Activity – concatenation, repetition, functions and methods

In **A11-operations.py** do the following - Follow Q&A as in you did in bash

a Define these variables

```
P=[110,100,90,67,89] # blood pressure
names=['maria','gio','dome','anna','caty'] #patients
s='abracadabra'
D={'a':'@','e':3,'i':1,'o':0}
```

b Print to screen the max blood pressure recorded in in P. Use a function to find the maximum.

c Calculate the average blood pressure - sum of elements/number of elements. Use two built-in functions to calculate the average and print the result to screen.

d List names contains patients' names, whose blood pressures are recorded in list P.
Use lists names and P, and the index method to find the blood pressure of gio. Assume you do not know where gio is located in list names. Print to screen:

```
The blood pressure of gio is 100
```

Do not use concatenation.
Hint: Use the index method and find the index referencing gio in list names and use the index to extract the corresponding blood pressure value in P. Print to screen.

Activity – concatenation, repetition, functions and methods

In **A11-operations.py**

e Count the number of times 'a' occurs in s and print it to screen. Use a method to count.

f Produce the following output by using one print function with concatenation and repetition

```
----------
HHHeeloooo
----------
```

Remember the \n string character in a print statement

g Use dictionary D and concatenation to translate ciao in c1@0. Access values in D for characters i a and o. Be careful – you should concatenate string objects.

```
110
91.2
The blood pressure of gio is 100
5
----------
HHHeeloooo
----------
c1@0
```

Activity - list methods

In **A11-list.py** do the following:

a. Define this list:

```
Lzoo=['pangolin','cobra','chicken','lion','elephant']
```

b. Make a copy of list Lzoo, and name the copy Lzoo1, so that if you modify Lzoo, Lzoo1 remains unchanged.

Use list Lzoo for all the following questions

c. The chicken got eaten by the fox. Replace the chicken with a fox. Use item assignment

```
listname[index]=newvalue
```

Print modified list to screen

d. Add another elephant to the end of list Lzoo. Print modified list to screen

e. Count the number of elephants in Lzoo, and print the count to screen

f. Extend the list by ["zebra","koala"]. Print modified list to screen

g. Reverse the list Lzoo. Print modified list to screen

h. Insert *emu* at index 2. Print modified list to screen

Activity - list methods

In **A11-list.py** do the following:

```
list.pop(index)
list.remove(item)
```

i. Remove an item from the list Lzoo, and assign the deleted value to variable var. Use the appropriate method. Print the modified list and variable var to screen

j. Remove an item of your choice. No value should be returned.

k. Print Lzoo1 to screen. Did it get modified? If yes, fix this in point b.

Run the script

Activity - dictionary

In a script called **A11-dict.py** do the following:

a. Define this dictionary

```
D={"orange":3, "blue":4, "black":5}
```

b. Make a copy of dictionary D and assign the copy to variable D1, so that if you modify D, D1 remains unchanged.

Use dictionary D for the following questions

c. Print to screen the number of key:value pairs. Use a function to obtain the number of items

d. Add this key:value pair **"purple":4** by using this syntax

```
d[key]=value
```

and print the updated dictionary to screen

e. Add this key:value pair **"red":6** by using the update method

Activity - dictionary

In **A11-dict.py**

f. Remove the last key:value pair. Use popitem, and assign the returned tuple to variable t. Print the tuple and the updated dictionary to screen.

g. Remove the key:value pair **"orange":3**. Use the pop method and assign the returned value to variable v. Print v and the updated dictionary to screen.

h. Remove **"blue":4** by using del d[key]

i. print to screen D1. Did it get modified? If yes, fix this in point b.

string methods

In a script called **A11-string.py**

a. Define this string

```
s="I like school"
```

b. Make string s uppercase and assign the modified string to s1
Print the modified string to screen

c. Produce the following output by using string methods in one line of code.

```
i LIKE school
```

d. Define this list

```
L=['co','o','l']
```

Create this string COOL out of list L, and print it to screen.
Use string methods and one line of code.

More on patients

The provided script **A11-patients.py** defines a dictionary which contains information on temperature, and pressure recorded for two patients in different months.

```
D={'maria': {'temperature':[80,70],
'pressure':[110,90,95],'month':['april','may','june']},
   'dome': {'temperature':[60,71,78],
'pressure':[120,110,98],'month':['april',’may','july']}}
```

Edit the script and do the following:

a There are two errors in this dictionary that should be fixed.
- The temperature recorded in June for maria is missing.
  Append the 75 to the temperature values.

- the last month for dome is wrong. Instead of july, it should be june.
  Access values in D and fix it

  Print the updated dictionary to screen

b Print to screen the max recorded temperature of dome. Use a function and access values in D

More on patients

The provided script **A11-patients.py** defines a dictionary which contains information on temperature, and pressure recorded for two patients in different month.

```
D={'maria': {'temperature':[80,70,75],
'pressure':[110,90,95],'month':['april','may','june']},
   'dome': {'temperature':[60,71,78],
'pressure':[120,110,98],'month':['april',’may','july']}}
```

c Print to screen the pressure value of maria recorded in may. Use a method and access values in D.

d Use type conversion, access value in D, and print to screen:

`temperature pressure month`

do not use concatenation

```
{'maria': {'temperature': [80, 70, 75], 'pressure': [110, 90, 95], 'month':
['april', 'may', 'june']}, 'dome': {'temperature': [60, 71, 78], 'pressure': [120,
110, 98], 'month': ['april', 'june', 'june']}}
78
90
temperature pressure month
```

Nested types

In a script called **A11-nested.py** do the following

a. Define this dictionary

```
# If you have a long line of code that you want to break up among multiple lines, you can use \

D={'s1':'GACTC', \
's2':'134,256',\
'd1':{'tuple1':(1,2,3),'L1':[1,2,3,4,5,6]},\
'L2':['a','b','c','d'],\
'L3':[{'d3':'you made it'},'bravo']}
```

b. Access values in D and apply a method to obtain:

```
BRAVO YOU MADE IT
```

c. Use append to add the character ‘e’ to the list associated to the key `'L2'`.
Keep in mind that a list and dictionary are both mutable. Print modified D to screen

d. Sum the numbers in the tuple associate with nested-key `'tuple1'`, and print result to screen

Nested types

In **A11-nested.py**

e. Extend the list associated to key 'list1' by this list [7,8] to obtain [1,2,3,4,5,6,7,8]
Print to screen the updated dictionary

f. Sum the first 4 elements of the list associated to key 'list1'. Use slicing and a function.
Do not hardcode. Print result to screen. It should be 10

g. Sum the last 3 elements of list associated to key 'list1'. Use slicing and a function.
Do not hardcode. Print result to screen.

h. Access value 'G,A,C,T,C', convert it to list ['G', 'A', 'C', 'T', 'C '] , and substitute the string with the list. In the dictionary the list should be the new value associated to key 'string1'

Nested types

In **A11-nested**.py

i. Access the nested value (1,2,3) and convert it to list [1,2,3]. Print the list to screen

j. Access the nested value [1,2,3,4,5,6,7,8] and convert it to tuple (1,2,3,4,5,6,7,8) .
Print the tuple to screen

k. Access value ['a','b','c','d’] and convert it to string 'ABCD' and substitute the string with the list. In the dictionary the string should be the new value associated to key 'list2'

Submit to A11

- **A11-operations.py**
- **A11-list.py**
- **A11-dict.py**

- **A11-string.py - optional**
- **A11-patients.py - optional**
- **A11-nested.py - optional**