Computational Thinking and Programming – II

(PYTHON)

FUNCTIONS

Question 1

Write a program to have following functions:

a) A function that takes a number as a parameter and calculates the cube for it. The function does not return a value. If there is no value passed to the function in the function call statement as an argument, the function should calculate and print the cube of 2

Source Code:

```
def calculate_cube(num=2):
    cube = num ** 3
    print("Cube:", cube)
calculate_cube(3)
calculate_cube()
```

Output:

```
Cube: 27
Cube: 8
```

b) A function that takes two strings as arguments and return True if both the strings are equal otherwise returns False

Source Code:

```
def compare_strings(str1, str2):
    if str1 == str2:
        return True
    else:
        return False
print(compare_strings("hello", "hello"))
print(compare_strings("hello", "world"))
```

Output:

```
True
False
```

Question 2

Write a function that has two numeric parameters and generates a random number between those two numbers. Using this function, the main program should be able to print three numbers randomly.

Source Code:

```
import random
def generate_random_number(start, end):
    return random.randint(start, end)
for i in range(3):
    print(generate_random_number(1, 10))
```

Output:

```
6
9
9
```

Question 3

Write a function that receives two string arguments and checks whether they are the same length strings (return True in this case otherwise False). Write Function Call statement and display “Same Length” or “Not Same Length” based on the result received from the function.

Source Code:

```
def check_string_length(str1, str2):
    if len(str1) == len(str2):
        return True
    else:
        return False
result = check_string_length("hello", "world")
if result:
    print("Same Length")
else:
    print("Not Same Length")
```

Output:

```
Same Length
```

Question 4

Write a function that takes a number n and then returns a number that has maximum face value in unit’s place.
Example: if numbers passed are 491 and 278, then the function will return 278.

Source Code:

```
def find_max_units(num1, num2):
    max_num = max(num1 % 10, num2 % 10)
    if num1 % 10 == max_num:
        return num1
    else:
        return num2
print(find_max_units(491, 278))
```

Output:

```
278
```

Question 5

Define a function named LINEARSEARCH (lst,k) that accepts a list lst and key to be searched k as parameters, Implements linear search to find all occurrences of k in lst and returns another list containing all the indexes/positions where k is found in lst. The function should return None if k is not found in lst. Write an appropriate function call statement and print the result accordingly.

Source Code:

```
def linear_search(lst, k):
    indexes = []
    for i in range(len(lst)):
        if lst[i] == k:
            indexes.append(i)
    if len(indexes) == 0:
        return None
    else:
        return indexes
lst = [1, 2, 3, 4, 5, 4, 3, 2, 1]
```

```
indexes = linear_search(lst, 3)
if indexes is not None:
    print("Found at indexes:", indexes)
else:
    print("Not found")
```

Output:

```
Found at indexes: [2, 6]
```

Question 6

Define a function that accepts a list of numbers as a parameter, find and print the sum and the average of all elements. Write an appropriate function call statement and print the result accordingly.

Source Code:

```
def calculate_sum_and_average(numbers):
    total = sum(numbers)
    average = total / len(numbers)
    return total, average
numbers = [1, 2, 3, 4, 5]
total, average = calculate_sum_and_average(numbers)
print("Sum:", total)
print("Average:", average)
```

Output:

```
Sum: 15
Average: 3.0
```

Question 7

Define a function that takes two lists as parameters and create a third list that contains the elements from both the lists and sort it in ascending order. Write an appropriate function call statement and print the result accordingly.

Source Code:

```
def merge_and_sort_lists(lst1, lst2):
    merged_list = lst1 + lst2
    merged_list.sort()
    return merged_list
list1 = [1, 3, 5, 7, 9]
list2 = [2, 4, 6, 8, 10]
merged_and_sorted = merge_and_sort_lists(list1,
list2)
print("Merged and Sorted List:", merged_and_sorted)
```

Output:

```
Merged and Sorted List: [1, 2, 3, 4, 5, 6, 7, 8, 9,
10]
```

Question 8

Write a function part_reverse(<list>,start, end) to reverse elements in a part of the list passed as a parameter to the function. The parameters start and end are the starting and ending index of the part of the list which must be reversed.

Assume that start < end, start>=0 and end<len(list)

Sample Input Data of List

my_list=[1,2,3,4,5,6,7,8,9,10]

Function Call = part_reverse(my_list,3,6)

Output is

my_list=[1,2,3,7,6,5,4,8,9,10]

Source Code:

```
def part_reverse(lst, start, end):
    reversed_part = lst[start:end+1][::-1]
    lst[start:end+1] = reversed_part
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
part_reverse(my_list, 3, 6)
print("Modified List:", my_list)
```

Output:

```
Modified List: [1, 2, 3, 7, 6, 5, 4, 8, 9, 10]
```

Question 9

Define a function named Unit_Seven(*n) that takes variable length parameters. The function should find and return a list of all values of input parameters that are ending with the digit seven. Write appropriate top level statements to call the function and to display the list obtained as result
Sample Function call:
L=Unit_Seven(127,200,400,207,17,-127)
After the execution L will have [127,207,17,-127]
Source Code:

```
def Unit_Seven(*n):
    values_ending_with_seven = []
    for value in n:
        if str(value)[-1] == '7':
            values_ending_with_seven.append(value)
    return values_ending_with_seven
L = Unit_Seven(127, 200, 400, 207, 17, -127)
print("Values ending with seven:", L)
```

Output:

```
Values ending with seven: [127, 207, 17, -127]
```

Question 10

Write a user defined function to check the validity of a password string passed as a parameter. The function returns True if all the Validation Rules given below are satisfied otherwise it returns False.
Validation Rules:
Given password must have
At least 1 letter between [a-z] and 1 letter between [A-Z].
At least 1 number between [0-9].
At least 1 character from [$#@].
Minimum length of 5 characters and Maximum length of 15 characters.
Write appropriate top level statements to accept a password from user, check and print whether the password

is Valid or Invalid by using the value returned by the above user defined function.

Source Code:

```
def validate_password(password):
    if len(password) < 5 or len(password) > 15:
        return False
    if not any(char.islower() for char in password):
        return False
    if not any(char.isupper() for char in password):
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char in "$#@!" for char in password):
        return False
    return True
password = input("Enter a password: ")
if validate_password(password):
    print("Valid password")
else:
    print("Invalid password")
```

Output:

```
Enter a password: Hello@123
Valid password
```