YZV 102E/104E - Introduction to Programming for Data Science (Python) Lab 7-8

08.05.2025

Res. Asst. Barış Bilen (bilenb20@itu.edu.tr)
Res. Asst. Erhan Biçer (bicer21@itu.edu.tr)
Res. Asst. Uğur Önal (onalug@itu.edu.tr)
Res. Asst. Sümeyye Öztürk (ozturks20@itu.edu.tr)
Res. Asst. Püren Tap (tap23@itu.edu.tr)

1 Exercise 1

In this part, you will complete the following tasks;

1. Take an integer named n from user.
2. Take n inputs that will be name and surname. Create a list that stores tuples of names and surnames. For example, the result is $lst = [('baris', 'bilen'), ('erhan', 'bicer'), ('ugur', 'onal')]$ when $n = 3$. Print the list.
3. Write a lambda function that takes a list of tuples and formats the values in the tuples. The lambda function makes the first item of the tuple capitalized and makes the second item of the tuple upper-cased. Call the lambda function with the list. For example, the result is $lst = [('Baris', 'BILEN'), ('Erhan', 'BICER'), ('Ugur', 'ONAL')]$. Print the list.
4. Write a lambda function that takes a list of tuples and joins the values in the tuples. The lambda function takes a list of tuples and returns a list of strings. The lambda function concatenates the items of tuples separated with a space. Call the lambda function with the list. For example, the result is $lst = ['Baris\ BILEN', 'Erhan\ BICER', 'Ugur\ ONAL']$. Print the list.

1.1 Solution of Exercise 1

The solution is given in Code Snippet 1;

Code Snippet 1: Solution of Exercise 1

```
1  n = int(input("Enter how many names will be entered: "))
2
3  lst = []
4
5  for _ in range(n):
6      name, surname = input().split()
7      lst.append((name, surname))
8
9  print("Name tuples:", lst)
10
11 name_formatting = lambda x: (x[0].capitalize(), x[1].upper())
12
13 lst = list(map(name_formatting, lst))
14
15 print("Formatted name tuples:", lst)
16
17 combine_names = lambda x: " ".join(x)
18
19 lst = list(map(combine_names, lst))
20
21 print("Combined names:", lst)
```

2 Exercise 2

Download the data via this URL: https://drive.google.com/file/d/1NfRuOm87ZF-uVDqPE4iDpjUGQZ7NNRvG/view. In this part, we will complete the following tasks;

1. Download the data.
2. Read the CSV and make a list of dictionaries for each person. CSV is structured as name, gender, age, height, weight, hair color, and eye color.
3. Take a string input which is a name.
4. Take a string input that is the gender preference of the user. It could be 'male' or 'female' for the question's simplicity. Other answers will not be accepted.
5. Take two integers that will be the minimum and the maximum age preference. The minimum age should be greater than or equal to 18 and the maximum age should be less than or equal to 90.
6. Take an integer from the user that will be the minimum height preference.
7. Take an integer from the user that will be the maximum weight preference.
8. Take a string from the user that will be the hair color preference. The options are 'brown', 'black', 'yellow', 'red', 'green', 'blue', 'pink', and 'purple'. Others are not accepted for simplicity.
9. Take a string from the user that will be the eye color preference. The options are 'brown', 'black', 'blue', 'green', or 'hazel'. Others are not accepted for simplicity.
10. Using filter and lambda functions filter the options by gender preference and print the remaining number of candidates.
11. Using filter and lambda functions filter the options by age preference and print the remaining number of candidates.
12. Using filter and lambda functions filter the options by height preference and print the remaining number of candidates.
13. Using filter and lambda functions filter the options by weight preference and print the remaining number of candidates.
14. Using filter and lambda functions filter the options by hair color preference and print the remaining number of candidates.
15. Using filter and lambda functions filter the options by eye preference and print the remaining number of candidates.
16. Finally print the filtered candidates.

2.1 Solution of Exercise 2

The solution is given in Code Snippet 4;

Code Snippet 2: Solution of Exercise 2

```python
people = []
with open("data.csv", "r") as file:
    for line in file.readlines():
        name, gender, age, height, weight, hair_color, eye_color = line.strip().split(",")
        people.append({
            "name": name,
            "gender": gender,
            "age": int(age),
            "height": int(height),
            "weight": int(weight),
            "hair_color": hair_color,
            "eye_color": eye_color
        })
name = input("Name: ")

preferred_gender = input("Preferred gender: ")
accepted_genders = ["male", "female"]
while preferred_gender not in accepted_genders:
    print(f"Please enter one of accepted genders: {accepted_genders}")
    preferred_gender = input("Preferred gender: ")

min_age = int(input("Min age (>= 18): "))
while min_age < 18:
    print("Minimum age should be greater than or equal to 18")
    min_age = int(input("Min age (>= 18): "))

max_age = int(input("Max age (<= 90): "))
while max_age > 90:
    print("Maximum age should be less than or equal to 90")
    max_age = int(input("Max age (<= 90): "))

min_height = int(input("Min height: "))
max_weight = int(input("Max weight: "))

accepted_hair_colors = ["brown", "black", "yellow", "red", "green", "blue", "pink", "purple"]
preferred_hair_color = input("Preferred hair color: ")
while preferred_hair_color not in accepted_hair_colors:
    print(f"Please enter one of accepted hair colors: {accepted_hair_colors}")
    preferred_hair_color = input("Preferred hair color: ")
```

```
40
41  accepted_eye_colors = ["brown", "black", "green", "blue", "hazel"]
42  preferred_eye_color = input("Preferred eye color: ")
43  while preferred_eye_color not in accepted_eye_colors:
44      print(f"Please enter one of accepted eye colors: {accepted_eye_colors}")
45      preferred_eye_color = input("Preferred eye color: ")
46
47  filtered_people = list(filter(lambda x: x["gender"] == preferred_gender, people))
48  print("Gender filtered count = ", len(filtered_people))
49
50  filtered_people = list(filter(lambda x: min_age <= x["age"] <= max_age, filtered_people))
51  print("Age filtered count = ", len(filtered_people))
52
53  filtered_people = list(filter(lambda x: min_height <= x["height"], filtered_people))
54  print("Height filtered count = ", len(filtered_people))
55
56  filtered_people = list(filter(lambda x: x["weight"] <= max_weight, filtered_people))
57  print("Weight filtered count = ", len(filtered_people))
58
59  filtered_people = list(filter(lambda x: x["hair_color"] == preferred_hair_color, filtered_people))
60  print("Hair color filtered count = ", len(filtered_people))
61
62  filtered_people = list(filter(lambda x: x["eye_color"] == preferred_eye_color, filtered_people))
63  print("Eye color filtered count = ", len(filtered_people))
64
65  print("Fully filtered people:")
66  print(filtered_people)
```

3 Exercise 3

In this part, you will complete the following tasks;

1. Write a lambda function for addition.
2. Write a lambda function for subtraction.
3. Write a lambda function for multiplication.
4. Write a lambda function for division.
5. Write a lambda function for factorial in a recursive way.
6. Write a lambda function for power in a recursive way. The exponent will be bigger than or equal to 0.
7. Write a lambda function for log, you can use the math library.
8. Write a code for calculating.
    - Users call the functions in the following format: "char param1 param2..." char will be the first character(one of "asmdfpl") of the name of the function. After char, each parameter of the function will be added with space in between.
    - The procedure will continue until the user enters a "-1" or keyboard interrupt.
    - In this procedure, you will try to catch all the errors and print proper messages using raise, assert, etc. Try to catch every error.

3.1 Solution of Exercise 3

The solution is given in Code Snippet 3;

Code Snippet 3: Solution of Exercise 3

```
import math

addition = lambda x, y: x + y
subtraction = lambda x, y: x - y
multiplication = lambda x, y: x * y
division = lambda x, y: x / y
factorial = lambda x: 1 if x == 0 else x * factorial(x - 1)
power = lambda x, y: 1 if y == 0 else x * power(x, y - 1)
log = lambda x, y: math.log(x, y)

while True:
    try:
        params = input()

        if params == "-1":
            break

        params = params.split()

        operator_characters = list("asmdfpl")
        assert params[0].lower() in operator_characters, f"Not a valid option please select from {operator_charact
        if params[0].lower() == "a":
            assert len(params) == 3, "The number of params is not correct"
            print(addition(int(params[1]), int(params[2])))
        elif params[0].lower() == "s":
            assert len(params) == 3, "The number of params is not correct"
            print(subtraction(int(params[1]), int(params[2])))
        elif params[0].lower() == "m":
            assert len(params) == 3, "The number of params is not correct"
            print(multiplication(int(params[1]), int(params[2])))
        elif params[0].lower() == "d":
            assert len(params) == 3, "The number of params is not correct"
            assert int(params[2]) != 0, ValueError("Zero division")
            print(division(int(params[1]), int(params[2])))
        elif params[0].lower() == "f":
            assert len(params) == 2, "The number of params is not correct"
            if int(params[1]) < 0:
                raise ValueError("Function domain error for factorial")
            print(factorial(int(params[1])))
```

```
40          elif params[0].lower() == "p":
41              assert len(params) == 3, "The number of params is not correct"
42              assert int(params[2]) >= 0, "Out of scope for power operator"
43              if int(params[1]) == 0 and int(params[2]) == 0:
44                  raise ValueError("Function domain error for factorial")
45              print(power(int(params[1]), int(params[2])))
46          elif params[0].lower() == "l":
47              assert len(params) == 3, "The number of params is not correct"
48              assert int(params[1]) >= 0, ValueError("Function domain error for log operator for the fi
49              assert int(params[2]) > 0, ValueError("Function domain error for log operator for the sec
50              print(log(int(params[1]), int(params[2])))
51
52      except AssertionError as error:
53          print("Oops something happened with the assertion! The error is:", error)
54      except ValueError as error:
55          print(f"There is an error with at least one value written and the error is {error}")
56      except Exception as error:
57          print("Oops something happened! The error is:", error)
58      except KeyboardInterrupt:
59          print("Keyboard interrupt is detected, terminating...")
60          break
61
```

---

4 Exercise 4

In this part, you have a folder and documents in it. You need to download the folder from https://drive.google.com/drive/folders/1K5bN3-uncxbGle6bCapXZKe2_MQqS7A1.

- The name format of the documents are fixed and it is like "doc_001.txt".
- The number is between 0 and 999.
- The file extensions are [".*txt*", ".*csv*", ".*png*", ".*svg*", ".*jpg*", ".*dat*", ".*py*", ".*c*", ".*cpp*"].
- For each extension, open the file and close.
- You need find the number of files for each extensions without getting any errors.
- You should use try, except block.
- You can use *zfill* for the string operation.
- sys and os libraries are forbidden.

4.1 Solution of Exercise 4

The solution is given in Code Snippet 4;

Code Snippet 4: Solution of Exercise 4

```
extensions = [".txt", ".csv", ".png", ".svg", ".jpg", ".dat", ".py", ".c", ".cpp"]
dic = {extension: 0 for extension in extensions}

for extension in extensions:
    for i in range(1000):
        name = f"data/doc_{str(i).zfill(3)}{extension}"
        try:
            open(name, "r").close()
        except:
            continue
        else:
            dic[extension] += 1
            print(name)

print(dic)
```