ENDG 233 – Programming with Data

Advance strings, list and Dictionaries

Date: Oct. 25th – 31st

Schedule for Week 8

UNIVERSITY OF CALGARY

- Examples on advance strings, lists, dictionaries
- Work on portfolio project #2
- Tuesday:
  - Term Test #1 at 9:30am
- Wednesday:
  - zyLabs graded exercise based on last week's material
- Friday:
  - Portfolio Project #2 due on Oct 29th at 11:59pm

Check in: Term test

- Will be held online (D2L) from 9:30 am to 10:45 am on Tuesday, 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

- 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

Review: Strings and String methods

UNIVERSITY OF CALGARY

- **Strings** – are surrounded by single/double quotation marks in python
- **String Methods** – There are set of built-in methods that can be use on Strings in python

Review: Useful String methods

UNIVERSITY OF CALGARY

Method()	Description
isalnum()	Returns True if all characters in string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabe
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
join()	Converts the elements of an iterable into a string
lower()	Converts a string into lower case
replace()	Returns a string where a specified value is replaced with a specified value
split()	Splits the string at the specified separator, and returns a list
count()	Returns the number of times a specified value occurs in a string
find()	Searches the string for a specified value and returns the position of where it was found
title()	Converts the first character of each word to upper case
upper()	Converts a string into upper case

Review: String Slicing

UNIVERSITY OF CALGARY

- To access a portion of a string instead of a single index, you can use slice notation:
- **my_str[start:stop]**
- Note that the character at the stop index is not included
- If my_str is 'Boggle', then **my_str[0:3]** yields string 'Bog'
- Other sequence types like lists and tuples also support slice notation
- You can leave the start/stop empty if you want to start at the beginning or go to the end

Review: String Slicing Cont'd

UNIVERSITY OF CALGARY

- Negative numbers can be used to specify an index relative to the end of the string
- Ex: If the variable my_str is 'Jane Doe!?', then:
- **my_str[0:-2]** or **my_str[:-2]**
- yields 'Jane Doe' because the -2 refers to the second-to-last character '!' (and the character at the end index is not included in the result string)

Review: String Slicing Cont'd

UNIVERSITY OF CALGARY

- The slicing notation can take a third variable: stride
- The stride value is the increment between indexes

str[start:stop:stride]

- For example, **my_str[0:10:2]** reads every other element between 0 and 10
- The stride defaults to 1 if not specified

Review: Substring Replacement

UNIVERSITY OF CALGARY

- Recall that string objects are immutable: once created, strings can not be changed
- The replace string method provides a simple way to create a new string by replacing all occurrences of a substring with a new substring
- **replace(old, new)**: Returns a copy of the string with all occurrences of the substring **old** replaced by the string **new**. The old and new arguments may be string variables or string literals
- **replace(old, new, count)**: Same as above, except only replaces the first **count** occurrences of **old**

Review: Substring find

UNIVERSITY OF CALGARY

- Some methods are useful for finding the position of where a character or substring is located in a string:
- **find(x)**: Returns the index of the first occurrence of item x in the string, else returns -1. x may be a string variable or string literal
- **find(x, start, end)**: Same as **find(x)** but starts at the **start** index and ends at the **end-1** index
- **rfind(x)**: Same as **find(x)** but searches the string in reverse, returning the last occurrence in the string

Review: Comparing Strings

UNIVERSITY OF CALGARY

- String objects may be compared using relational operators (<, <=, >, >=), equality operators (==, !=), membership operators (in, not in), and identity operators (is, is not)
- Evaluation of relational and equality operator comparisons occurs by first comparing the corresponding characters at element 0, then at element 1, etc., stopping as soon as a determination can be made
- For a relational comparison (<, >, etc.), the result will be the result of comparing the ASCII/Unicode values of the first differing character pair

Review: Splitting and joining string

- To break a larger string into smaller substrings:

my_list = my_str.split(separator)

- Divides my_str into a list of strings separated by the desired separator. The default separator is ‘ ‘
  - The separator is not included in the list
- To convert the list back into a string, use join()

my_str = separator.join(my_list)

Review: Splitting and joining string

- To break a larger string into smaller substrings:

  my_list = my_str.split(separator)

- Divides my_str into a list of strings separated by the desired separator. The default separator is ‘ ‘
  - The separator is not included in the list
- To convert the list back into a string, use join()

  my_str = separator.join(my_list)

Tutorial 9.1: Login Name

UNIVERSITY OF CALGARY

- **Task**: Write a program that creates a login name for a user, given the user's first name, last name, and a four-digit integer as input. Output the login name, which is made up of the first five letters of the last name, followed by the first letter of the first name, and then the last two digits of the number (use the % operator). If the last name has less than five letters, then use all letters of the last name.

- **Input :** Michael Jordan 1991
- **Output:** JordaM91

Tutorial 9.1: Login Name

UNIVERSITY OF CALGARY

Pseudocode

```
# split the string input values to list
# assign first value to first_name
# assign sec value to last_name
#assign integer value to year

#condition check if last name >5
# get first five letters
#print the login name
```

Tutorial 9.1: Login Name

UNIVERSITY OF CALGARY

```
values = input().split()   # split the string input values to list
first_name = values[0]  # assign first value to first_name
last_name = values[1]   # assign first value to last_name
year = int(values[2])      #assign integer value to year

if len(last_name) > 5:     #condition check
    last_name = last_name[:5]   # get first five letters
#print the login name
print(f"Your login name: {last_name}{first_name}{year % 100}")
```

Tutorial 9.2: Palindrome

UNIVERSITY OF CALGARY

- A palindrome is a word or a phrase that is the same when read both forward and backward. Examples are: "bob," "sees," or "never odd or even" (ignoring spaces).
- **Task** - Write a program whose input is a word or phrase, and that outputs whether the input is a palindrome.

- **Input :** bob
- **Output:** bob is a palindrome

Tutorial 9.2: Palindrome

UNIVERSITY OF CALGARY

Pseudocode

```
# input the string
# remove the spaces between phrases.
# reverse the string
# check if the reversed string and input string are equal
# print the message
```

Tutorial 9.2: Palindrome

UNIVERSITY OF CALGARY

```
user_input = input()          #input the string

normal_str = user_input.replace(" ", "")     # remove the space between phrase
reverse_str = normal_str[::-1]               # reverse the string

if normal_str == reverse_str: # check if the reversed string and input string are equal
    print(f'{user_input} is a palindrome') # print the message
else:
    print(f'{user_input} is not a palindrome')
```

Tutorial 9.3: Text Analyzer & Modifier

UNIVERSITY OF CALGARY

- Prompt the user to enter a string of their choosing. Output the string
- Complete the get_num_of_characters() function, which returns the number of characters in the user's string. We encourage you to use a for loop in this function
- Complete the output_without_whitespace() function, which outputs the string's characters except for whitespace (spaces, tabs) (**HINT**: use isspace() function)
- Call both functions and then output the returned results

Tutorial 9.3: Text Analyzer & Modifier

UNIVERSITY OF CALGARY

If the Input is -

The only thing we have to fear is fear itself.

The Output is –

Enter a sentence or phrase:

You entered: The only thing we have to fear is fear itself.

Number of characters: 46

String with no whitespace: Theonlythingwehavetofearisfearitself.

Tutorial 9.3: Text Analyzer & Modifier

UNIVERSITY OF CALGARY

```
# functions to get the number of characters
#def get_num_of_characters(input_str):    # function definition
        #initialize number of char to zero
        # loop through string
                # increment the number of characters
        return num_chars


# function to print without spaces
#def output_without_whitespace(input_str):    #function definition
            #loop through the string
                    # if the string[index] is not a space
                        print the string
```

Tutorial 9.3: Text Analyzer & Modifier

UNIVERSITY OF CALGARY

```
# functions to get the number of characters
def get_num_of_characters(input_str):
        num_chars = 0;
        for i in input_str:
                num_chars += 1
        return num_chars


# function to print without spaces
def output_without_whitespace(input_str):
        print('String with no whitespace:', end = " ")
        for i in range(len(input_str)):
                # i != ' ' or i !='\t'
                if not input_str[i].isspace():
                        print(input_str[i], end = "")
        print('\n', end = "")
```

Tutorial 9.3: Text Analyzer & Modifier

UNIVERSITY OF CALGARY

```
# main function

    user_str = input('Enter a sentence or phrase:\n')

    print('\nYou entered:', user_str)

    num_chars =  get_num_of_characters(user_str)
    print('\nNumber of characters:', num_chars)

    output_without_whitespace(user_str)
```

Review: List methods

UNIVERSITY OF CALGARY

- list.append(x): Add an item to the end of list
- list.extend([x]): Add all items in [x] to list
- list.insert(i, x): Insert x into list before position I
- list.remove(x): Remove first item from list with value x
- list.pop(i): Remove and return item at position i in list
- list.sort() : Sort the items of list in-place
- list.reverse(): Reverse the elements of list in-place

Review: List built-in function

UNIVERSITY OF CALGARY

- all(list): True if every element in list is True (!= 0), or if the list is empty
- any(list): True if any element in the list is True
- max(list): Get the maximum element in the list
- min(list): Get the minimum element in the list
- sum(list): Get the sum of all elements in the list

Review: main function

- Convert the main body of code into a function using:

if __name__ == '__main__’:

The main body of code is indented below this line of code
The other functions are defined above this line and the code within the main function calls the other functions
zyLabs tests function output, so make sure you read the function specifications carefully

Tutorial 9.4: Convert to Binary

- **Task1 –** Write a program that takes in a positive integer as input, and outputs a string of 1's and 0's representing the integer in binary.
- The program must define and call the following two functions. Define a function named int_to_reverse_binary() that takes an integer as a parameter and returns a string of 1's and 0's representing the integer in binary (in reverse). Define a function named string_reverse() that takes an input string as a parameter and returns a string representing the input string in reverse.
- Note: You will need to write a second function to reverse the string.

2	35	1
2	17	1
2	8	0
2	4	0
2	2	0
	1	

$(1\,0\,0\,0\,1\,1)_2$

Tutorial 9.4: Convert to Binary

UNIVERSITY OF CALGARY

```
# integer to reverse binary function
def int_to_reverse_binary(num1):
    binary_val = ‘ '

    while num1 > 0:
        if (num1 % 2) == 0:
            binary_val += '0'
        else:
            binary_val += '1'
        num1 = num1 // 2

    return binary_val;
```

```
# reverse the string
def string_reverse(input_string):
    reverse_input = ''

    for i in range(len(input_string)-1, -1, -1):
        reverse_input += input_string[i]

    return reverse_input
```

Tutorial 9.4: Convert to Binary

UNIVERSITY OF CALGARY

```
#main function
if __name__ == '__main__':

    user_input = int(input())      # user input

    binary_string = str(int_to_reverse_binary(user_input))
    binary_string = str(string_reverse(binary_string))

    print(binary_string)
```

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'}
```