1. Manipulate using a list.

i) To add new elements to the end of the list ii) To reverse elements in the list iii) To display the same list of elements multiple. iv) To concatenate two list v) To sort the elements in the list in ascending order.

```
# i)
list=[11,22,33]
list.append(44)
print('list =',list)
```

```
     list = [11, 22, 33, 44]
```

```
# ii)
list=[1,2,0]
list.reverse()
print('reversed list',list)
```

```
     reversed list [0, 2, 1]
```

```
# iii)
list=[1,2,11,22,44]
print(list)
```

```
     [1, 2, 11, 22, 44]
```

```
# iv)
list1 =[1,2,3]
list2 =[4,5,6]
result = list1+list2
print('list =',result)
```

```
     list = [1, 2, 3, 4, 5, 6]
```

```
# v)
list = [3,1,2,400,100,44,1000,144]
list.sort()
print('sorted list',list)
```

```
     sorted list [1, 2, 3, 44, 100, 144, 400, 1000]
```

2. Write a python program to do in the tuples.

i) Manupulating using tuples. ii) To add new elements to the end of the tuples iii) To reverse elements in the list iv) To display the elements of the same tuple multiple times. v) To concatenate two tuples vi) To sort the elements in the list in ascending order.

```
# i)
tuple =(1,44,80,300,67)
print('i)',tuple[2])

# ii)
new = 1000
tuple = tuple+(new,)
print('ii)new element added',tuple)

# iii)
reverse = tuple[::-1]
print('iii) reversed elements',reverse)

# iv)
repeat = 3
multiple = tuple * repeat
print('iv) multiple times',multiple)

# v)
tuple1 = (7,8,9)
concatenated = tuple + tuple1
print('v) concatenate two tuples',concatenated)

# vi)
sort =sorted(tuple)
print('vi) sorted elements',sort)
```

```
i) 80
ii)new element added (1, 44, 80, 300, 67, 1000)
iii) reversed elements (1000, 67, 300, 80, 44, 1)
iv) multiple times (1, 44, 80, 300, 67, 1000, 1, 44, 80, 300, 67, 1000, 1, 44, 80, 30
v) concatenate two tuples (1, 44, 80, 300, 67, 1000, 7, 8, 9)
vi) sorted elements [1, 44, 67, 80, 300, 1000]
```

Double-click (or enter) to edit

3. Write a python program to implement the following using list.

i) Create a list with integer(minimum 10 numbers) ii) How to display the last number in the list iii) Command for displaying the values from the list[0:4] iv) Command for displaying the values from the list[2:] v) Command for displaying the values from the list[:6]

```
# i)
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print('i)integer',my_list)

# ii)
last_number = my_list[-1]
print("ii)Last number in the list:", last_number)

# iii)
slice_0_to_4 = my_list[0:4]
print("iii)Values :", slice_0_to_4)

# iv)
slice_from_2 = my_list[2:]
print("iv)Values :", slice_from_2)

# v)
slice_to_6 = my_list[:6]
print("v)Values :", slice_to_6)
```

```
i)integer [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ii)Last number in the list: 10
iii)Values : [1, 2, 3, 4]
iv)Values : [3, 4, 5, 6, 7, 8, 9, 10]
v)Values : [1, 2, 3, 4, 5, 6]
```

Writre a python program: tuple1 =(10,50,20,40,30)

i. To display the elements 10 and 50 From Tuple1. ii. To display the length of a tuple1. iii.To find the minimum element from tuple1. iv. To add all elements in the tuple1. v. To display the same tuple1 multiple times.

```
tuple1 = (10, 50, 20, 40, 30)

# i.
```

```
print("Elements :", 10,50)

# ii.
print("Length of Tuple1:", len(tuple1))

# iii.
min_element = min(tuple1)
print("Minimum element :", min_element)

# iv.
add_all = sum(tuple1)
print("add all elements :", add_all)

# v.
n = 3
tuple1_multiplied = tuple1 * n
print("multiplie times :", tuple1_multiplied)
```

```
Elements : 10 50
Length of Tuple1: 5
Minimum element : 10
add all elements : 150
multiplie times : (10, 50, 20, 40, 30, 10, 50, 20, 40, 30, 10, 50, 20, 40, 30)
```

5. Write a python program:

i. To calclate the length of a string ii. To reverse words in a string iii. To display the same string multiple times iv. To concatenate two strings v. str1="south India", using string slicing to display "India"

```
str1 = "south India"

# i.
print("i. length of the string :", len(str1))

# ii.
print("ii. reversed string :", str1[::-1])

# iii.
print("iii. string repeated :",str1 * 3)

# iv.
str2 = "is good"
print("iv. concatenated string :", str1 + str2)

# v.
print("v. :", str1[6:])
```

```
i. length of the string : 11
ii. reversed string : aidnI htuos
iii. string repeated : south Indiasouth Indiasouth India
iv. concatenated string : south Indiais good
v. : India
```

6. perform the program:

i) Creating the Dictionary. ii) Accessing values and keys in the Dictionary. iii) Updating the dictionary using a function. iv) Clear and delete the dictionary values.

```
# i)
dict = {'name': 'Ram', 'age': 30, 'city': 'Chennai'}

# ii)
print(dict['name'])
print(dict['age'])
print(dict.keys())
print(dict.values())

# iii)
def update_dict(dict, key, value):
    dict[key] = value

update_dict(dict, 'age', 31)
print(dict['age'])

# iv) clear the dictionary
dict.clear()
print(dict)

# iv) Delete the dictionary
del dict
print(dict)
```

```
Ram
30
dict_keys(['name', 'age', 'city'])
dict_values(['Ram', 30, 'Chennai'])
31
{}
<class 'dict'>
```

7. Python program to insert a number to any position in a list.

```
my_list = [1, 2, 3, 4, 5]


insert = 10
position = 2
my_list.insert(position, insert)

print("Updated List:", my_list)
```

```
Updated List: [1, 2, 10, 3, 4, 5]
```

8. Python program to delete an element from a list by index.

```
list = [1, 2, 3, 4, 5]
index = 2

new_list = delete_element(list, index)

print(new_list)
```

```
[1, 2, 4, 5]
```

9. Write a program to display a number from 1 to 100.

```
for number in range(1, 101):
    print(number)
```

```
1
2
3
4
5
6
7
```

```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
```

Double-click (or enter) to edit

10. Write a python program to find the sum of all items in a tuple.

```
my_tuple = (1, 2, 3, 4, 5)


total = 0
for item in my_tuple:
    total += item

print("Sum of tuple elements:", total)
```

```
 Sum of tuple elements: 15
```

11. Create a dictionary containing three lambda functions square, cube and square root.

i) E.g. dict = {'Square': function for squaring, 'Cube': function for cube, 'Squareroot': function for square root}

ii) Pass the values (input from the user) to the functions in the dictionary respectively.

iii) Then add the outputs of each function and print it.

```
functions = {
    'Square': lambda x: x**2,
    'Cube': lambda x: x**3,
    'Squareroot': lambda x: x**0.5
}


value = float(input("Enter a number: "))


result = 0
```

```
for function_name, func in functions.items():
    output = func(value)
    print(f"{function_name}({value}) = {output}")
    result += output

print("Sum of function outputs:", result)
```

```
Enter a number: 2
Square(2.0) = 4.0
Cube(2.0) = 8.0
Squareroot(2.0) = 1.4142135623730951
Sum of function outputs: 13.414213562373096
```

12. A list of words is given. Find the words from the list that have their second character in uppercase. Is = ['hello', 'Dear', 'how', 'ARe', 'You']

```
word_list = ['hello', 'Dear', 'how', 'ARe', 'You']

result_words = []

for word in word_list:
    if len(word) >= 2 and word[1].isupper():
        result_words.append(word)

print("Words with the second character in uppercase:")
print(result_words)
```

```
Words with the second character in uppercase:
['ARe']
```

13. A dictionary of names and their weights on earth is given. Find how much they will weigh on the moon. (Use map and lambda functions) Formula: wMoon = (wEarth * GMoon) / GEarth

i) # Weight of people in kg

WeightOnEarth ('John':45, "Shelly':65, 'Marry':35)

ii) # Gravitational force on the Moon: 1.622 m/s2

GMoon = 1.622

iii) # Gravitational force on the Earth: 9.81 m/s2

GEarth = 9.81

```
WeightOnEarth = {'John': 45, 'Shelly': 65, 'Marry': 35}
```

```
GMoon = 1.622
GEarth = 9.81
```

```
WeightOnMoon = list(map(lambda w: (w * GMoon) / GEarth, WeightOnEarth.values()))
```

```
WeightOnMoonDict = dict(zip(WeightOnEarth.keys(), WeightOnMoon))
```

```
print("Weight on the Moon:")
for name, weight in WeightOnMoonDict.items():
    print(f"{name}: {weight} kg")
```

```
Weight on the Moon:
John: 7.440366972477065 kg
Shelly: 10.747196738022426 kg
Marry: 5.786952089704383 kg
```

Control Structures:

1.Write a python program to find the first N prime numbers.

```
def is_prime(number):
  if number <= 1:
    return False
```

```
  for i in range(2, number):
    if number % i == 0:
      return False

  return True


def find_primes(n):
  primes = []
  for i in range(2, n + 1):
    if is_prime(i):
      primes.append(i)

  return primes


print(find_primes(10))
```

```
    [2, 3, 5, 7]
```

2. Write the python code that calculates the salary of an employee. Prompt the user to enter the Basic Salary, HRA, TA, and DA. Add these components to calculate the Gross Salary. Also, deduct 10% of salary from the Gross Salary to be paid as tax and display gross minus tax as net salary.

```
basic_salary = float(input("Enter Basic Salary: "))
hra = float(input("Enter HRA: "))
ta = float(input("Enter TA: "))
da = float(input("Enter DA: "))
gross_salary = basic_salary + hra + ta + da


tax = 0.1 * gross_salary


net_salary = gross_salary - tax


print(f"Basic Salary: {basic_salary}")
print(f"HRA: {hra}")
```

```
print(f"TA: {ta}")
print(f"DA: {da}")
print(f"Gross Salary: {gross_salary}")
print(f"Tax (10% of Gross Salary): {tax}")
print(f"Net Salary: {net_salary}")
```

```
Enter Basic Salary: 1234
Enter HRA: 13245
Enter TA: 354423
Enter DA: 214345
Basic Salary: 1234.0
HRA: 13245.0
TA: 354423.0
DA: 214345.0
Gross Salary: 583247.0
Tax (10% of Gross Salary): 58324.700000000004
Net Salary: 524922.3
```

3. Write a python program to search for a string in the given list.

```
my_list = ['animal', 'dog', 'cat', 'cow', 'pig']


search_string = input("Enter a string to search for: ")

found = False

for item in my_list:
    if search_string == item:
        found = True
        break

if found:
    print(f"The string '{search_string}' was found in the list.")
else:
    print(f"The string '{search_string}' was not found in the list.")
```

```
Enter a string to search for: cat
The string 'cat' was found in the list.
```

4. Write a Python function that accepts a string and

calculates the number of upper-case letters and lower-case letters.

```
def count_upper_lower(string):

    upper_count = 0
    lower_count = 0


    for char in string:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1

    return upper_count, lower_count


input_string = input("Enter a string: ")


upper, lower = count_upper_lower(input_string)


print(f"Uppercase letters: {upper}")
print(f"Lowercase letters: {lower}")
```

```
 Enter a string: head
 Uppercase letters: 0
 Lowercase letters: 4
```

5. Write a program to display the sum of odd numbers and even numbers that fall between 12 and 37.

```
sum_odd = 0
sum_even = 0


for num in range(12, 38):
    if num % 2 == 0:
        sum_even += num
    else:
            ..
```

```
        sum_odd += num

print(f"Sum of even numbers between 12 and 37: {sum_even}")
print(f"Sum of odd numbers between 12 and 37: {sum_odd}")
```

```
Sum of even numbers between 12 and 37: 312
Sum of odd numbers between 12 and 37: 325
```

6. Write a python Program to print the table of any number.

```
num = int(input("Enter a number: "))

start = 1
end = 10

print(f"Multiplication Table for {num}:")
for i in range(start, end + 1):
    result = num * i
    print(f"{num} x {i} = {result}")
```

```
Enter a number: 4
Multiplication Table for 4:
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

7. Write a Python program to sum the first 10 prime numbers.

```
def is_prime(num):
    if num <= 1:
        return False
    elif num <= 3:
        return True
    elif num % 2 == 0 or num % 3 == 0:
        return False
```

```
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

count = 0
sum_primes = 0
number = 2

while count < 10:
    if is_prime(number):
        sum_primes += number
        count += 1
    number += 1

print("Sum of the first 10 prime numbers:", sum_primes)
```

```
 Sum of the first 10 prime numbers: 129
```

8. Write a python program to implement arithmetic operations using nested if statement.

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
operation = input("Enter the operation (+, -, *, /): ")

if operation == '+':
    result = num1 + num2
elif operation == '-':
    result = num1 - num2
elif operation == '*':
    result = num1 * num2
elif operation == '/':
    if num2 != 0:
        result = num1 / num2
    else:
        result = "Division by zero is not allowed."
else:
    result = "Invalid operation."

print(f"Result: {result}")
```

```
Enter the first number: 4
Enter the second number: 4
Enter the operation (+, -, *, /): /
Result: 1.0
```

9. Write a python program to take the temperature in Celsius and convert it to a Fahrenheit.

```
celsius = float(input("Enter temperature in Celsius: "))

fahrenheit = (celsius * 9/5) + 32

print(f"celsius to Fehrenheit.",fahrenheit)
```

```
Enter temperature in Celsius: 96
celsius to Fehrenheit. 204.8
```

10. Write a python program to find a maximum and minimum number in a list without using an inbuilt function.

```
def find_max_min(list1):
  max_number = list1[0]
  min_number = list1[0]

  for number in list1:
    if number > max_number:
      max_number = number
    elif number < min_number:
      min_number = number

  return max_number, min_number


list1 = [10, 4, 2, 9, 7]

max_number, min_number = find_max_min(list1)

print("The maximum number is:", max_number)
print("The minimum number is:", min_number)
```

```
The maximum number is: 10
The minimum number is: 2
```

11. Write a program in python to print out the number of seconds in 30-day month 30 days, 24 hours in a day, 60 minutes per day, 60 seconds in a minute.

```
def number_of_seconds_in_month():
  number_of_seconds_in_day = 24 * 60 * 60
  number_of_seconds_in_month = number_of_seconds_in_day * 30

  return number_of_seconds_in_month


print("The number of seconds in a 30-day month is:", number_of_seconds_in_month())
```

```
The number of seconds in a 30-day month is: 2592000
```

12. Write a program in python to print out the number of seconds in a year.

```
days_per_year = 365
hours_per_day = 24
minutes_per_hour = 60
seconds_per_minute = 60

seconds_in_a_year = days_per_year * hours_per_day * minutes_per_hour * seconds_per_minute

print(f"Number of seconds in a year: {seconds_in_a_year} seconds")
```

```
Number of seconds in a year: 31536000 seconds
```

13. A high-speed train can travel at an average speed of 150 mph, how long will it take a train travelling at this speed to travel from London to Glasgow which is 414 miles away?

```
speed_mph = 150
distance_miles = 414

time_hours = distance_miles / speed_mph

time_hours_integer = int(time_hours)
time_minutes = (time_hours - time_hours_integer) * 60

print(f"It will take {time_hours_integer} hours and {time_minutes:.2f}.")
```

```
It will take 2 hours and 45.60.
```

14. Write a python program that defines a variable called days_in_each_school_year and assign 192 to the variable. The program should then print out the total hours that you spend in school from year 7 to year 11, if each day you spend 6 hours in school days_in_each_school_year = 192

```
days_in_each_school_year = 192

total_hours_spent_in_school = 6 * days_in_each_school_year * 5

print(total_hours_spent_in_school)
```

```
5760
```

15. If the age of Ram,Sam and Khan are input through the keyboard, write a python program to determine the eldest and youngest of the three.

```
ram_age = int(input("Enter the age of Ram: "))
```

```
sam_age = int(input("Enter the age of Sam: "))
khan_age = int(input("Enter the age of Khan: "))


if ram_age >= sam_age and ram_age >= khan_age:
    eldest = "Ram"
elif sam_age >= ram_age and sam_age >= khan_age:
    eldest = "Sam"
else:
    eldest = "Khan"

if ram_age <= sam_age and ram_age <= khan_age:
    youngest = "Ram"
elif sam_age <= ram_age and sam_age <= khan_age:
    youngest = "Sam"
else:
    youngest = "Khan"


print(f"The eldest among Ram, Sam, and Khan is: {eldest}")
print(f"The youngest among Ram, Sam, and Khan is: {youngest}")
```

```
Enter the age of Ram: 52
Enter the age of Sam: 42
Enter the age of Khan: 80
The eldest among Ram, Sam, and Khan is: Khan
The youngest among Ram, Sam, and Khan is: Sam
```

16. Write a python program to rotate a list by right n times with and without slicing technique.

```
def rotate_right_with_slicing(lst, n):
    n = n % len(lst)
    rotated_list = lst[-n:] + lst[:-n]
    return rotated_list

original_list = [1, 2, 3, 4, 5]


n = int(input("Enter the number of times to rotate right: "))

rotated_list_with_slicing = rotate_right_with_slicing(original_list, n)

print("Rotated list with slicing technique:", rotated_list_with_slicing)

def rotate_right_without_slicing(lst, n):
```

```
    n = n % len(lst)
    rotated_list = [0] * len(lst)
    for i in range(len(lst)):
        rotated_list[(i + n) % len(lst)] = lst[i]
    return rotated_list

original_list = [1, 2, 3, 4, 5]

n = int(input("Enter the number of times to rotate right: "))

rotated_list_without_slicing = rotate_right_without_slicing(original_list, n)

print("Rotated list without slicing technique:", rotated_list_without_slicing)
```

```
Enter the number of times to rotate right: 2
Rotated list with slicing technique: [4, 5, 1, 2, 3]
Enter the number of times to rotate right: 2
Rotated list without slicing technique: [4, 5, 1, 2, 3]
```

17. Python program to print the patterns given below:

```
def print_pattern(n):
  for i in range(1, n + 1):
    num = i
    for j in range(1, i + 1):
      print(num, end="")
      num = num * 7 - i
    print()


print_pattern(6)
```

```
n = 5

for i in range(1, n + 1):
    for j in range(i):
        print("*", end="")
    print()

n = 5

for i in range(1, n + 1):
    for j in range(n - i):
        print(" ", end="")
```

```
    for k in range(2 * i - 1):
        print("*", end="")

    print()

word = "Python"
for i in range(1, len(word) + 1):
    print(word[:i])
```

```
 1
 212
 318123
 4241641144
 530205143010005
 63624617161200684036
 *
 **
 ***
 ****
 *****
     *
    ***
   *****
  *******
 *********
 P
 Py
 Pyt
 Pyth
 Pytho
 Python
```

Colab paid products - Cancel contracts here