Halil_Kolatan

March 17, 2023

1 Coding Case Questions

Halil Kolatan Bu case çalışmasında girdi ve çıktıların daha rahat okunabilmesi için ".py" formatı yerine ".ipynb" formatı kullanılmıştır.

1 . Write a Python program that takes a user's name as input and prints out a personalized greeting.

[1]:
```
name = input("İsminiz: ")
print("Selamlar " + name +"!")
```

```
İsminiz:  Halil
Selamlar Halil!
```

[2]:
```
# Ünlem işaretini istersek değişkene istersek print fonksiyonuna atayabiliriz.
name = input("İsminiz: ") + "!"
print("Selamlar", name)
```

```
İsminiz:  Halil
Selamlar Halil!
```

---

2 . Write a Python function that takes a list of strings as input and returns a new list with all the strings in reverse order.

[3]:
```
# İlk olarak listemizi oluşturup çıktısına bakalım.
liste = ["Bu", "bir", "liste", "denemesidir."]
print(liste)
```

```
['Bu', 'bir', 'liste', 'denemesidir.']
```

[4]:
```
liste = ["Bu", "bir", "liste", "denemesidir."]
yeni_liste = list(reversed(liste))
print(yeni_liste)
```

```
['denemesidir.', 'liste', 'bir', 'Bu']
```

---

3 . Write a Python program that asks the user to input a number and then prints out whether the number is positive, negative, or zero.

[5]:
```
sayi = int(input("Bir sayı giriniz: "))
if sayi < 0:
    print("Girdiğiniz sayı negatif ve", sayi, end=".") # burada değişkenden␣
 ↪sonra yazacağımız string ifadesiyle arasında boşluk olmaması için end␣
 ↪parametresini ekliyoruz.
elif sayi == 0:
    print("Girdiğiniz sayı 0'dır.")
elif sayi > 0:
    print("Girdiğiniz sayı pozitif ve", sayi, end=".")
```

```
Bir sayı giriniz:  985451
```

```
Girdiğiniz sayı pozitif ve 985451.
```

[6]:
```
# Aynı şekilde daha farklı bir döngü yazarak yine aynı sonuca ulaşabiliriz.
sayi = int(input("Bir sayı giriniz: "))
if sayi < 0:
   if sayi == 0:
       print("Girdiğiniz sayı 0'dır.")
   else:
       print("Girdiğiniz sayı negatif ve", sayi, end=".")
else:
   print("Girdiğiniz sayı pozitif ve", sayi, end=".")
```

```
Bir sayı giriniz:  -156465
```

```
Girdiğiniz sayı negatif ve -156465.
```

---

4 . Write a Python function that takes two lists as input and returns a new list that contains only the elements that are common between the two lists.)

[7]:
```
liste_1 = [1,2,3,4,5,6,7,8,9,10]
liste_2 = [1,3,5,7,9,11,13,15]
ortak_liste = set(liste_1).intersection(liste_2) # intersection fonksiyonu ile␣
 ↪bu işlemi kolayca yapabiliriz.
print(ortak_liste)
```

```
{1, 3, 5, 7, 9}
```

[8]:
```
# Aynı işlemi fonksiyon ile de yapabiliriz. Fakat son adımda bu işlemi listeye␣
 ↪çevirmemiz gerekiyor.
# yeni_liste değişkeni ilk olarak aslında bir fonksiyon sonucu olduğu için bunu␣
 ↪listeye çeviriyoruz.
def ortak_deger(liste_1 , liste_2):
    deger = [i for i in liste_1 if i in liste_2]
    return deger

liste_1 = [1,2,3,4,5,6,7,8,9,10]
```

```
liste_2 = [1,3,5,7,9,11,13,15]

yeni_liste = [ortak_deger(liste_1 , liste_2) for i in liste_1 and liste_2]
print(ortak_deger(liste_1 , liste_2))
```

```
[1, 3, 5, 7, 9]
```

[9]:
```
# Gördüğümüz gibi son adımda değişkenimiz liste tipindedir.
type(yeni_liste)
```

[9]:
```
list
```

---

5. Write a Python program that reads a CSV file and prints out the total number of rows in the file.

[10]:
```
import pandas as pd # Bu işlem için ilk olarak pandas kütüphanesini içeri␣
↪aktaralım.
df = pd.read_csv("Sleep_Efficiency.csv")
df # Burada csv dosyasının ilk 5 ve son 5 elemanını görebiliriz. Aynı zamanda␣
↪satır ve sütun sayısını da görebiliyoruz. 452 satır ve 15 sütun.
```

[10]:

	ID	Age	Gender	Bedtime	Wakeup time	\
0	1	65	Female	2021-03-06 01:00:00	2021-03-06 07:00:00	
1	2	69	Male	2021-12-05 02:00:00	2021-12-05 09:00:00	
2	3	40	Female	2021-05-25 21:30:00	2021-05-25 05:30:00	
3	4	40	Female	2021-11-03 02:30:00	2021-11-03 08:30:00	
4	5	57	Male	2021-03-13 01:00:00	2021-03-13 09:00:00	
..	…	…	…	…	…	
447	448	27	Female	2021-11-13 22:00:00	2021-11-13 05:30:00	
448	449	52	Male	2021-03-31 21:00:00	2021-03-31 03:00:00	
449	450	40	Female	2021-09-07 23:00:00	2021-09-07 07:30:00	
450	451	45	Male	2021-07-29 21:00:00	2021-07-29 04:00:00	
451	452	18	Male	2021-03-17 02:30:00	2021-03-17 10:00:00	

	Sleep duration	Sleep efficiency	REM sleep percentage	\
0	6.0	0.88	18	
1	7.0	0.66	19	
2	8.0	0.89	20	
3	6.0	0.51	23	
4	8.0	0.76	27	
..	…	…	…	
447	7.5	0.91	22	
448	6.0	0.74	28	
449	8.5	0.55	20	
450	7.0	0.76	18	
451	7.5	0.63	22	

```
     Deep sleep percentage  Light sleep percentage  Awakenings  \
0                       70                      12         0.0
1                       28                      53         3.0
2                       70                      10         1.0
3                       25                      52         3.0
4                       55                      18         3.0
..                     …                       …          …
447                     57                      21         0.0
448                     57                      15         4.0
449                     32                      48         1.0
450                     72                      10         3.0
451                     23                      55         1.0

     Caffeine consumption  Alcohol consumption Smoking status  \
0                     0.0                  0.0            Yes
1                     0.0                  3.0            Yes
2                     0.0                  0.0             No
3                    50.0                  5.0            Yes
4                     0.0                  3.0             No
..                    …                   …            …
447                   0.0                  0.0             No
448                  25.0                  0.0             No
449                   NaN                  3.0            Yes
450                   0.0                  0.0             No
451                  50.0                  0.0             No

     Exercise frequency
0                   3.0
1                   3.0
2                   3.0
3                   1.0
4                   3.0
..                  …
447                 5.0
448                 3.0
449                 0.0
450                 3.0
451                 1.0

[452 rows x 15 columns]
```

[11]:
```
df.shape # Yine shape metotu ile veri setinin satır ve sütun sayısını␣
↪görebiliriz.
```

[11]:
```
(452, 15)
```

---

6. Write a Python function that takes a string as input and returns a dictionary where each key is a character in the string and the value is the number of times that character appears in the string.

[12]:
```
# Uzun zamandır dictionarylerle çalışmadığım için dictionary guide'na baktığımı␣
 ↪belirtmek isterim.
# Bir süredir sadece data framelerle çalıştığım için bu kısımları çok␣
 ↪hatırlamıyorum.
def görünme_sayısı(string):
    sözlük_sayım = {}

    for kelime in string:
        if kelime in sözlük_sayım:
            sözlük_sayım[kelime] += 1
        else:
            sözlük_sayım[kelime] = 1
    return sözlük_sayım
görünme_sayısı("Bu bir deneme cümlesidir.")
```

[12]:
```
{'B': 1,
 'u': 1,
 ' ': 3,
 'b': 1,
 'i': 3,
 'r': 2,
 'd': 2,
 'e': 4,
 'n': 1,
 'm': 2,
 'c': 1,
 'ü': 1,
 'l': 1,
 's': 1,
 '.': 1}
```

---

7. Write a Python program that reads a large text file and finds the 10 most common words in the file, along with their frequency.

[14]:
```
# Burada William Shakespeare'in "The Project Gutenberg"den alınan Romeo ve␣
 ↪Juliet oyununun ücretsiz versiyonunu textfile çevirerek kullanıyoruz.
from collections import Counter # Counter kütüphanesini ile bu işlemi kolayca␣
 ↪yapabiliriz. İlk olarak Counter kütüphnesini içeri aktaralım.
def kelime_tekrarı(yaygın):
        with open(yaygın) as f: # Buradaki parametleri hatırlamak için␣
 ↪internetten Counter guide'da tekrar baktığımı belirtmek isterim.
                return Counter(f.read().split())
```

```
print("Romeo ve Juliet Oyunundaki en yaygın 10 kelime :", kelime_tekrarı("metin.
 ↪txt").most_common(10))
```

```
Romeo ve Juliet Oyunundaki en yaygın 10 kelime : [('the', 780), ('I', 551),
('and', 541), ('to', 524), ('of', 476), ('a', 458), ('in', 350), ('is', 313),
('my', 304), ('with', 274)]
```

[15]:
```
# Burada prettytable kütüphanesi ile yukarıda aldığımız sonuçları daha rahat␣
 ↪görebilmek için tablo haline getiriyoruz.
from prettytable import PrettyTable

tablo = PrettyTable(["Kelime", "Tekrarlama Sayısı"])
tekrarlama_sayısı = [('the', 780), ('I', 551), ('and', 541), ('to', 524),␣
 ↪('of', 476), ('a', 458), ('in', 350), ('is', 313), ('my', 304), ('with',␣
 ↪274)]

for i in tekrarlama_sayısı:
    tablo.add_row([i[0],i[1]])

print(tablo)
```

```
+--------+-------------------+
| Kelime | Tekrarlama Sayısı |
+--------+-------------------+
|  the   |        780        |
|   I    |        551        |
|  and   |        541        |
|   to   |        524        |
|   of   |        476        |
|   a    |        458        |
|   in   |        350        |
|   is   |        313        |
|   my   |        304        |
|  with  |        274        |
+--------+-------------------+
```

---

8. Write a Python function that takes a list of integers as input and returns a new list that contains all the possible combinations of three integers from the original list (i.e. all possible sets of three numbers).

[16]:
```
# Daha önce böyle bir fonksiyon yazmadığım için direkt olarak itertools␣
 ↪kütüphanesinin özelliğini kullandım.
import itertools

liste = [1,2,3,4,5,6,7,8,9,10,11,12,13]
kombinasyon_list = list(itertools.combinations(liste,3)) # Listedeki sayıların␣
 ↪3'lü tüm olası kombinasyonlarını alıyoruz.
```

```
print(kombinasyon_list)
```

```
[(1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 2, 6), (1, 2, 7), (1, 2, 8), (1, 2, 9),
(1, 2, 10), (1, 2, 11), (1, 2, 12), (1, 2, 13), (1, 3, 4), (1, 3, 5), (1, 3, 6),
(1, 3, 7), (1, 3, 8), (1, 3, 9), (1, 3, 10), (1, 3, 11), (1, 3, 12), (1, 3, 13),
(1, 4, 5), (1, 4, 6), (1, 4, 7), (1, 4, 8), (1, 4, 9), (1, 4, 10), (1, 4, 11),
(1, 4, 12), (1, 4, 13), (1, 5, 6), (1, 5, 7), (1, 5, 8), (1, 5, 9), (1, 5, 10),
(1, 5, 11), (1, 5, 12), (1, 5, 13), (1, 6, 7), (1, 6, 8), (1, 6, 9), (1, 6, 10),
(1, 6, 11), (1, 6, 12), (1, 6, 13), (1, 7, 8), (1, 7, 9), (1, 7, 10), (1, 7,
11), (1, 7, 12), (1, 7, 13), (1, 8, 9), (1, 8, 10), (1, 8, 11), (1, 8, 12), (1,
8, 13), (1, 9, 10), (1, 9, 11), (1, 9, 12), (1, 9, 13), (1, 10, 11), (1, 10,
12), (1, 10, 13), (1, 11, 12), (1, 11, 13), (1, 12, 13), (2, 3, 4), (2, 3, 5),
(2, 3, 6), (2, 3, 7), (2, 3, 8), (2, 3, 9), (2, 3, 10), (2, 3, 11), (2, 3, 12),
(2, 3, 13), (2, 4, 5), (2, 4, 6), (2, 4, 7), (2, 4, 8), (2, 4, 9), (2, 4, 10),
(2, 4, 11), (2, 4, 12), (2, 4, 13), (2, 5, 6), (2, 5, 7), (2, 5, 8), (2, 5, 9),
(2, 5, 10), (2, 5, 11), (2, 5, 12), (2, 5, 13), (2, 6, 7), (2, 6, 8), (2, 6, 9),
(2, 6, 10), (2, 6, 11), (2, 6, 12), (2, 6, 13), (2, 7, 8), (2, 7, 9), (2, 7,
10), (2, 7, 11), (2, 7, 12), (2, 7, 13), (2, 8, 9), (2, 8, 10), (2, 8, 11), (2,
8, 12), (2, 8, 13), (2, 9, 10), (2, 9, 11), (2, 9, 12), (2, 9, 13), (2, 10, 11),
(2, 10, 12), (2, 10, 13), (2, 11, 12), (2, 11, 13), (2, 12, 13), (3, 4, 5), (3,
4, 6), (3, 4, 7), (3, 4, 8), (3, 4, 9), (3, 4, 10), (3, 4, 11), (3, 4, 12), (3,
4, 13), (3, 5, 6), (3, 5, 7), (3, 5, 8), (3, 5, 9), (3, 5, 10), (3, 5, 11), (3,
5, 12), (3, 5, 13), (3, 6, 7), (3, 6, 8), (3, 6, 9), (3, 6, 10), (3, 6, 11), (3,
6, 12), (3, 6, 13), (3, 7, 8), (3, 7, 9), (3, 7, 10), (3, 7, 11), (3, 7, 12),
(3, 7, 13), (3, 8, 9), (3, 8, 10), (3, 8, 11), (3, 8, 12), (3, 8, 13), (3, 9,
10), (3, 9, 11), (3, 9, 12), (3, 9, 13), (3, 10, 11), (3, 10, 12), (3, 10, 13),
(3, 11, 12), (3, 11, 13), (3, 12, 13), (4, 5, 6), (4, 5, 7), (4, 5, 8), (4, 5,
9), (4, 5, 10), (4, 5, 11), (4, 5, 12), (4, 5, 13), (4, 6, 7), (4, 6, 8), (4, 6,
9), (4, 6, 10), (4, 6, 11), (4, 6, 12), (4, 6, 13), (4, 7, 8), (4, 7, 9), (4, 7,
10), (4, 7, 11), (4, 7, 12), (4, 7, 13), (4, 8, 9), (4, 8, 10), (4, 8, 11), (4,
8, 12), (4, 8, 13), (4, 9, 10), (4, 9, 11), (4, 9, 12), (4, 9, 13), (4, 10, 11),
(4, 10, 12), (4, 10, 13), (4, 11, 12), (4, 11, 13), (4, 12, 13), (5, 6, 7), (5,
6, 8), (5, 6, 9), (5, 6, 10), (5, 6, 11), (5, 6, 12), (5, 6, 13), (5, 7, 8), (5,
7, 9), (5, 7, 10), (5, 7, 11), (5, 7, 12), (5, 7, 13), (5, 8, 9), (5, 8, 10),
(5, 8, 11), (5, 8, 12), (5, 8, 13), (5, 9, 10), (5, 9, 11), (5, 9, 12), (5, 9,
13), (5, 10, 11), (5, 10, 12), (5, 10, 13), (5, 11, 12), (5, 11, 13), (5, 12,
13), (6, 7, 8), (6, 7, 9), (6, 7, 10), (6, 7, 11), (6, 7, 12), (6, 7, 13), (6,
8, 9), (6, 8, 10), (6, 8, 11), (6, 8, 12), (6, 8, 13), (6, 9, 10), (6, 9, 11),
(6, 9, 12), (6, 9, 13), (6, 10, 11), (6, 10, 12), (6, 10, 13), (6, 11, 12), (6,
11, 13), (6, 12, 13), (7, 8, 9), (7, 8, 10), (7, 8, 11), (7, 8, 12), (7, 8, 13),
(7, 9, 10), (7, 9, 11), (7, 9, 12), (7, 9, 13), (7, 10, 11), (7, 10, 12), (7,
10, 13), (7, 11, 12), (7, 11, 13), (7, 12, 13), (8, 9, 10), (8, 9, 11), (8, 9,
12), (8, 9, 13), (8, 10, 11), (8, 10, 12), (8, 10, 13), (8, 11, 12), (8, 11,
13), (8, 12, 13), (9, 10, 11), (9, 10, 12), (9, 10, 13), (9, 11, 12), (9, 11,
13), (9, 12, 13), (10, 11, 12), (10, 11, 13), (10, 12, 13), (11, 12, 13)]
```