INDICE

INDICE……..1
INTRODUZIONE E OBIETTIVO……..2
DEFINIZIONE FUNZIONI CALCOLO……3
FUNZIONE CONTROLLO INPUT……..4
FUNZIONE PRINCIPALE……5-6
AVVIO PROGRAMMA……7
CONCLUSIONE……8
GLOSSARIO……9

INTRODUZIONE ED OBIETTIVO

Introduzione: l'attività in laboratorio prevede lo sviluppo di un programma in Python per il calcolo del perimetro di tre figure geometriche : quadrato, cerchio, rettangolo.
Il software presenta un menù testuale che consente di selezionare la figura, inserire i parametri richiesti e visualizzare, poi, il risultato.
Il menù continua a rimanere attivo finché non viene scelta l'uscita, così da poter eseguire più calcoli in sequenza.

Obiettivo: a livello formativo gli obiettivi si concentrano nell'applicare la scomposizione in funzioni per i singoli calcoli, gestire l'interazione da tastiera con convalida dell'input ed infine controllare il flusso tramite condizioni e ciclo ripetuto del menù.
Dal punto di vista operativo è previsto l'avvio del programma da terminale, la verifica del corretto funzionamento del menù, l'inserimento guidato delle misure per ciascuna figura e la conferma dell'esattezza dei risultati restituiti.
L'applicazione deve comportarsi solidamente in presenza di dati non validi e consentire all'utente di terminare l'esecuzione in maniera esplicita.

DEFINIZIONE FUNZIONI CALCOLO

```
  GNU nano 8.6
# ESERCIZIO M2W6D4 - GENOVESE VIKI SUSANNA

# IMPORTAZIONE LIBRERIE
import math

# DEFINIZIONE FUNZIONI DI CALCOLO

def perimetro_quadrato(lato):
    """
    Calcola il perimetro del quadrato a partire dalla misura del lato.
    La formula utilizzata è: perimetro = 4 × lato
    Restituisce il valore numerico del perimetro calcolato.
    """
    return 4 * lato


def circonferenza_cerchio(raggio):
    """
    Calcola la circonferenza del cerchio a partire dal raggio.
    La formula utilizzata è: circonferenza = 2 × π × raggio
    Restituisce il valore numerico della circonferenza calcolata.
    """
    return 2 * math.pi * raggio


def perimetro_rettangolo(base, altezza):
    """
    Calcola il perimetro del rettangolo partendo da base e altezza.
    La formula utilizzata è: perimetro = 2 × base + 2 × altezza
    Restituisce il valore numerico del perimetro calcolato.
    """
    return 2 * base + 2 * altezza
```

Il laboratori si apre con l'import della **libreria math** necessaria per accedere alla **costante pi** utilizzata nel calcolo della circonferenza; segue la definizione di **tre funzioni pure** e indipendenti che incapsulano le formule geometriche dei perimetri restituendo sempre un valore numerico.
1.**perimetro_quadrato(lato)** implementa la relazione lineare tra lato e perimetro del quadrato applicando 4 * lato, assumendo un valore reale positivo. 2.**circonferenza_cerchio(raggio)** usa 2 * *math.pi* * raggio per ottenere la lunghezza della circonferenza, sfruttando *math.pi* per garantire precisione della costante π.
3.**perimetro_rettangolo(base, altezza)** calcola il contorno del rettangolo con 2 * base + 2 * altezza, mantenendo simmetria e leggibilità rispetto alle grandezze in ingresso.
Tutte le funzioni sono commentante con *docstring* affinchè spiegHino lo scopo, la formula adottata ed il tipo di valore restituito facilitando la compresione dell'operazione adottata.

FUNZIONE CONTROLLO INPUT

```python
# FUNZIONE CONTROLLO INPUT

def leggi_float_positivo(prompt):
    """
    Questa funzione chiede all'utente di inserire un numero maggiore di zero.
    Accetta anche la virgola come separatore decimale e controlla che
    l'input sia effettivamente numerico. In caso contrario, richiede
    nuovamente il valore finché non viene inserito un dato corretto.
    """
    while True:
        testo = input(prompt).strip().replace(",", ".")
        try:
            val = float(testo)
            if val > 0:
                return val
            print("Il valore deve essere maggiore di zero.\n")
        except ValueError:
            print("Inserisci un numero valido (es. 3 o 2.5).\n")
```

La **funzione leggi_float_positivo(prompt)** ha l'obiettivo di restituire solo un numero reale strettamente positivo, respingendo ogni altro input.

While True crea un ciclo potenzialmente infinito che termina solo quando si ottiene un valore valido, ho adottato questa soluzione iperché la validità dell'input non è nota a priori e non esiste una condizione da valutare prima del primo tentativo.

All'interno del ciclo **testo = input(prompt).strip().replace(",", ".")** si legge ciò che l'utente digita, rimuove spazi superflui ai bordi con **strip()** e normalizza la notazione decimale sostituendo la virgola con il punto: questo permette di accettare indifferentemente "3,5" o "3.5", riducendo gli errori formali.

Il blocco **try** tenta la *conversione* con **val = float(testo)** qui passa da stringa a numero in virgola mobile e, se la conversione non è possibile , Python interagisce con l'opzione ValueError. Successivamente viene mostrato un messaggio guida ed il flusso torna in cima al ciclo, formulando un nuovo input.

Quando la conversione riesce, l'istruzione **if val > 0** rappresenta il controllo semantico escludendo zero e i negativi perché un lato, un raggio o una misura geometrica non possono essere nulli o sottozero; se la **condizione è vera**, ***return val conclude*** la funzione restituendo il numero e interrompendo anche il ciclo (non serve break perché l'uscita dalla funzione ferma tutto).

Se invece il numero è <= 0 viene stampato un avviso ("Il valore deve essere maggiore di zero") e, non essendoci return, il ciclo riparte.

Sostanzialmente **while True** garantisce ripetizione finché non arriva un dato corretto, **input/strip/replace** produce una stringa pulita ,

Try/except separa errori di forma (non numerico) da errori di significato,

if val > 0 formalizza il vincolo del dominio ,

ed infine **return** fornisce un'unica via d'uscita pulita non appena il requisito è soddisfatto, mantenendo la funzione riutilizzabile e il resto del programma libero da controlli ridondanti.

FUNZIONE PRINCIPALE

```python
# FUNZIONE/PROGRAMMA PRINCIPALE

def main():
    """
    Mostra un menu con le tre figure geometriche.
    In base alla scelta dell'utente, chiede le misure necessarie
    e calcola il perimetro utilizzando la funzione appropriata.
    Il programma continua finché l'utente non sceglie di uscire.
    """
    print("CALCOLATORE DI PERIMETRI")

    while True:
        print("\nScegli la figura:")
        print("1) Quadrato")
        print("2) Cerchio")
        print("3) Rettangolo")
        print("0) Esci")

        scelta = input("Seleziona un'opzione: ").strip()

        # QUADRATO
        if scelta == "1":
            lato = leggi_float_positivo("Inserisci la lunghezza del lato: ")
            risultato = perimetro_quadrato(lato)
            print(f"Perimetro del quadrato = {risultato:.4f}")

        # CERCHIO
        elif scelta == "2":
            r = leggi_float_positivo("Inserisci il raggio del cerchio: ")
            risultato = circonferenza_cerchio(r)
            print(f"Circonferenza del cerchio = {risultato:.4f}")

        # RETTANGOLO
        elif scelta == "3":
            b = leggi_float_positivo("Inserisci la base del rettangolo: ")
            h = leggi_float_positivo("Inserisci l'altezza del rettangolo: ")
            risultato = perimetro_rettangolo(b, h)
            print(f"Perimetro del rettangolo = {risultato:.4f}")

        # USCITA
        elif scelta == "0":
            print("Uscita dal programma. Ciao!")
            break

        # SCELTA NON VALIDA
        else:
            print("Scelta non valida. Riprova.")
```

Dopo la definizione delle funzioni di calcolo e di controllo dell'input, il programma introduce un menu testuale che consente di scegliere quale figura geometrica calcolare.

La struttura è racchiusa in un ciclo **while True:,** che mantiene il programma attivo finché l'utente non decide esplicitamente di uscire; questo approccio garantisce una navigazione fluida in quanto dopo ogni calcolo il menu riappare, evitando di dover riavviare lo script manualmente.

All'interno del ciclo la variabile scelta riceve l'input dell'utente tramite *input(),* e la stringa ottenuta viene confrontata con una serie di condizioni **if, elif e else**.

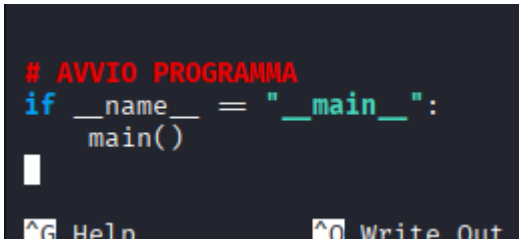
La prima verifica (**if scelta == "1":**) corrisponde al calcolo del perimetro del quadrato, quando questa condizione è vera, il programma invoca la **funzione leggi_float_positivo()** per richiedere la misura del lato, e ne memorizza il risultato in una variabile locale.

Subito dopo, richiama la **funzione perimetro_quadrato(lato)** e visualizza il valore ottenuto con un messaggio esplicativo.

La seconda condizione (**elif scelta == "2":)** gestisce il cerchio, richiedendo il raggio e calcolando la circonferenza con **circonferenza_cerchio(raggio).**

La terza **(elif scelta == "3":)** riguarda il rettangolo, per cui vengono chiesti base e altezza, e viene eseguita la funzione perimetro_rettangolo(base, altezza).
Ogni ramo è indipendente, ma segue lo stesso schema logico:
1) lettura del valore numerico controllato,
2) invocazione della funzione specifica,
3) stampa del risultato.
L'ultima condizione **(elif scelta == "4":)** rappresenta l'uscita dal programma.
Quando l'utente digita "4", viene eseguito un **break, che interrompe il ciclo while True** e conclude il programma in modo ordinato.
Infine, un blocco else cattura ogni altra possibilità non prevista, mostrando un messaggio di errore come *"Scelta non valida"* e rilanciando il menu.
Questa architettura con *while True* e catena di *if* offre semplicità e chiarezza: ogni opzione è gestita in un ramo separato, non serve una logica di controllo esterna, e l'utente può ripetere più calcoli consecutivi.
Per concludere la struttura sequenziale rende il codice facilmente estendibile ad esempio per aggiungere nuove figure geometriche o funzioni di area senza alterare la logica principale.

AVVIO PROGRAMMA



La parte finale del codice introduce il blocco **if __name__ == "__main__": seguito dalla chiamata a main(),** elemento fondamentale per stabilire il punto d'ingresso del programma.
La condizione if __name__ == "__main__": serve dunque come filtro: garantisce che il blocco sottostante venga eseguito solo quando il file rappresenta l'entry point dell'applicazione, impedendo che il menu principale si attivi automaticamente.
Quando la condizione risulta vera, la funzione main() viene richiamata e da quel momento inizia la logica interattiva che mostra il menu, richiede l'input e calcola i perimetri.
Questa struttura separa la parte eseguibile dalle definizioni logichein quanto consente di importare le funzioni di calcolo o di validazione da altri file senza attivare il programma, migliora la leggibilità e favorisce la manutenzione del codice.
Inoltre, rende chiaro che main() è il punto di partenza ufficiale dell'applicazione, gArantendo continuità al processo.

CONCLUSIONE

L’esercizio rappresenta un esempio strutturato in cui ogni parte del codice ha un ruolo preciso e contribuisce alla chiarezza complessiva del progetto.

Le funzioni di calcolo racchiudono le formule geometriche in unità indipendenti, rendendo il programma facilmente estendibile e mantenibile inoltre la funzione di controllo dell’input introduce un meccanismo robusto di validazione, capace di gestire errori comuni e di garantire che i dati elaborati rispettino i requisiti numerici richiesti.

Il menu principale, sostenuto dal ciclo while True, offre un’interfaccia semplice ma funzionale, che consente di ripetere i calcoli e di navigare tra le diverse opzioni senza interruzioni.

Infine, l’inserimento della “main guard” if __name__ == "__main__": completa la struttura rendendo il codice riutilizzabile in contesti diversi.

GLOSSARIO

Funzione: blocco di codice che svolge un compito specifico e può essere richiamato più volte nel programma; migliora l'organizzazione e la riusabilità del codice.
Parametro: valore passato a una funzione al momento della chiamata, utilizzato per eseguire operazioni personalizzate all'interno della funzione.
Return: istruzione che interrompe l'esecuzione di una funzione e restituisce un valore al punto da cui è stata chiamata.
While True: ciclo potenzialmente infinito che continua a ripetersi finché non viene interrotto da un comando come *break* o *return*; utile quando la condizione di uscita non è nota in anticipo.
If / Elif / Else: struttura di controllo che permette di eseguire blocchi di codice differenti in base al verificarsi di determinate condizioni logiche.
Try / Except: costrutto per la gestione delle eccezioni; il blocco *try* contiene il codice che può generare un errore, mentre *except* specifica cosa fare se l'errore si verifica.
ValueError: tipo di errore che si presenta quando si tenta di convertire una stringa in numero o di eseguire un'operazione con un valore del tipo errato.
Float: tipo di dato numerico che rappresenta numeri con la virgola decimale, utile per calcoli che richiedono precisione.
Input(): funzione integrata in Python che consente di ricevere dati digitati dall'utente durante l'esecuzione del programma.
Main(): funzione principale che racchiude la logica operativa del programma, in questo caso la gestione del menu e dei calcoli.
If name == "main": condizione speciale che indica il punto d'ingresso del programma; assicura che il codice venga eseguito solo se il file è lanciato direttamente e non importato.
Modulo: file Python che contiene definizioni di funzioni, classi o variabili riutilizzabili in altri programmi tramite l'istruzione import.
Break: istruzione che interrompe immediatamente un ciclo, utile per fermare l'esecuzione quando si raggiunge una determinata condizione.
Libreria math: insieme di funzioni matematiche predefinite di Python, come *math.pi* per ottenere il valore della costante π.
Pi (π): costante matematica che rappresenta il rapporto tra la circonferenza e il diametro di un cerchio, approssimata a 3.14159.