HÖGSKOLAN VÄST

# TENTAMEN GPP101

## Grundläggande programmering med Python

| | |
|---|---|
| Datum | 10 januari 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<br>4: 20 – 24 poäng<br>5: 25 – 30 poäng |
| Övrigt Resultat | Glöm inte att testa din kod. Skriv egna test för att komplettera de givna testfallen. |
| anslås | senast 1 februari 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)

I ett datorspel får spelarna poäng. Skriv en Python-funktion som tar spelarens poäng som argument och beräknar motsvarande nivå och återstående poäng. Varje nivå är värd 100 poäng, och eventuella återstående poäng bör returneras. Funktionen bör returnera en tuple med två heltal som representerar nivån och återstående poäng.

```
def calculate_level_and_remaining_points(score):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_level_and_remaining_points(250))   # Output:  (2, 50)
print(calculate_level_and_remaining_points(435))   # Output:  (4, 35)
```

2. (5p)

Body Mass Index (BMI) är ett mått på övervikt baserat på längd och vikt. BMI Formeln är kroppsvikten i kg delad med kroppslängden i meter i kvadrat (kg/m2).

Kategorierna definieras enligt följande:
**Underweight**: BMI mindre än 18.5
**Normal weight:** BMI mellan 18.5 och 24.9
**Overweight**: BMI mellan 25 och 29.9
**Obesity**: BMI av 30 eller högre
Skriv en Python-funktion som tar en persons längd (i meter) och vikt (i kilogram) som argument och returnerar motsvarande BMI-kategori.

```
def calculate_bmi_category(height, weight):
```

Använd följande testfall för att kontrollera din kod:

```
print(calculate_bmi_category(1.75, 70)) # Output:  Normal weight
print(calculate_bmi_category(1.8, 89))  # Output:  Overweight
```

3. (5p)

Skriv en Python-funktion som genererar en e-postadress från ett givet fullständigt namn. Det fullständiga namnet ges i formatet "Förnamn Efternamn". E-postadressen bör bestå av de två första bokstäverna i förnamnet följt av de två sista bokstäverna i efternamnet. Funktionen bör returnera den genererade e-postadressen ( `@example.com`).

```
def generate_email(full_name):
```

Använd följande testfall för att kontrollera din kod:

```
print(generate_email("john Doe"))  # Output: Jooe@example.com
print(generate_email("Alex Turner"))  # Output: Aler@example.com
```

4. (5p)

Skapa en Python-funktion som tar två listor med heltal som argument och returnerar en ny lista som endast innehåller de gemensamma elementen i de två listorna. Funktionen bör inte ändra på de ursprungliga listorna.

```
def find_common_elements(list1, list2):
```

Använd följande testfall för att kontrollera din kod:

```
print(find_common_elements([1, 2, 3, 4, 5], [3, 5, 7, 9]))
# Output: [3, 5]

print(find_common_elements([4, 7, 2, 9, 1], [12, 1, 4, 3,7]))
# Output: [4, 7, 1]
```

5. (5p)

Skriv en Python-funktion som tar en matris representerad som en nästlad lista och ett skalärt värde (enskilt värde) och returnerar resultatet av att multiplicera matrisen med det skalära värdet. Funktionen bör inte ändra den ursprungliga matrisen.

```
def scalar_multiply_matrix(matrix, scalar):
```

Använd följande testfall för att kontrollera din kod:

```
# Test case
original_matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(scalar_multiply_matrix(original_matrix, 2))
# Output: [[2, 4, 6], [8, 10, 12], [14, 16, 18]]

original_matrix = [[2, 5, 1], [6, 2, 7], [3, 8, 4]]
print(scalar_multiply_matrix(original_matrix, 3))
# Output: [[[6, 15, 3], [18, 6, 21], [9, 24, 12]]
```

6. (5p)

Skriv ett Python-program som läser två JSON-filer, 'file1.json' och 'file2.json', var och en innehållande en ordlista med deras frekvenser. Programmet bör sammanfoga ordböckerna genom att summera frekvenserna för gemensamma ord och sedan returnera den sammanslagna ordboken.

```
def merge_json_files(file1, file2):
```

Använd följande testfall för att kontrollera din kod:

```
print(merge_json_files("file1.json", "file2.json"))
# file1.json  = { "apple": 10, "banana": 5, "orange": 8, "grape": 12}
# file2.json  = {"apple": 5, "banana": 3, "kiwi": 6, "grape": 8 }
# output{'apple': 15, 'banana': 8, 'orange': 8, 'grape': 20, 'kiwi': 6}
```