Övningstenta (avancerad)

Uppgift 1 (3p)

Beskriv vad koden gör och vad som skrivs ut:

```
1. ordlista = ["algoritm", "kompilera", "variabel"]
2. valt_ord = ordlista[ordlista.index("kompilera") - 1]
3. print(valt_ord[3:7])
```

Skriv en kort kommentar för varje rad.

Uppgift 2 (3p)

Skriv en funktion som tar in en radie *r* för basen och en höjd *h* som inparametrar och som returnerar både volymen *V* och ytan *A* av en kon.

Formlerna för en kon är:

Volym:
V = πr^2h

Yta (inklusive basytan):
A = πr(r + l)

där *l* är slantlängden och den kan beräknas som:

l = sqrt(r^2 + h^2)

- Låt användaren mata in en radie *r* för basen och en höjd *h* och kör funktionen. Funktionen ska returnera båda värdena som en tuple.
- Användaren ska kunna välja om hen vill ha resultatet som en decimal eller som en bråkdel. Om användaren väljer bråkdel, omvandla decimalen till en bråk med hjälp av fractions-modulen i Python.
- Implementera lämplig felhantering för att säkerställa att användarens inmatningar är positiva tal. Hantera även andra potentiella fel som kan uppstå.

Uppgift 3 (3p)

Antag att du har tre olika tärningar:

1. En standard tärning med sidorna 1 till 6.
2. En åttasidig tärning med sidorna 1 till 8.

3. En tiosidig tärning med sidorna 1 till 10.

Simulera 1 000 000 kast med varje tärning.

- Beräkna och spara antalet av varje möjligt resultat för varje tärning (t.ex., hur många gånger du fick en 3:a med den åttasidiga tärningen).
- Beräkna den genomsnittliga poängen för varje tärning.
- Bestäm vilken tärning som i genomsnitt gav högst poäng och vilken som gav lägst.
- Spara all information i en lämplig datastruktur och presentera resultaten på ett överskådligt sätt.

Uppgift 4 (4p)

Skapa en funktion som tar in ett heltal *n* som parameter och skriver ut skriver ut ett alternerande mönster av 'x' och 'o' baserat på värdet av *n*.

Mönstret ska bestämmas av värdet *n* enligt följande:

- Om *n* är ett jämnt tal, börja mönstret med 'x'.
- Om *n* är ett udda tal, börja mönstret med 'o'.

Mönstret ska sedan fortsätta att alternera mellan 'x' och 'o' både horisontellt och vertikalt.

Exempel när *n* = 5:

```
o x o x o
x o x o x
o x o x o
x o x o x
o x o x o
```

Exempel när *n* = 6:

```
x o x o x o
o x o x o x
x o x o x o
o x o x o x
x o x o x o
o x o x o x
```

Ditt mål är att skapa en funktion som kan generera dessa mönster för alla positiva heltal *n*.

Uppgift 5 (4p)

Skapa en funktion som tar in ett heltal mellan 1 och 3999 (detta är det högsta värdet man kan representera med klassiska romerska siffror). Funktionen ska returnera det givna talet i form av romerska siffror.

Här är några grundläggande värden för romerska siffror:

```
I - 1
V - 5
X - 10
L - 50
C - 100
D - 500
M - 1000
```

Kom ihåg att det finns speciella regler för hur romerska siffror ska kombineras, till exempel representeras 4 som "IV" (som betyder "ett mindre än fem") snarare än "IIII".

Exempel:

Input: 4 -> Output: IV
Input: 9 -> Output: IX
Input: 58 -> Output: LVIII
Input: 1994 -> Output: MCMXCIV

Uppgift 6 (4p)

Du har följande kod:

```python
class Property:
    def __init__(self, address):
        self.address = address
        self._price = None

    @property
    def price(self):
        return self._price

    @price.setter
    def price(self, value):
        if isinstance(value, (int, float)) and value > 0:
            self._price = value
        else:
            raise ValueError("Price must be a positive number.")

    def __str__(self):
        return f'Property at {self.address} priced at {self.price}'

class Apartment(Property):
    def __repr__(self):
        return f'Apartment at {self.address} priced at {self.price}'

class House(Property):
```

```
    def __repr__(self):
        return f'House at {self.address} priced at {self.price}'

# TO-DO: implementera PropertyOwner
class PropertyOwner:
    ...

# Manuell test av PropertyOwner
owner1 = PropertyOwner("Alice", apartments=["Central St 1", "Hillside
Ave 5"], houses=["Meadow Rd 10"])
owner2 = PropertyOwner("Bob", houses=["Lakeview Rd 2", "Forest Ln 8"])

# Lägg till priser
owner1.apartments[0].price = 500_000
owner1.apartments[1].price = 450_000
owner1.houses[0].price = 900_000
owner2.houses[0].price = 600_000
owner2.houses[1].price = 800_000

print(owner1)
print(owner2)
```

Implementera **PropertyOwner** klassen så att du får liknande utskrift som nedan med det manuella testet:

```
Alice owns:
[Apartment at Central St 1 priced at 500000, Apartment at Hillside Ave
5 priced at 450000]
[House at Meadow Rd 10 priced at 900000]

Bob owns:
[House at Lakeview Rd 2 priced at 600000, House at Forest Ln 8 priced
at 800000]
```