INDICE

INTRODUZIONE ED OBIETTIVO

Questo esercizio facoltativo rappresenta un'estensione maggiore del programma precedente dedicato al calcolo dei perimetri, con l'obiettivo di integrare una logica più dinamica e interattiva.
Oltre al semplice calcolo del perimetro, il programma aggiunge la **determinazione dell'area** per ciascuna figura geometrica, restituendo entrambe le misure come risultato.
La principale differenza rispetto all'esercizio base risiede nella **gestione del valore iniziale**, che viene richiesto una sola volta all'avvio e successivamente aggiornato a ogni iterazione: l'area calcolata in un ciclo diventa infatti il nuovo valore di riferimento per il ciclo successivo.
Inoltre, la struttura del menu è resa *progressiva*, poiché le **figure disponibili vengono rimosse** man mano che vengono selezionate, impedendo ripetizioni e rendendo il flusso più coerente e realistico.
L'obiettivo finale è quello di dimostrare una progettazione del codice capace di mantenere chiarezza e modularità pur introducendo una logica più articolata.

FUNZIONI DI CALCOLO

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

# IMPORTAZIONE LIBRERIE
import math

# FUNZIONI DI CALCOLO (perimetro e area)

def quadrato_perimetro_area(lato):
    """
    Calcola perimetro e area del quadrato dato il lato.
    Formule: perimetro = 4 × lato ; area = lato × lato
    Ritorna una coppia (perimetro, area).
    """
    perimetro = 4 * lato
    area = lato * lato
    return perimetro, area


def rettangolo_perimetro_area(base, altezza):
    """
    Calcola perimetro e area del rettangolo date base e altezza.
    Formule: perimetro = 2 × base + 2 × altezza ; area = base × altezza
    Ritorna una coppia (perimetro, area).
    """
    perimetro = 2 * base + 2 * altezza
    area = base * altezza
    return perimetro, area


def cerchio_perimetro_area(raggio):
    """
    Calcola circonferenza e area del cerchio dato il raggio.
    Formule: circonferenza = 2 × π × r ; area = π × r²
    Ritorna una coppia (circonferenza, area).
    """
    perimetro = 2 * math.pi * raggio
    area = math.pi * (raggio ** 2)
    return perimetro, area
```

In questa prima parte del programma la struttura rimane sostanzialmente invariata rispetto all'esercizio precedente poiché le stesse formule matematiche vengono riutilizzate per calcolare i perimetri delle tre figure geometriche principali: **quadrato**, **rettangolo** e **cerchio**.
Non è stato necessario modificare la logica di base, in quanto le relazioni geometriche alla base del calcolo del perimetro non cambiano.
L'unica aggiunta significativa riguarda l'inclusione del calcolo dell'**area** per ciascuna figura, che viene restituita insieme al perimetro come coppia di valori, questa scelta consente di ampliare le informazioni restituite senza alterare la chiarezza e la modularità del codice, mantenendo le funzioni pure e riutilizzabili anche in contesti diversi.

FUNZIONE INPUT

```python
# FUNZIONE DI CONTROLLO DELL'INPUT

def leggi_float_positivo(prompt):
    """
    Chiede un numero > 0 (accetta anche la virgola come separatore decimale).
    Ripete la richiesta finché l'input non è valido.
    """
    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**) resta invariata rispetto all'esercizio precedente, poiché già garantiva una gestione corretta e robusta dell'input numerico.
Continua a utilizzare un ciclo **while True**: per ripetere la richiesta finché non viene fornito un valore valido, controllando che l'input sia effettivamente numerico e maggiore di zero;la normalizzazione del testo tramite **replace(",", ".")** permette di accettare indifferentemente la virgola o il punto come separatore decimale, migliorando l'esperienza utente.
La scelta di mantenere la stessa funzione nasce dal fatto che il suo comportamento era già completo e pienamente adeguato anche per l'esercizio facoltativo: non vi era necessità di introdurre modifiche, il controllo dell'input positivo continua a essere essenziale e sufficiente per tutte le figure geometriche utilizzate.

FUNZIONE PRINCIPALE

```python
# PROGRAMMA PRINCIPALE

def main():
    """
    Logica del facoltativo:
    - valore_iniziale letto una sola volta all'avvio;
    - menu con figure rimanenti; dopo la scelta, quella figura viene rimossa;
    - l'area calcolata diventa il nuovo valore_iniziale per il prossimo giro.
    """
    print("CALCOLATORE (FACOLTATIVO): PERIMETRO + AREA CON VALORE CHE SI AGGIORNA\n")

    # 1) Valore iniziale fornito una sola volta
    valore_iniziale = leggi_float_positivo("Inserisci il valore iniziale: ")

    # 2) Elenco figure disponibili (si svuota man mano)
    figure = ["Quadrato", "Rettangolo", "Cerchio"]

    # 3) Ciclo finché ci sono figure
    while figure:
        print("\nFigure disponibili:")
        for i, f in enumerate(figure, start=1):
            print(f"{i}) {f}")

        # Scelta utente
        scelta_txt = input("Scegli una figura (numero) oppure premi Invio per uscire: ").strip()
        if scelta_txt == "":
            print("Uscita richiesta. Fine.")
            break

        # Validazione scelta
        try:
            scelta = int(scelta_txt)
        except ValueError:
            print("Scelta non valida. Riprova.")
            continue

        if scelta < 1 or scelta > len(figure):
            print("Scelta non valida. Riprova.")
            continue

        figura = figure.pop(scelta - 1)  # rimuove la figura scelta dal menu

        # 4) Calcolo in base alla figura scelta
        if figura == "Quadrato":
            lato = valore_iniziale                          # usAre direttamente il valore corrente
            perimetro, area = quadrato_perimetro_area(lato)
            print(f"\nHai scelto: {figura}")
            print(f"lato = {lato:g}")
            print(f"Perimetro = {perimetro:.4f}")
            print(f"Area = {area:.4f}")

        elif figura == "Rettangolo":
            base = valore_iniziale
            altezza = valore_iniziale / 2                   # come nell'esempio: altezza = base / 2
            perimetro, area = rettangolo_perimetro_area(base, altezza)
            print(f"\nHai scelto: {figura}")
            print(f"base = {base:g} ; altezza = {altezza:g}")
            print(f"Perimetro = {perimetro:.4f}")
            print(f"Area = {area:.4f}")

        else:  # Cerchio
            r = valore_iniziale                             # raggio = valore corrente
            perimetro, area = cerchio_perimetro_area(r)
            print(f"\nHai scelto: {figura}")
            print(f"raggio = {r:g}")
            print(f"Circonferenza = {perimetro:.4f}")
            print(f"Area = {area:.4f}")

        # 5) L'area diventa il nuovo valore di partenza
        valore_iniziale = area
        print(f"\n→ Nuovo valore iniziale (area appena calcolata) = {valore_iniziale:.4f}")

    if not figure:
        print("\nNon ci sono più figure disponibili. Il programma termina.")
```

In questa sezione si introduce una logica più dinamica rispetto al primo esercizio.
Il programma comincia chiedendo all'utente un **valore iniziale**, ottenuto tramite la funzione **leggi_float_positivo().**

Questo valore viene utilizzato come **punto di partenza** per i calcoli delle figure e, dopo ogni iterazione, viene aggiornato assumendo come nuovo valore l'**area** appena calcolata.
L'inizializzazione di una **lista figure = ["Quadrato", "Rettangolo", "Cerchio"]** consente di gestire le opzioni del menu in modo dinamico: ogni volta che una figura viene scelta, viene rimossa dall'elenco grazie al **metodo pop(),** così da non poter essere selezionata una seconda volta.
Il ciclo while figure: controlla che esistano ancora figure disponibili; finché la lista non è vuota, il programma continua a proporre le opzioni rimaste. All'interno del ciclo, le figure vengono elencate utilizzando *enumerate(),* che associa automaticamente un numero progressivo a ciascuna voce, rendendo la selezione più chiara e ordinata.
L'utente può uscire in qualsiasi momento premendo semplicemente "Invio": in questo caso, la stringa vuota viene intercettata dal controllo ed il ciclo viene interrotto con **break**.
Segue poi la fase di **validazione della scelta**, che assicura che l'input sia numerico e coerente con le opzioni del menu.
Il **blocco try/except** gestisce eventuali errori di conversione (ad esempio se l'utente scrive una lettera invece di un numero), mentre le condizioni **if** impediscono di selezionare numeri non presenti nel menu; questa parte rappresenta un miglioramento importante rispetto all'esercizio base poiché introduce un controllo più rigoroso dell'input e una gestione automatica delle opzioni, rendendo l'interazione più fluida e riducendo la possibilità di errore da parte dell'utente.

Una volta validata la scelta, il programma esegue il blocco di calcolo corrispondente alla figura selezionata.
La struttura utilizza una serie di **condizioni if, elif ed else** per distinguere i tre casi: quadrato, rettangolo e cerchio.
In ogni ramo, la ***variabile valore_iniziale*** viene impiegata come grandezza di riferimento, assumendo il ruolo di lato, base o raggio a seconda della figura.
Nel caso del **rettangolo**,si introduce una logica leggermente più articolata: la base corrisponde al valore corrente, mentre l'altezza viene calcolata come metà della base (altezza = valore_iniziale / 2), così da mantenere un rapporto costante tra le dimensioni.
Dopo la chiamata della funzione specifica per la figura il programma riceve due risultati: il **perimetro** e l'**area**, restituiti come coppia di valori; entrambi vengono mostrati all'utente con un formato ordinato e leggibile, grazie all'uso di ***f-string*** che limitano i decimali a quattro cifre, garantendo precisione e coerenza nella presentazione dei risultati.
La vera novità di questa sezione è tuttavia la gestione dello **stato dinamico del valore iniziale**: dopo ogni calcolo, l'area appena ottenuta diventa il nuovo valore di partenza per il ciclo successivo (valore_iniziale = area), questo meccanismo crea una progressione logica tra le figure, dove ogni risultato influenza il passo seguente, simulando un processo di aggiornamento continuo dei dati.
Quando la lista delle figure si esaurisce o l'utente decide di terminare, il programma comunica la fine delle operazioni con un messaggio conclusivo, garantendo così un flusso ordinato e privo di errori.

AVVIO DEL PROGRAMMA

```
# AVVIO
if __name__ == "__main__":
    main()
```

L'ultima parte contiene la **condizione if __name__ == "__main__":**, che fa partire il programma richiamando la **funzione main().**
Il suo scopo è lo stesso dell'esercizio precedente, ma qui il contesto è più ampio: non avvia solo un calcolatore di perimetri, bensì un sistema più dinamico che gestisce anche le aree, aggiorna automaticamente i valori e rimuove le figure già usate.

CONCLUSIONE

Questo esercizio facoltativo rappresenta un'evoluzione naturale del progetto precedente, mantenendo la stessa chiarezza strutturale ma introducendo una logica più dinamica e interattiva.

L'aggiunta del calcolo delle aree, la gestione del valore iniziale che si aggiorna ad ogni iterazione ed il menu progressivo che rimuove le figure già selezionate rendono il programma più realistico.

Pur ampliando le funzioni, il codice resta ordinato, leggibile e fedele ai principi di modularità e riutilizzo.

GLOSSARIO

Lista (list): struttura che contiene più elementi ordinati; in questo programma serve per memorizzare le figure geometriche disponibili e rimuoverle man mano che vengono scelte.
Pop(): comando che elimina un elemento da una lista; qui viene usato per togliere la figura già selezionata dal menu.
Enumerate(): funzione che permette di numerare automaticamente gli elementi di una lista, semplificando la creazione del menu con numeri progressivi.
Break: istruzione che interrompe l'esecuzione di un ciclo; usata per uscire dal programma quando l'utente non inserisce alcuna scelta.
Try / Except: blocco che gestisce gli errori; serve per evitare che il programma si blocchi se l'utente inserisce un valore non numerico.
While: ciclo che ripete un blocco di istruzioni finché la condizione rimane vera; qui mantiene attivo il programma finché ci sono figure da calcolare.
If / Elif / Else: struttura che controlla le diverse scelte dell'utente, permettendo di eseguire azioni diverse per ogni figura geometrica selezionata.
Return: comando che conclude l'esecuzione di una funzione e restituisce il risultato al programma principale.
Main(): parte centrale del programma che gestisce il menu, le scelte e i calcoli.
If name == "main": blocco che fa partire il programma solo quando viene eseguito direttamente, evitando che si avvii se il file è importato altrove.
Area: misura della superficie interna di una figura geometrica, calcolata in base alla sua forma.
Perimetro: misura del contorno di una figura geometrica, ottenuta sommando la lunghezza di tutti i suoi lati.