HÖGSKOLAN VÄST

TENTAMEN GPP101

Grundläggande programmering med Python

Datum	16 Februari 2024
Tid	14:00 – 18:00
Examinator	Darwen Al-Taeshi
Lärare Besök	Darwen Al-Taeshi
Hjälpmedel	Kursboken
Antal uppgifter	6
Antal sidor Max	4
poäng	30
Betygsgränser	3: 15 – 19 poäng 4: 20 – 24 poäng 5: 25 – 30 poäng
Övrigt Resultat	Glöm inte att testa din kod. Skriv egna test för att komplettera de givna testfallen.
anslås	senast 8 Mars 2024

Skriv din anonymitetskod i varje inlämnad fil. Lämna in filerna utan att ändra deras namn. Lägg dem i en katalog som har din anonymitetskod som namn.

1. (5p)

En bil färdas med en konstant hastighet av 60 km/h. Skriv en Python-funktion som tar en sträcka (i kilometer) som inmatning och beräknar tiden det tar att färdas den sträckan med den givna farten. Funktionen skall returnera en tuple med timmar och minuter, avrundat till närmaste heltal.

```
def calculate_travel_time(distance):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_travel_time(120)) #(2, 0)
print(calculate_travel_time(70))  #(1, 10)
print(calculate_travel_time(30))  #(0, 30)
```

2. (5p)

Du vill skapa ett program för att beräkna priset på en produkt efter att olika rabatter har tillämpats, baserat på det ursprungliga priset. Skriv en Python-funktion som tar det ursprungliga priset och returnerar det rabatterade priset i SEK enligt följande regler:

Om det ursprungliga priset är högst 50 SEK, tillämpa ingen rabatt.
För priser över 50 SEK men högst 500 SEK, tillämpa en rabatt på 10%.
För priser över 500 SEK men högst 1000 SEK, tillämpa en rabatt på 20%.
För priser över 1000 SEK, tillämpa en rabatt på 30%.

```
def calculate_discounted_price(original_price_sek):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_discounted_price(40))   # Output: 40
print(calculate_discounted_price(100))  # Output: 90
```

3. (5p)

Du utvecklar ett program för att skapa akronymer för fraser. Skriv en Python-funktion som tar en mening som input och genererar en akronym. Akronymen skall bestå av första-bokstäverna i samtliga ord, separerade med punkter, och alla bokstäverna skall vara 'stora' (upper case).

```
def generate_acronym(sentence):
```

Använd följande testfall för att kontrollera din kod:

```
print(generate_acronym("central processing unit"))  # Output: C.P.U
print(generate_acronym("As soon as possible"))      # Output: A.S.A.P
```

4. (5p)

Du arbetar med ett program för att analysera försäljningsdata. Skriv en Python-funktion som tar en lista med månatliga försäljningssiffror och beräknar genomsnittsförsäljningen per månad. Funktionen skall returnera en ny lista som visar hur respektive månads försäljning skiljer sig från den genomsnittliga försäljningsnivån, avrundat till närmsta heltal.

```
def calculate_sales_difference(sales_figures):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_sales_difference([900, 1200, 1300, 1000, 600]))
# Output: [-100, 200, 300, 0, -400]
print(calculate_sales_difference([35, 100, 45]))
# Output: [-25, 40, -15]
```

5. (5p)

Du arbetar med ett program för att analysera säsongsberoende temperaturer. Skriv en Python-funktion som tar en 2D-lista som representerar temperaturer för varje månad av året, organiserade efter säsonger (vår, sommar, höst, vinter). Funktionen skall beräkna och returnera en lista med den genomsnittliga temperaturen för varje säsong. Temperaturmedelvärdena skall avrundas till en decimal.

```
def calculate_seasonal_averages(temperature_data):
```

Använd följande testfall för att kontrollera din kod:

```
temperature_matrix = [[25, 28, 23], [30, 35, 28], [22, 20, 25], [27, 30, 29]]
print(calculate_seasonal_averages(temperature_matrix))
# Output: [25.3, 31.0, 22.3, 28.7]

temperature_matrix = [[15, 18, 21], [27, 31, 35], [33, 31, 25], [20, 15, 10]]
print(calculate_seasonal_averages(temperature_matrix))
# Output: [18.0, 31.0, 29.7, 15.0]
```

6. (5p)

Skapa ett Python-program som läser in en JSON-fil som innehåller information om länder och deras befolkningar. Programmet skall ta ett argument (landets namn). Om det angivna landet finns i filen skall programmet returnera befolkningen. Om landet *inte* finns i filen skall programmet lägga till det landet med standardbefolkningen (1 000 000) i JSON-filen och sedan returnera standardbefolkningen.

Tips: För att undersöka om en variabel är tom kan man använda nyckelordet None.

```
def get_population_or_add_default(country_name):
```

Använd följande testfall för att kontrollera din kod:

```
print(get_population_or_add_default( 'Poland')) # 40525329
print(get_population_or_add_default( 'Japan'))  # 122882843
print(get_population_or_add_default( 'Qatar'))  # 1000000
```