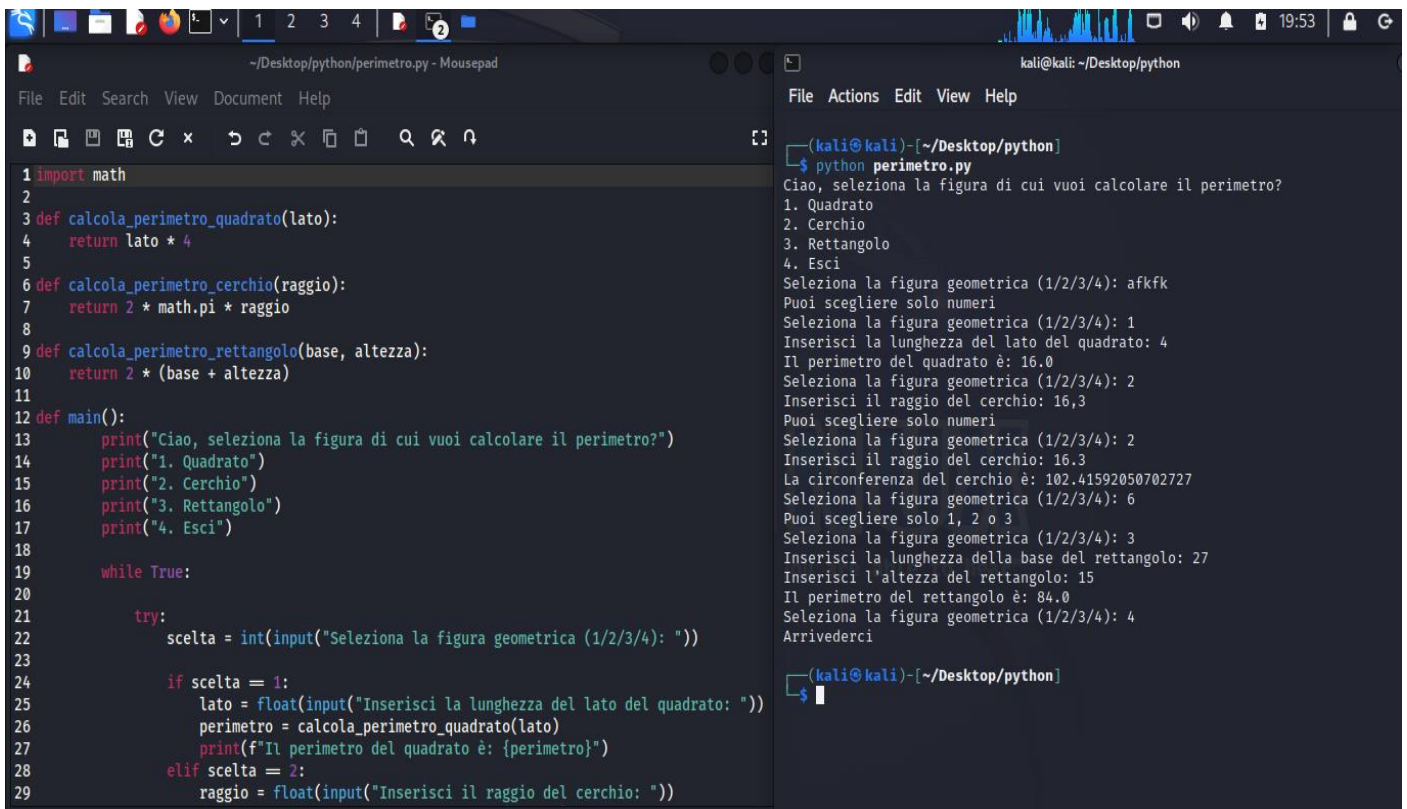


## Traccia:

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).

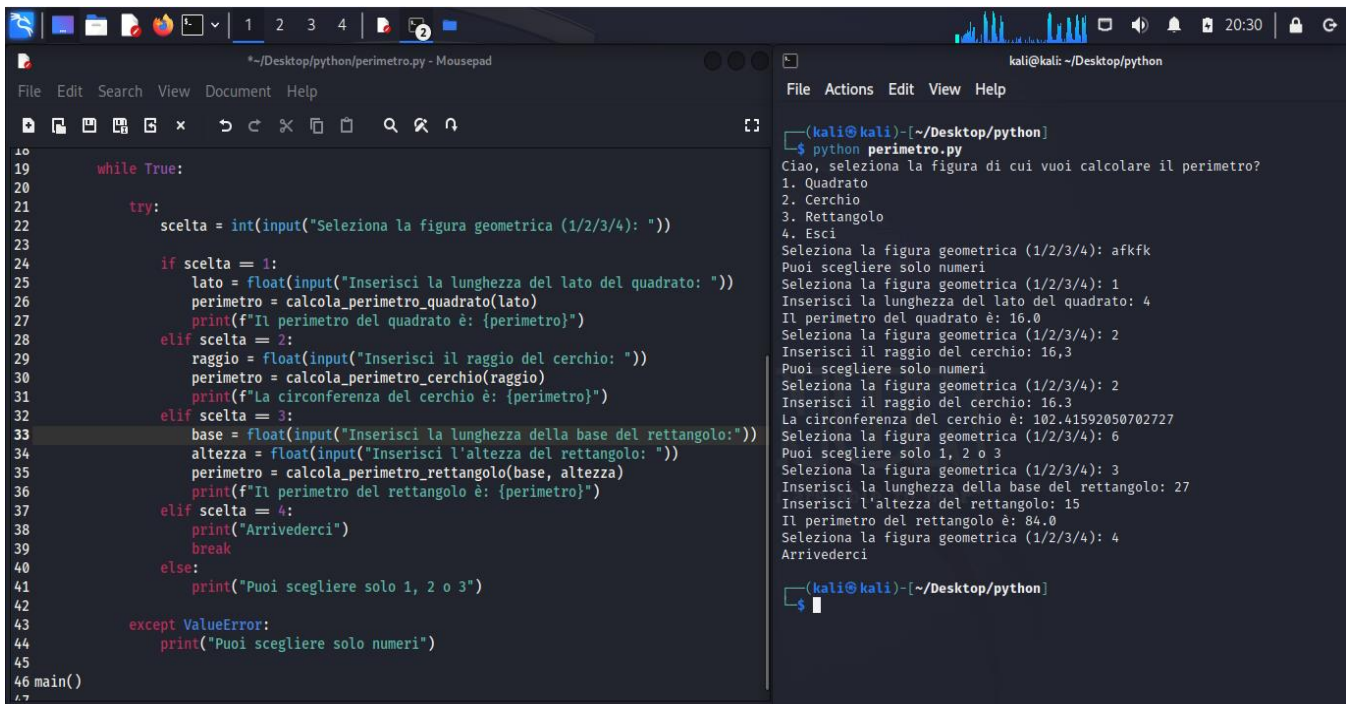
Per la risoluzione dell'esercizio abbiamo scelto:

- Quadrato (perimetro = lato\*4)
- Cerchio (circonferenza =  $2 * \pi * \text{raggio}$ )
- Rettangolo (perimetro = base\*2 + altezza\*2)



```
~/Desktop/python/perimetro.py - Mousepad
File Edit Search View Document Help
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 def main():
13    print("Ciao, seleziona la figura di cui vuoi calcolare il perimetro?")
14    print("1. Quadrato")
15    print("2. Cerchio")
16    print("3. Rettangolo")
17    print("4. Esci")
18
19    while True:
20
21        try:
22            scelta = int(input("Seleziona la figura geometrica (1/2/3/4): "))
23
24            if scelta == 1:
25                lato = float(input("Inserisci la lunghezza del lato del quadrato: "))
26                perimetro = calcola_perimetro_quadrato(lato)
27                print(f"il perimetro del quadrato è: {perimetro}")
28            elif scelta == 2:
29                raggio = float(input("Inserisci il raggio del cerchio: "))
```

```
(kali@kali) - [~/Desktop/python]
$ python perimetro.py
Ciao, seleziona la figura di cui vuoi calcolare il perimetro?
1. Quadrato
2. Cerchio
3. Rettangolo
4. Esci
Seleziona la figura geometrica (1/2/3/4): afkfk
Puoi scegliere solo numeri
Seleziona la figura geometrica (1/2/3/4): 1
Inserisci la lunghezza del lato del quadrato: 4
Il perimetro del quadrato è: 16.0
Seleziona la figura geometrica (1/2/3/4): 2
Inserisci il raggio del cerchio: 16,3
Puoi scegliere solo numeri
Seleziona la figura geometrica (1/2/3/4): 2
Inserisci il raggio del cerchio: 16.3
La circonferenza del cerchio è: 102.41592050702727
Seleziona la figura geometrica (1/2/3/4): 6
Puoi scegliere solo 1, 2 o 3
Seleziona la figura geometrica (1/2/3/4): 3
Inserisci la lunghezza della base del rettangolo: 27
Inserisci l'altezza del rettangolo: 15
Il perimetro del rettangolo è: 84.0
Seleziona la figura geometrica (1/2/3/4): 4
Arrivederci
(kali@kali) - [~/Desktop/python]
```



The screenshot shows a Kali Linux desktop environment. On the left, a text editor (Mousepad) displays a Python script named `perimetro.py`. The script is a loop that prompts the user to select a geometric shape (1 for square, 2 for circle, 3 for rectangle, 4 to exit) and then calculates its perimeter based on the input. It uses `input()` for user input and `float()` for numerical values. The script includes error handling for `ValueError` and a `main()` function. On the right, a terminal window shows the execution of the script. The user enters '1' for a square, '4' for the side length, and the program outputs the perimeter as 16.0. The user then enters '2' for a circle, '16.3' for the radius, and the program outputs the circumference as 102.41592050702727. Finally, the user enters '3' for a rectangle, '27' for the base, and '15' for the height, resulting in a perimeter of 84.0. The terminal prompt is `kali@kali: ~/Desktop/python`.

Per l'esercizio della lezione S3-L1 ho innanzitutto creato la cartella "python" su desktop Kali (`mkdir /home/kali/Desktop/python/`), poi mi sono spostato sulla cartella (`cd etc`) e ho creato il file `perimetro.py` con l'editor di testo **nano** `perimetro.py` (`ctrl + x` e `avvio`).

Per quanto riguarda il codice ho importato il **modulo math** che consente di utilizzare funzioni matematiche e costanti matematiche come il  $\pi$ , è il motivo per cui ho richiamato il modulo `math`.

Ho poi **dichiarato le tre funzioni** che si occupano del calcolo del perimetro per ciascuna figura geometrica, specificando i parametri delle funzioni tra le `()`, che poi ho richiamato nella parte centrale del codice per effettivamente effettuare le operazioni di calcolo. Infatti il `return` è ciò che consente alla funzione di restituire un valore specifico che può essere utilizzato altrove nel tuo programma.

Da notare che nella funzione relativa al cerchio ho utilizzato la costante matematica `math.pi` che consente di avere il valore del  $\pi$ .

Con il **def main()** inizia la definizione della funzione principale del programma.

Poi viene il messaggio di benvenuto all'utente con la stampa a video delle opzioni disponibili, tra le quali risulta anche la possibilità di uscire dal programma.

Poi ho definito la **variabile scelta** come **input** che l'utente digita sulla tastiera prevedendo che tale input possa essere rappresentato solo da numeri interi(**int**) in modo tale che il programma non legga altro che numeri. Qualora utente inserisse parole o lettere il programma non le assocerebbe alla variabile scelta. **Input()** è una funzione incorporata in Python che permette di leggere l'input dall'utente attraverso la

console. **In pratica, si richiede all'utente di inserire un numero corrispondente alla scelta che compare sull'interfaccia (messaggio).**

La struttura condizionale **if-elif-else** è usata per far sì che in base alla scelta dell'utente, il programma esegua il blocco di istruzioni corrispondente.

Se l'utente sceglie "1", viene richiesto il lato del quadrato;

se sceglie "2", viene richiesto il raggio del cerchio;

se sceglie "3", vengono richiesti la base e l'altezza del rettangolo.

Se l'utente fa una scelta numerica diversa dalle 4 disponibili, con l'else viene visualizzato un messaggio di errore. Un esempio valido per i tre: se l'utente sceglie 2, una volta ottenuto il valore del raggio, il programma chiama la funzione **calcola\_perimetro\_cerchio(raggio)** per calcolare il perimetro del cerchio. Il risultato viene memorizzato nella variabile **perimetro**. Infine, il programma stampa il risultato del calcolo del perimetro del cerchio utilizzando la funzione **print**.

Per ogni blocco vengono **chiamate le funzioni specifiche per il calcolo del perimetro** della figura geometrica selezionata e vengono stampati i risultati dei calcoli.

Ho utilizzato il costrutto **while** per consentire un loop che riporta alla proposizione delle domande fino a che l'utente non decide di uscire dal programma (opzione 4).

Questo costrutto si combina al **ciclo try-error value** che serve a gestire le eccezioni, ovvero l'eventualità in cui l'utente inserisca un input non valido, ovvero caratteri o stringhe di caratteri. In altre parole, se l'utente inserisce qualcosa che non può essere convertito in un intero (opzioni non numeriche), viene catturata un'eccezione di tipo **ValueError** e viene stampato un messaggio di errore.

Infine ho richiamato il **main()** per dare avvio al programma.