HÖGSKOLAN VÄST

TENTAMEN GPP101

Grundläggande programmering med Python

Datum	21 Augusti 2024
Tid	14:00 – 18:00
Examinator	Darwen Al-Taeshi
Lärare Besök	Darwen Al-Taeshi
Hjälpmedel	Ja, 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 11 september 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 biograf säljer popcorn i olika stora mängder, och priset baseras på popcornens vikt i gram (g). Kostnaden är 0,20 SEK per gram. Skriv en funktion som tar vikten av popcornens vikt i gram som ett heltal och beräknar den totala kostnaden. Funktionen ska returnera en tupel bestående av två värden: det totala beloppet i kronor som ett heltal och de återstående örena som ett heltal. Säkerställ att örevärdet är mindre än 100.

```
def calculate_popcorn_cost_in_sek (grams):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_popcorn_cost_in_sek(750))  # Output: (150, 0)
print(calculate_popcorn_cost_in_sek(76))  # Output: (15, 20)
```

2. (5p)

Ett biluthyrningsföretag beräknar hyreskostnader baserat på antalet dagar en bil hyrs:

- 1 dag kostar $40
- 2 till 5 dagar kostar $35 per dag
- 6 till 10 dagar kostar $30 per dag
- Mer än 10 dagar kostar $25 per dag

Skriv en funktion som tar antalet dagar en bil hyrs som inmatning och returnerar den totala kostnaden.

```
def calculate_rental_cost (days):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_rental_cost(2))  # Output: (70)
print(calculate_rental_cost(4))  # Output: (140)
print(calculate_rental_cost(12)) # Output: (300)
```

3. (5p)

Skriv en funktion som tar en sträng och returnerar en ny sträng där varje tecken upprepas enligt dess position i den ursprungliga strängen. Till exempel upprepas det första tecknet en gång, det andra tecknet upprepas två gånger, och så vidare. Den resulterande strängen ska kombinera alla dessa upprepningar.

```
def repeat_characters (word):
```

Använd följande testfall för att kontrollera din kod:

```
print(repeat_characters("ABC"))  # Output: ABBCCC
print(repeat_characters("Sweden"))  # Output: Swweeeddddeeeeennnnnn
```

4. (5p)

Skriv en funktion som tar en lista av heltal och ett heltal ***d*** som inmatning. Funktionen ska ta bort alla element från listan som är delbara med ***d*** och returnera den modifierade listan. Om inga element är delbara med ***d***, returnera den ursprungliga listan.

```
def remove_divisibles(lst, d):
```

Använd följande testfall för att kontrollera din kod:

```
lst = [2, 4, 5, 10, 15, 20]
d = 5
result = remove_divisibles(lst, d)
print(result)  # Output: [2, 4]
```

5. (5p)

Skriv en Python-funktion som tar en nästlad lista där varje underlista innehåller par av värden som representerar koordinater på ett rutnät. Funktionen ska returnera en ny nästlad lista där varje koordinat är förskjuten med en angiven offset. Offseten är en tupel ***(dx, dy)*** som representerar förändringen i ***x***- och ***y***-koordinaterna.

Exempel på indata: [[[1, 2], [3, 4]], [[5, 6], [7, 8]]] och offset ***(1, -1)***
Exempel på utdata: [[[2, 1], [4, 3]], [[6, 5], [8, 7]]]

```
def shift_coordinates (nested_list, offset):
```

Använd följande testfall för att kontrollera din kod:

```
nested_list = [[[5, 1], [-2, 3]], [[0, 1], [-2, 3]],[[11, 0], [3, 3]]]
offset = (3, -2)
print(shift_coordinates(nested_list, offset))
#output: [[[8, -1], [1, 1]], [[3, -1], [1, 1]], [[14, -2], [6, 1]]]
```

6. (5p)

Skriv en funktion som tar två ***dictionaries*** som inmatning. Den första ***dictionary***, *stock_dict*, innehåller produktnamn som nycklar och deras lagersaldo (hur många som finns i lager) som värden. Den andra ***dictionary***, sales_dict, innehåller produktnamn som nycklar och antalet sålda enheter som värden. Funktionen ska uppdatera stock_dict för att visa vilka nya lagersaldon som erhålls efter försäljningen enligt sales_dict. För varje produkt i sales_dict, minska motsvarande mängd i stock_dict med den sålda mängden. Om den resulterande mängden blir mindre än noll, sätt den till noll. Funktionen ska returnera den uppdaterade stock_dict.

```
def update_stock (stock_dict, sales_dict):
```

Använd följande testfall för att kontrollera din kod:

```
stock = {'apple': 50, 'banana': 30, 'orange': 20}
sales = {'apple': 10, 'banana': 5, 'orange': 25}

print(update_stock(stock, sales))
#output: {'apple': 40, 'banana': 25, 'orange': 0}
```