ENDG 233 – Programming with Data

Functions

UNIVERSITY OF CALGARY

Date: Oct. 18th – 24th

Check in: Term test

- Will be held online (D2L) from 9:30 am to 10:45 am on October 26th.
- No late submission will be accepted.
- It is worth 10% of your overall grade
- computer and a strong internet connection for submission
- The content will include everything up to and including Functions (Week 6 videos/Week 7 active learning)

Check in: Term test

- If you are in a different timezone or have a similar conflict, you may request to write at an alternate start time by emailing **endg233@ucalgary**.ca with your name, block number, ID number, and requested start time (option #1 or #2) **no later than Friday, October 22nd at 11:59 pm.**
  - Alternate start time #1: October 25, 4:30 pm MT
  - Alternate start time #2: October 25, 9:30 pm MT

- The test will consist of two parts:
  a) Multiple choice/multi-select/T-F/blanks, etc. (same as the video checks)
  b) A written code exercise that should be written/tested in VS Code and submitted via the D2L dropbox

Schedule for Week 7

- Review
- Examples on functions
- Example on term test
- portfolio project #2
- In-Lab exercise (graded)

Review: Function

UNIVERSITY OF CALGARY

- **Function** – is a block of code which only runs when it is called.
- **Parameter/argument** – information that passed into function.
  - Parameter is the variable inside the parenthesis in function
  - An argument is the value that is sent to the function
- Python has two types of functions:
    - Built-in functions
    - User-defined functions

Review: Function

- **Creating a function:** in python a function is defined using **def** for example:

```
def myfunc():                # defining function
    print ('hello world')
```

- **Calling a function:** in python use function name with parenthesis

```
def myfunc():                # defining function
    print ('hello world')

myfunc()
```

Review: Function - arguments

- **arguments:** in python information can be passed into function as arguments for example:

```
def myfunc(message ):        # defining function
        print (message)

myfunc("welcome to ENDG 233")
```

- **Passing argument as a list**

```
def print_food_list(food):        # defining function
    for x in food:
        print(x)

fruits = ["apple", "banana", "cherry"]       # fruit list
veggies = [""lettuce", "cucumber", "spinach", "pepper"].    # veggies list

print_food_list(fruits)
print_food_list(veggies)
```

Review: Function

UNIVERSITY OF CALGARY

local variables: Such variables defined inside a function are called local variables.

global variable: A variable defined outside of a function is called a global variable.

Review: Function

- If the object is **immutable**, such as a string or integer, then the modification is limited to inside the function.
- If the object is **mutable**, then in-place modification of the object can be seen outside the scope of the function.

Review: Function

Example 1:

```
1   def modify(num_list):
2       num_list[1] = 99
3
4   my_list = [10, 20, 30]
5   modify(my_list)
6   print(my_list)  # my_list still contains 99!
7
```

```
[10, 99, 30]
```

Example 2:

```
def modify(num_list):
    num_list[1] = 99
    print(num_list)

my_list = [10, 20, 30]
modify(my_list[:])
print(my_list)
```

```
[10, 99, 30]
[10, 20, 30]
```

Tutorial 7.1 : Exact change

UNIVERSITY OF CALGARY

- **Task-** Define a function called **exact_change** that **takes** the **total change amount** in cents and calculates the change using the fewest coins. The coin types are pennies, nickels, dimes, and quarters.
- Then write a main program that reads the total change amount as **an integer input**, **calls exact_change(),** and outputs the change, one coin type per line.
- Use singular and plural coin names as appropriate, like 1 penny vs. 2 pennies. Output "no change" if the input is 0 or less. (quarter = 25 cents, dime = 10 cents, nickel = 5 cents, penny = 1 cent)

Tutorial 7.1 : Exact change

UNIVERSITY OF CALGARY

- Ex1: If the input is:

  0

- The output is:

  No change

- Ex2: if the input is:

  45

- The output is:

  2 dimes

  1 quarter

Note:
Your program must define and call the following function. The function exact_change() should return a list with **four values:** the number of pennies, the number of nickels, the number of dimes, and the number of quarters.
def exact_change(**user_total**)

Tutorial 7.1 : Exact change

UNIVERSITY OF CALGARY

- **Definition of function:**

```
def exact_change(user_total):
    money_total = user_total
    num_quarters = money_total // 25    #get the quarters
    money_total -= num_quarters * 25    #subtract quarters from the total

    num_dimes = money_total // 10       #get the dimes
    money_total -= num_dimes * 10       #subtract dimes from the total


    num_nickels = money_total // 5      #get the nickels
    money_total -= num_nickels * 5      #subtract nickels from the total


    num_pennies = money_total
    num_coins = [num_pennies, num_nickels, num_dimes, num_quarters]     #list of coins
    return num_coins
```

Tutorial 7.1 : Exact change

UNIVERSITY OF CALGARY

```
input_val = int(input())
num_coins = exact_change(input_val)


if input_val <= 0:
        print("no change")
#print the value of pennies
if num_coins[0] > 0:
        print (str(num_coins[0]), end=' ‘)
        if num_coins[0] == 1:
                print("penny")
        else:
                print("pennies")
#print the value of nickles
if num_coins[1] > 0:
        print (str(num_coins[1]), end=' ‘)
        if num_coins[1] == 1:
                print ("nickel")
        else:
                print ("nickels")
```

```
#print the value of dimes
if num_coins[2] > 0:
        print (str(num_coins[2]), end=' ‘)
        if num_coins[2] == 1:
                print ("dime")
        else:
                print ("dimes")
#print the value of quarters
if num_coins[3] > 0:
        print (str(num_coins[3]), end = ' ‘)
        if num_coins[3] == 1:
                print ("quarter")
        else:
                print ("quarters")
```

Tutorial 7.2 : Fibonacci Sequence

UNIVERSITY OF CALGARY

- **Task** – The Fibonacci sequence begins with the numbers 0 and 1. All subsequent values are the sum of the previous two, ex: 0, 1, 1, 2, 3, 5, 8, 13.
- Complete the **fibonacci()** function, which has an index n as a parameter and returns the nth value in the sequence. Any negative index values should return -1.

Tutorial 7.2 : Fibonacci Sequence

UNIVERSITY OF CALGARY

- **Task** – The Fibonacci sequence begins with the numbers 0 and 1. All subsequent values are the sum of the previous two, ex: 0, 1, 1, 2, 3, 5, 8, 13.
- Complete the **fibonacci()** function, which has an index n as a parameter and returns the nth value in the sequence. Any negative index values should return -1.

Ex: if the **input** is:

7

The **output** is:

Fibonacci (7) is 13

Tutorial 7.2 : Fibonacci Sequence

UNIVERSITY OF CALGARY

```
def fibonacci(n):       # function definition with parameter n

        if n < 0:
            return -1
        if n == 0:
            return 0
        if n == 1:
            return 1

        last = 1
        before_last = 0
        for i in range(n):
                fib = last + before_last        # add last and before last to fib
                before_last = last              # assign last value to before last
                    last = fib                  # assign fib to last

            return before_last


start_num = int(input())                        # input the start
print(f'fibonacci({start_num}) is {fibonacci(start_num)}')      #funclion calling
```

Review 8.1 : Rock Paper Scissors

UNIVERSITY OF CALGARY

- **Task -** Write a program to play an automated game of Rock, Paper, Scissors. Two players make one of three hand signals at the same time. Hand signals represent a rock, a piece of paper, or a pair of scissors. Each combination results in a win for one of the players. Rock crushes scissors, paper covers rock, and scissors cut paper. A tie occurs if both players make the same signal. Use a random number generator of 0, 1, or 2 to represent the three signals.

- **Note**: this program is designed for *incremental development*. Complete each step and submit before starting the next step. Only a portion of tests pass after each step but confirm progress.

Review 8.1 : Rock Paper Scissors

- **Step 0.** Read starter template and do not change the provided code. Variables are defined for ROCK, PAPER, and SCISSORS. A seed is read from input to initialize the random number generator. This supports automated testing and creates predictable results that would otherwise be random. This step forms your 10 lines in your program. Please do not change the first 10 line in your program. signals.

```
import random

ROCK = 0
PAPER = 1
SCISSORS = 2

# Read random seed to support testing (do not alter) and starting credits
seed = int(input())
# Set the seed for random
random.seed(int(seed))
```

Review 8.1 : Rock Paper Scissors

- **Step 1.** Read two player names from input (str). Read a number of rounds from the input. If the round value is less than 1, provide an error message (Rounds must be > 0) and read a new number until the user enter a number equal or greater than 1. Output player names and number of rounds.

```
player1_name = input()              # input player1
player2_name = input()               # input player2
rounds = int(input())               # number of rounds

while rounds <= 0:                      # rounds must be greater than 0
    print("Rounds must be > 0")
    rounds = int(input())

print(player1_name, "vs", player2_name, "for", rounds, "rounds")
#initialize the payer value and player win
player1_value = 0
player2_value = 0
player1_wins = 0
player2_wins = 0
```

Review 8.1 : Rock Paper Scissors

- **Step 2 .** Use random.randint(0, 2) to generate random values (0 - 2) for player 1 and player 2. Continue to generate random values for both players until both values do not match. Output "Tie" when the values are same for both player.

```
for n in range(rounds):
    player1_value = random.randint(0, 2)
    player2_value = random.randint(0, 2)
    while player1_value == player2_value:
        print("Tie")
        player1_value = random.randint(0, 2)
        player2_value = random.randint(0, 2)
```

Review 8.1 : Rock Paper Scissors

- **Step 3 .** Identify the winner and output a message. The message will be "Tie" if two players have the same value and show the winner and reason for the win. For instance, "Bert wins with rock". These are the rules that you should consider in your program and message: Rock crushes scissors, scissors cut paper, and paper covers rock.

```
# Step #3
    # Did player 1 win?
    if player1_value == ROCK and player2_value == SCISSORS:
        print(player1_name, "wins with rock")
        player1_wins += 1
    elif player1_value == PAPER and player2_value == ROCK:
        print(player1_name, "wins with paper")
        player1_wins += 1
    elif player1_value == SCISSORS and player2_value == PAPER:
        print(player1_name, "wins with scissors")
        player1_wins += 1
```

Review 8.1 : Rock Paper Scissors

- **Step 3 .** Identify the winner and output a message. The message will be "Tie" if two players have the same value and show the winner and reason for the win. For instance, "Bert wins with rock". These are the rules that you should consider in your program and message: Rock crushes scissors, scissors cut paper, and paper covers rock.

```
# Did player 2 win?
    if player2_value == ROCK and player1_value == SCISSORS:
        print(player2_name, "wins with rock")
        player2_wins += 1
    elif player2_value == PAPER and player1_value == ROCK:
        print(player2_name, "wins with paper")
        player2_wins += 1
    elif player2_value == SCISSORS and player1_value == PAPER:
        print(player2_name, "wins with scissors")
        player2_wins += 1
```

Review 8.1 : Rock Paper Scissors

- **Step 4.** Add a loop to repeat steps 2 and 3 for the number of rounds. Output total wins for each player after all rounds are complete.

```
print(f"{player1_name} wins {player1_wins} and {player2_name} wins {player2_wins}")
```

Tutorial 5.4 : Car Wash Costs

UNIVERSITY OF CALGARY

- **Solution:**

```
#defining a dictionary. Keys:values
services = { 'Air freshener' : 1 ,
             'Rain repellent': 2,
             'Tire shine' : 2,
             'Wax' : 3,
             'Vacuum' : 5 }
base_wash = 10
total = 0

service_choice1 = input()       #input service choice 1
service_choice2 = input()       #input service choice 2
total += base_wash

print('ZyCar Wash')
print('Base car wash -- $10')
```

Tutorial 5.4 : Car Wash Costs

- **Solution:**

```
if service_choice1 in services.keys():
total += services[service_choice1]
print(service_choice1, f'-- ${services[service_choice1]}' )

if service_choice2 in services.keys():
total += services[service_choice2]
print(service_choice2, f'-- ${services[service_choice2]}')


print(f'----\nTotal price: ${total}' )
```

Tutorial 7.3: Multiples of ten in a list

UNIVERSITY OF CALGARY

- **Task** – Write a program that **reads a list of integers**, and outputs whether the **list contains** all multiples of 10, no multiples of 10, or mixed values. Define a function named **is_list_mult10** that takes a list as a parameter, and returns a boolean that represents whether the list contains all multiples of ten.
- Define a second function named **is_list_no_mult10** that takes a list as a parameter and returns a boolean that represents whether the list contains no multiples of ten.
- The program should first take an integer, representing the size of the list, then take the list values. The first integer is not included in the list.

Tutorial 7.3: Multiples of ten in a list

UNIVERSITY OF CALGARY

- Ex: if the input is:

  5
  20
  40
  60
  80
  100

- The output is:

  all multiples of 10

Portfolio Assignment 2

- Due date: Oct 29th,2021 @ 11:59pm

Portfolio Assignment 2: Tips

- Import string
- string.ascii_lowercase
- “”.join()

```
import string
alphobet = string.ascii_lowercase
print(alphobet)
alphobet_list=list(alphobet)
print()
print(alphobet_list)
print()
print("".join(alphobet_list))
```

```
abcdefghijklmnopqrstuvwxyz

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

abcdefghijklmnopqrstuvwxyz
```

Portfolio Assignment 2: Tips

- **Zip():** returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together etc
- **Dict():** creates a dictionary

```
a=['a','d','z','y','h']
b=['*','@','d','W','U']
pair_ab=zip(a,b)
dict_pair_ab=dict(pair_ab)
print(dict_pair_ab)
```

```
{'a': '*', 'd': '@', 'z': 'd', 'y': 'W', 'h': 'U'}
```