M.Sc. Degree ID: E7321008

Continuous Assessment - 1

CSC 250: Python Programming

Part - A

1) Rocket Launch Countdown

- Assigining a variable 'start' to get user's start value in integer as input.
- Assigning a variable 'stop' to get stop value as integer from user as ~~int~~ input.
- start = 5 ; stop = 1 for example, Until the stop value is less than or equal to the start value, the start value should be printed and decremented. while stop<= start:

```
stop <= start : #(stop = 1) <= (start = 5) :
                                              start = 4
print(start) # prints(5), prints(4).          start = 3
             prints(3), prints(2)             start = 2
                        prints(1).            start = 1
start -= 1   # [5-1=4], [4-1=3], [3-1=2] [2-1 = 1]
#start = 5-1 => 4 ->
```

- The stop value remains 1 until the start value decrements to 1.
- After the while loop the print statement outside the loop will be executed.

```
Enter the start count :5
Enter the stop count :1
Count Down starts
5
4
3
2
1
Ready to launch
```

2) Random number guessing :

- Importing a function called 'random' to generate random values.
- Creating a variable 'guesses' and assigning it to 5 because we are giving only 5 chances for a player.
- Creating a variable 'comp-guess' and assigning it to random function 'randint' to generate integer values from 1 to 100.
- for i in range(1, guesses+1): # i = 1, 2, 3, 4, 5

(The condition inside the loop runs five times.)

- Getting user input as integer (number between one to 100). and assigning it to variable 'player'.
- If the player's guess is equal to the computer's guess, the player wins. if (player == comp_guess): player wins

Break -> stops the iteration if the guess is true.

- Else if player's guess is less than comp-guess the computer prints a clue to the player. elif (player < compguess)
- Else if player's guess is greater than comp-guess the print statement below the condition gets executed.
- To find the number of guesses left ⇒

guesses - iteration

5 - 1 = 4
5 - 2 = 3
5 - 3 = 2
5 - 4 = 1
5 - 5 = 0

- Iterating the final i value and incrementing it so that the if condition after it gets executed. i = 5+1 ⇒ 6.

If (i == 6): # the player has lost the game as there are 0 no: of guesses ~~left~~ left.

```
Enter your guess (1-100) : 40
Computer guess is Higher
you have 4 guesses left
Enter your guess (1-100) : 50
Computer guess is Lower
you have 3 guesses left
Enter your guess (1-100) : 45
Computer guess is Higher
you have 2 guesses left
Enter your guess (1-100) : 46
 You won
```

3) Substitution Cipher

- Assigning a variable 'encrypt' to an empty string.
- Assigning a variable 'decrypt' to an empty string.
- ~~Assig~~ Declaring a variable 'string_1' to get plain text from user and converting to uppercase.
- Declaring a variable 'string_2' to get encrypted text from ~~from~~ user & converting to lowercase.
- Creating a list of 26 alphabets in uppercase & assigning it to a variable 'letters'.

letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
18 19 20 21 22 23 24 25

(Index value)

- Creating a list of 26 alphabets in reverse order in lowercase & assigning it to a variable 'code'.

code = ['z', 'y', 'x', 'w', 'v', 'u', 't', 's', ~~'r'~~, 'q', 'p', 'o', 'n', 'm', 'l', 'k', 'j', 'i', 'h', 'g',
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

'f', 'e', 'd', 'c', 'b', 'a']
20 21 22 23 24 25

(Index value)

For example: string_1 = BOOK ~~##~~ Index position in letters => 1, 14, 14, 10

string_2 = yllp # in code => 1, 14, 14, 10

↓

(encrypted text)

- Access the ~~each~~ value of plain text 'string_1' using for loop. The loop ends till the full text is iterated.

```
for i in string_1 :   # i = B, O, O, K
    if i in letters :   # i = B ; True   letters = [.. B ... ]
                                          1 → (index value)
```

- If value of 'i' is in letters then the statement below will get executed.
  - Create a variable 'index' & storing the index value of 'i' in 'letters' to it.

```
index = letters.index(i)   # 1st iteration  i = B
```

```
encrypt = encrypt + code[index]   # ~~code[1] = y~~
```

~~#~~ # Index value of alphabets inside variable 'code'.

```
# code[1] ⇒ y
code[2] ⇒ l
code[3] = l
code[4] = p
```

encrypt = yllp

⇒ index = 1
2nd iteration i = O
index = 14
3rd iteration i = O
index = 14
4th iteration i = K
index = 10

```
decrypt = decrypt + letters[index]   # decrypt = B O O K
```

- Comparing 'string_2' is equal to encrypt. If they are equal 'string_2' is encoded version of string_1.
- Else, 'string_2' is not encoded version of string_1.
- Printing the plain text and the correct encrypted text found.

```
String 1 : ABCD
String 2 : ZYXW
String 2 is the encoded version of String 1
Plain Text :  ABCD
Encrypted Text  is :  zyxw
```

4) Rock, Paper & Scissors

- Creating a variable 'score_a' & assigning it to zero for Player A. `score_a = 0`
- Declaring a variable 'score_b' & assigning it to zero for Player B. `score_b = 0`
- Assigning value 15 to variable 'game' because total no: of games to be played is 15. `game = 15`
  - for i in range (1, game + 1): # i= 1,2,3...15

The condition inside the loop gets executed 15 times.

condition inside loop:

Player Inputs ⇒ A B	Score goes to
Paper > Rock ⇒ Score_a = 0+1 ⇒ 1	A score_a = 1
Rock < Paper → Score_b += 1 → Score_b = 0+1 ⇒ 1	B Score_b = 1
Rock > Scissors → score_a+=1 ⇒ score_a = 1+1 ⇒ 2	A score_a = 2
Scissors < Rock → score_b+=1 ⇒ score_b = 1+1 ⇒ 2	B score_b = 2
Scissors > Paper → Score_a+=1 ⇒ score_a = 2+1 ⇒ 3	A Score_a = 3
Paper < Scissors → Score_b+=1 ⇒ score_b = 2+1 ⇒ 3	B score_b = 3

For example: Total score: score_a = 3

score_b = 3

- The loop runs 15 time as there are 15 iterations in the loop. The conditions above will also be executed 15 time.

if (Player A == Player B): # no score
∴ not iteration in score.

- If the input of both players is equal there wont be any score credited.

- After 15 iterations, the following conditions outside the loop will be executed. For example,

  - score_a = 3 score_b = 3

if score_a is greater than score_b ⇒ 'Player A wins'

3 > 3 ⇒ False

elif score_a is less than score_b ⇒ 'Player B wins'

3 < 3 ⇒ False

elif score_a is equal to score_b ⇒ 'Draw Match'

3 == 3 ⇒ TRUE

- The output will be printed only if the condition is true.

```
--------------------------------------------------------------------------------
Game 1
Player A : paper
Player B : paper
No score, Try Again
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 2
Player A : rock
Player B : paper
Paper covers rock . Player B wins
Player B Score = 1
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 3
Player A : scissors
Player B : paper
Scissors cuts Paper. Player A wins
Player A Score = 1
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 4
Player A : rock
Player B : scissors
Rock breaks Scissors. Player A wins
Player A Score = 2
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 5
Player A : paper
Player B : rock
Paper covers rock . Player A wins
Player A Score = 3
```

```
Game 6
Player A : rock
Player B : paper
Paper covers rock . Player B wins
Player B Score = 2
------------------------------------------------------------------------
------------------------------------------------------------------------
Game 7
Player A : scissors
Player B : rock
Rock breaks Scissors. Player B wins
Player B Score = 3
------------------------------------------------------------------------
------------------------------------------------------------------------
Game 8
Player A : paper
Player B : rock
Paper covers rock . Player A wins
Player A Score = 4
------------------------------------------------------------------------
------------------------------------------------------------------------
Game 9
Player A : rock
Player B : scissors
Rock breaks Scissors. Player A wins
Player A Score = 5
------------------------------------------------------------------------
------------------------------------------------------------------------
Game 10
Player A : rock
Player B : scissors
Rock breaks Scissors. Player A wins
Player A Score = 6
```

```
Game 11
Player A : paper
Player B : rock
Paper covers rock . Player A wins
Player A Score = 7
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 12
Player A : scissors
Player B : paper
Scissors cuts Paper. Player A wins
Player A Score = 8
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 13
Player A : rock
Player B : paper
Paper covers rock . Player B wins
Player B Score = 4
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 14
Player A : paper
Player B : scissors
Scissors cuts Paper. Player B wins
Player B Score = 5
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 15
Player A : rock
Player B : scissors
Rock breaks Scissors. Player A wins
Player A Score = 9
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 14
Player A : paper
Player B : scissors
Scissors cuts Paper. Player B wins
Player B Score = 5
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Game 15
Player A : rock
Player B : scissors
Rock breaks Scissors. Player A wins
Player A Score = 9

_____________________________________________
_____________________________________________


Player A Total Score :  9
Player B Total Score :  5
Player A wins
```

5) BMI Calculation

- ~~Declaring~~ • Declaring variable 'weight' to get input of user's weight.
- If the weight is a negative value the the user have to reenter the weight.

For example: weight = -20

while weight < 0: # -20 < 0 ⇒ True

if weight < 0: # user have to re-enter weight.

- The condition inside the while loop will get executed ~~on~~ until the value of weight is less than zero.
- Else if the weight positive or greater than zero, the user will input the height value which is assigned to the variable 'height'.

For example: weight = 50 ; height = 1.52

- Calculating BMI, by assigning a variable BMI to its formula: BMI = weight/(height * height)

$$BMI = 50/(1.52 * 1.52)$$

$$BMI = 21.64$$

- Printing the resulting BMI.
- Comparing value of BMI with three conditions.

- If BMI is greater than 25. BMI is overweight.

  # if BMI > 25 :

  # 21.64 > 25 ⇒ False

  goes to next condition

- If BMI >= 18 and BMI < 25 :

  Elif BMI is between 18 to 25, BMI is normal.

  21.64 >= 18 and 21.64 < 25

  #TRUE #TRUE

- Else if the above condition becomes false, it checks next condition.

  elif BMI is less than 18, BMI is underweight.

  BMI < 18 :

  # 21.64 < 18 ⇒ False

- The output will be printed if any one of the condition is true.

```
Enter the weight in kg : -50
Weight can't be negative, Please re-enter
Enter the weight in kg : 50
Enter the height in meters : 1.52
BMI IS :  21.641274238227147
Estimated BMI is: Normal
```

6) Addition Game

- Importing 'random' function to generate random values.
- Declaring a variable 'count' & assigning a value '1' for counting the consecutive rows answered.
- count = 1
  - while count is less than or equal to 3, the conditions insid the loop will be executed.
  - Inside the loop, we assign two variables a, b to random function for a range of 1 to 100.
  - c = a+b # adding two variables a & b which ~~for~~ generates random numbers in each iteration and assigning it to variable 'c'.
  - In the print statement, asking the user for the sum of two variables a & b. & getting input using variable 'answer'.

For example: a = 2 b = 40

count = 1

while count <= 3 : TRUE

a b

c = a+b # 2 + 40

c = 42

user input ⇒ answer = 42

- Comparing the calculated value ~~in~~ in 'c' & ~~user~~ user input value in 'answer'
- If 'answer' is equal to 'c', # the user has got 1 correct in a row.

- The value of count gets incremented.

count += 1 ⇒ count = 1 + 1 = 2

count = 2

- In the print statement, we use 'count - 1' to tell the player has answered for a certain number of rows. Now the player has got count - 1 correct in a row.

2 - 1 = 1

- Again the count value goes to the loop and checks whether its less than or equal to three.

while count <= 3; # count = 2; <= 3

TRUE

- Now variable a & b generates two random numbers again and asks the user to input the answer for the sum.

a = 25 b = 43

c = a + b ⇒ 25 + 43

c = 68

user Input ⇒ answer = 72

- If 'answer' is not equal to 'c', we print that its an 'incorrect' answer & display the expected answer.

- And now count = 1 because we have to get three consecutive correct answers in a row.

- Now the count value goes to the loop and executes.

- The loop ends until the player answers correctly 3 consecutive times in a row.

```
What is  2 + 2 ?
Your answer : 4
Correct! You've gotten 1 correct in a row
_____________________________________________________________________
What is  3 + 5 ?
Your answer : 2
Incorrect! Expected answer is :  8
_____________________________________________________________________
What is  8 + 5 ?
Your answer : 12
Incorrect! Expected answer is :  13
_____________________________________________________________________
What is  1 + 9 ?
Your answer : 10
Correct! You've gotten 1 correct in a row
_____________________________________________________________________
What is  9 + 7 ?
Your answer : 17
Incorrect! Expected answer is :  16
_____________________________________________________________________
What is  7 + 9 ?
Your answer : 16
Correct! You've gotten 1 correct in a row
_____________________________________________________________________
What is  6 + 5 ?
Your answer : 11
Correct! You've gotten 2 correct in a row
_____________________________________________________________________
What is  1 + 3 ?
Your answer : 4
Correct! You've gotten 3 correct in a row
You've mastered addition
```

1) Checking Infected Cases :

- Assigning variable 'Evermore' to list of infected cases per day for a week in location Evermore.
- Assigning variable 'Vanguard_city' to a list of infected cases per day for a week in location vanguard_city.
- Assigning variable 'Excelsior' to a list of infeted cases per day for a week in location Excelsior.

```
Evermore = [1, 1, 1, 1, 1, 1, 1]
           0  1  2  3  4  5  6
vanguard_city = [1, 2, 3, 4, 5, 6, 7]
                0  1  2  3  4  5  6
Excelsior = [1, 1, 2, 3, 5, 8, 13]
            0  1  2  3  4  5  6
```

→ Index positions

- Declaring variable 'Evermore_length' to length of 'Evermore' using 'len()' method.
- Declaring variable 'Vanguard_length' to length of 'Vanguard_city' using 'len()' method.
- Declaring variable 'Excelsior_length' to length of 'Excelsior' using 'len()' function.

```
Evermore_length = len(Evermore)  # 7
Vanguard_length = len(Vanguard_city)  # 7
Excelsior_length = len(Excelsior)  # 7
```

- Declaring variable 'Evermore_sum' to zero # before calculation to find the total no: of infected in Evermore.
- Declaring variable 'Vanguard_sum' to zero before calculation to find the total no: of infected in Vanguard_city.

- Declaring variable 'Excelsior_sum' to zero, before calculation, to find total no: of infected in 'Excelsior'.
- Declaring variable 'i' to zero for the iteration value.

i) Finding total no: of infection in Evermore:

Evermore = [1 , 1 , 1 , 1 , 1 , 1 , 1]

Index positions → 0 1 2 3 4 5 6

Evermore_Sum = 0 ; i = 0

- Until 'i' is less than Evermore_length, the conditions inside the loop will be executed.

while i < Evermore_length (accessing the index values of Evermore)

i = 0 < 7 ⇒ Evermore_Sum = Evermore[i] + Evermore_Sum

i+=1 ⇒ i = 0+1=1 Evermore_Sum = Evermore[0] + 0 ⇒ 1 + 0 = 1

Evermore_Sum = 1

i = 1 < 7 ⇒ Evermore_sum = Evermore[1] + 1 ⇒ 1+1 = 2

i+=1 ⇒ 1+1=2 Evermore_Sum = 2

i = 2 < 7 ⇒ Evermore_Sum = Evermore[2] + 2 ⇒ 1+2 = 3

i+=1 → 2+1=3 Evermore_Sum = 3

i = 3 < 7 ⇒ Evermore_Sum = Evermore[3] + 3 ⇒ 1+3 = 4

i+=1 → 3+1=4 Evermore_Sum = 4

i = 4 < 7 ⇒ Evermore_Sum = Evermore[4] + 4 ⇒ 1+4 = 5

i+=1 → 4+1=5 Evermore_Sum = 5

i = 5 < 7 ⇒ Evermore_Sum = Evermore[5] + 5 ⇒ 1+5 = 6

i+=1 → 5+1=6 Evermore_Sum = 6

i = 6 < 7 ⇒ Evermore_Sum = Evermore[6] + 6 ⇒ 1+6 = 7

i+=1 → 6+1=7 Evermore_Sum = 7

i = 7 < 7 ⇒ FALSE; The Loop ends.

- Printing the ~~total~~ final value of Evermore_sum as the total number of infected in Evermore.
- ii) Finding total no: of infections in Vanguard_city and Excelsior using the same logic which was used to find total no: of infections for Evermore.
- After repeating the above steps in 'i', we are calculating the number of infections per daily at each location.
- Again assigning the value of 'i' to zero, for iteration.
- Since the length of all three locations are '7', we keep the range till 7.
- Until the iteration is less than 7 the condition inside the loop will be executed.

  while $i < 7$ # $i = 0, 1, 2, 3, 4, 5, 6$.
- Declaring variable 'Per' to calculate total infections per day at each location.

  while $0 < 7$:

$i = 0 \Rightarrow$ Per = Evermore[i] + Vanguard[i] + Excelsior[i]

$\Rightarrow$ Evermore[0] + vanguard[0] + Excelsior[0] = 1+1+1

i+=1 = 0+1 = 1 | Per = 3 → For day 1 in each location

$i = 1 \Rightarrow$ Per = Evermore[1] + vanguard[1] + Excelsior[1] = 1+2+1

i+=1 = 1+1 = 2 | Per = 4 → For day (i+1) ⇒ day 2

$i = 2 \Rightarrow$ Per = Evermore[2] + vanguard[2] + Excelsior[2] = 1+3+2

i+=1→2+1=3 | Per = 6 → For day (2+1) → day 3

$i = 3 \Rightarrow$ Per = Evermore[3] + Vanguard[3] + Excelsior[3] = 1+4+3

Per = 8 → For day (3+1) → day 4

i+=1 → 3+1 = 4

$i = 4 \Rightarrow$ Per = Evermore[4] + Vanguard[4] + Excelsior[4] = 1+5+5

i+=1 ⇒ 4+1=5 | Per = 11 → For day (4+1) → day 5

$i = 5 \Rightarrow$ Per = Evermore[5] + Vanguard[5] + Excelsior[5] = 1+6+8

i+=1 ⇒ 5+1=6 | Per = 15 → For day (5+1) → day 6.

$i = 6 \Rightarrow$ Per = Evermore[6] + Vanguard[6] + Excelsior[6] = 1+7+13

Per = 21 → For day (6+1) → day 7.

i+=1 = 6+1=7

i = 7 < 7 : FALSE. The loop ends the execution.

- ~~Com~~ Printing the final values of 'per' in each day till 7 days.
- Comparing the sum of infections in Evermore, Vanguard_city & Excelsior & printing the highest infected population.
- Evermore_sum = 7 ; Vanguard_sum = 28 ; Excelsior_sum = 33
- If Evermore_sum is greater than Vanguard_sum & Excelsior_sum, then, Evermore has highest infected population.
- Else if Vanguard_sum is greater than Evermore_sum & Excelsior_sum, then, Vanguard_city has highest infected population.
- Else if both the above conditions are FALSE, Excelsior has highest infected population.

```
_____________________________________________

Total Numbers Of Infections

_____________________________________________

Total number of infections in Evermore is :  7
Total number of infections in Vanguard_City is :  28
Total number of infections in Excelsior is :  33

_____________________________________________

Infections per day at each location

_____________________________________________

Infections per day at each location : Day 1 : 3 Cases
Infections per day at each location : Day 2 : 4 Cases
Infections per day at each location : Day 3 : 6 Cases
Infections per day at each location : Day 4 : 8 Cases
Infections per day at each location : Day 5 : 11 Cases
Infections per day at each location : Day 6 : 15 Cases
Infections per day at each location : Day 7 : 21 Cases

_____________________________________________

Highest Infected population

_____________________________________________

Location Excelsior has highest infected population of :  33  in these 7 days
```