

PYTHON PROGRAMMING

Dt: 27-05-2024

1. Swap 2 numbers using XNOR.

```
exp4.py - C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py (3.12.3)
File Edit Format Run Options Window Help
a=int(input("enter the number:"))
b=int(input("enter the number:"))
a,b=b,a
print(a)
print(b)
print(a*b)

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py
py
enter the number:10
enter the number:20
20
10
30

>>>
```

2. Given input is 10 should get output as 11 using NOT operators.

```
exp4.py - C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py (3.12.3)
File Edit Format Run Options Window Help
a=10
print(~a)

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py
py
11

>>>
```

3. Check whether the given number is odd or even using Bitwise operators.

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled 'exp4.py - C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py (3.12.3)'. It contains the following Python code:

```
#!/usr/bin/python
a=1111
if(a&1==0):
    print("even")
else:
    print("odd")

a=256
if(a&1==0):
    print("even")
else:
    print("odd")
```

In the bottom-right corner, there is an IDLE Shell window titled 'IDLE Shell 3.12.3'. It shows the output of the code:

```
>>> print("odd")
odd
>>> print("even")
even
```

The desktop taskbar at the bottom shows various icons, and the system tray indicates it's 03:36 PM on 27-05-2024.

4. Write a program whether the year is leap year or not.

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled 'exp4.py - C:/Users/sravy/OneDrive/Documents/Technical Training/27-05-2024/exp4.py (3.12.3)'. It contains the following Python code:

```
year = int(input("Enter a year: "))
def is_leap_year(year):
    if year % 4 == 0:
        return True
    elif year % 100 == 0:
        return False
    else:
        return True
if is_leap_year:
    print("year is a leap year")
else:
    print("year is not a leap year.")
```

In the bottom-right corner, there is an IDLE Shell window titled 'IDLE Shell 3.12.3'. It shows the output of the code:

```
>>> Enter a year: 2004
year is a leap year
```

The desktop taskbar at the bottom shows various icons, and the system tray indicates it's 04:09 PM on 27-05-2024.

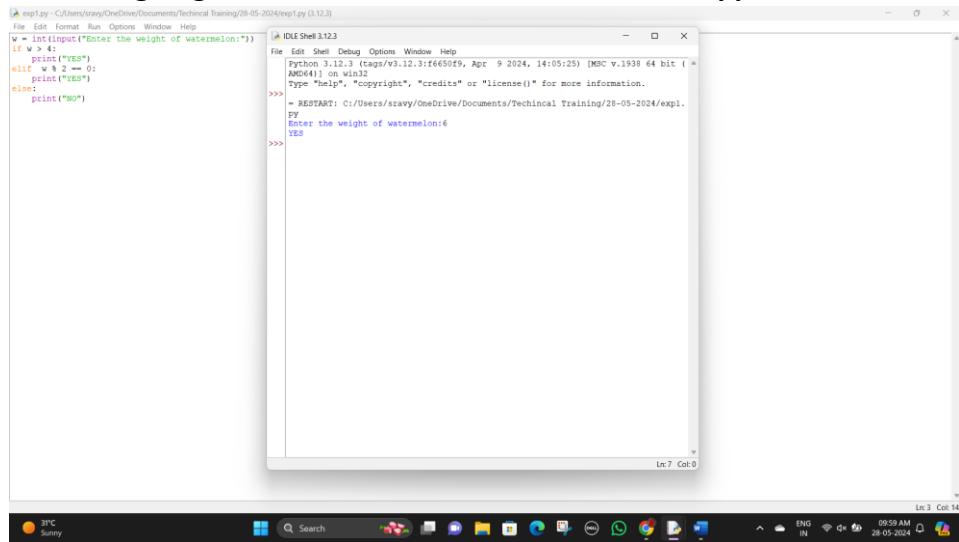
Dt:28-05-2024

CODEFORCE

1. One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.
Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

Input: The first (and the only) input line contains integer number w ($1 \leq w \leq 100$) — the weight of the watermelon bought by the boys.

Output: Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.



```
exp1.py - C:/Users/savy/OneDrive/Documents/Technical Training/28-05-2024/exp1.py (3.12.3)
File Edit Format Run Options Window Help
w = int(input("Enter the weight of watermelon:"))
if w > 42:
    print("YES")
elif w % 2 == 0:
    print("YES")
else:
    print("NO")

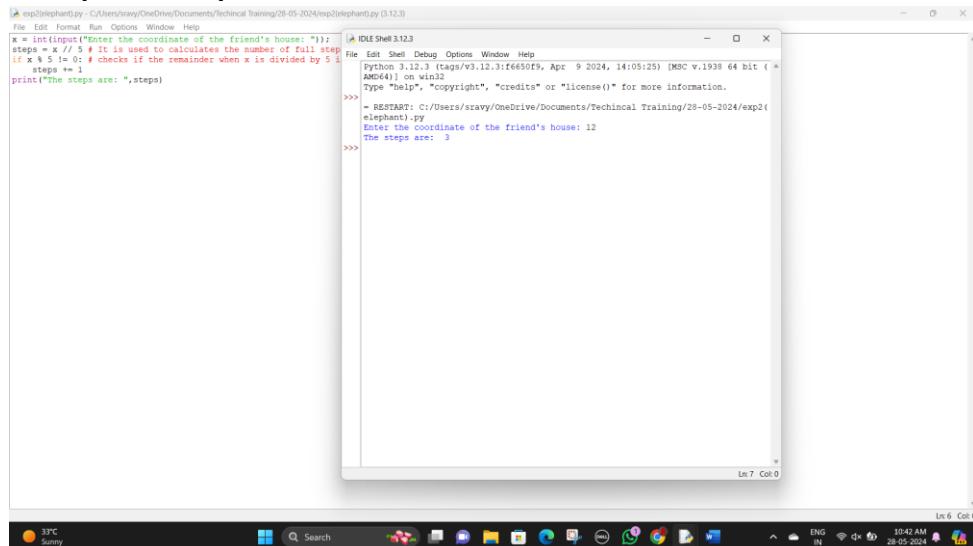
IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/savy/OneDrive/Documents/Technical Training/28-05-2024/exp1.py
PY
Enter the weight of watermelon:6
YES
>>>

Ln 7 Col:8
Ln 3 Col:14
```

2. An elephant decided to visit his friend. It turned out that the elephant's house is located at point 0 and his friend's house is located at point x ($x > 0$) of the coordinate line. In one step the elephant can move 1, 2, 3, 4 or 5 positions forward. Determine, what is the minimum number of steps he need to make in order to get to his friend's house.

Input: The first line of the input contains an integer x ($1 \leq x \leq 1\,000\,000$) — The coordinate of the friend's house.

Output: Print the minimum number of steps that elephant needs to make to get from point 0 to point x .



```
exp2elephant0.py - C:/Users/savy/OneDrive/Documents/Technical Training/28-05-2024/exp2elephant0.py (3.12.3)
File Edit Format Run Options Window Help
x = int(input("Enter the coordinate of the friend's house:"))
steps = x // 5 # It is used to calculates the number of full step
if x % 5 != 0: # checks if the remainder when x is divided by 5
    steps += 1
print("The steps are: ",steps)

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/savy/OneDrive/Documents/Technical Training/28-05-2024/exp2elephant0.py
Enter the coordinate of the friend's house: 12
The steps are: 3
>>>

Ln 7 Col:0
Ln 6 Col:0
```

3. Round-off:

```

exp3]round off.py - C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp3/round off.py (3.12.3)
File Edit Format Run Options Window Help
g>float(input("Enter the number:"))
print(round(x,2))
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp3/
round off.py
Enter the number:3.999999
4.0
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp3/
round off.py
Enter the number:3.777777
3.78
>>>

```

33°C Sunny 11:37 AM 28-05-2024

4. Write a python program to find sum of the first N natural numbers using a loop.

```

exp6[natural numbers using loop.py] - C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp6/natural numbers using loop.py (3.12.3)
File Edit Format Run Options Window Help
N = int(input("Enter the value of N: "))
total_sum = 0
for i in range(1, N + 1):
    total_sum += i
    print(f"The sum of the first {N} natural numbers is: {total_sum}")
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp6/
natural numbers using loop.py
Enter the value of N: 5
The sum of the first 5 natural numbers is: 1
The sum of the first 5 natural numbers is: 3
The sum of the first 5 natural numbers is: 6
The sum of the first 5 natural numbers is: 10
The sum of the first 5 natural numbers is: 15
>>>

```

38°C Sunny 02:06 PM 28-05-2024

5. Write a python program to find the factorial of a given number using a loop.

```

exp7[factorial numbers.py] - C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp7/factorial numbers.py (3.12.3)
File Edit Format Run Options Window Help
N = int(input("Enter the value:"))
factorial = 1
for i in range(1, N + 1):
    factorial *= i
    print(f"The factorial of {N} is: {factorial}")
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/28-05-2024/exp7/
factorial numbers.py
Enter the value: 15
The factorial of 15 is: 1307674368000
>>>

```

02:11 PM 28-05-2024

6. Write a python program to print the Fibonacci series up to N terms.

```
# exp06fibonacci.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\28-05-2024\exp06(fibonacci series).py (3.12.3)
File Edit Format Run Options Window Help
n = int(input("Enter the number of terms: "))
a, b = 0, 1
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/28-05-2024/exp06(fibonacci series).py
Enter the number of terms: 10
0 1 1 2 3 5 8 13 21 34
```

7. Write a python program to count the number of digits in a given numbers.

```
# exp07CountDigits.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\28-05-2024\exp07(CountDigits).py (3.12.3)
num = int(input("Enter the number:"))
count = 0
while num != 0:
    num //= 10
    count += 1
print("Number of digits:", count)
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/28-05-2024/exp07(CountDigits).py
Enter the number:041494
Number of digits: 5
```

8. Write a python program to print the multiplication table of a given number.

```
# exp10Multiplication.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\28-05-2024\exp10(multiplication).py (3.12.3)
num = int(input("Enter a number: "))
for i in range(1, 11):
    print(str(num) + " x " + str(i) + " = " + str(num * i))
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/28-05-2024/exp10(multiplication).py
Enter a number: 20
20 x 1 = 20
20 x 2 = 40
20 x 3 = 60
20 x 4 = 80
20 x 5 = 100
20 x 6 = 120
20 x 7 = 140
20 x 8 = 160
20 x 9 = 180
20 x 10 = 200
```

9. Write a python program to reverse a given number.

```
# Enter the number:1234567890
reverse=0
while n>0:
    digit=n%10
    reverse=reverse*10+digit
    n=n//10
print("Reversed number:", reverse)
```

```
# Enter the number:1234567890
Reversed number: 9876543210
```

10. Write a python program to check if a given number is prime or not.

```
# Enter the value:11
count=0
for i in range(1,8,1):
    if i==1:
        count=count+1
if(count==2):
    print("prime number")
else:
    print("not an prime number")
```

```
# Enter the value:11
not an prime number
```

Dt:29-05-2024

LOOPS

Loops using Time Complexity with XOR

```
# Loops using Time complexity.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\29-05-2024\exp1\Loops using Time complexity.py (3.12.2)
File Edit Format Run Options Window Help
x=int(input("Enter the value:"))
for i in range (1,x+1):
    x=x*1
print(x)
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/29-05-2024/exp1/Loops using Time complexity.py
Enter the value:
1
```

The screenshot shows the Python IDLE Shell 3.12.2 interface. The code defines a variable `x`, initializes it to 1, and then enters a loop where `x` is multiplied by itself for each integer from 1 to the user-specified value. The output shows that for `x=1`, the result is 1.

```
# Loops using Time complexity.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\29-05-2024\exp1\Loops using Time complexity.py (3.12.3)
File Edit Shell Debug Options Window Help
x=int(input("Enter the value:"))
for i in range (1,x+1):
    x=x*1
print(x)
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/29-05-2024/exp1/Loops using Time complexity.py
Enter the value:5
1
3
0
4
1
>>>
```

The screenshot shows the Python IDLE Shell 3.12.3 interface. The code is identical to the previous one, but it prints intermediate values of `x` during the loop iteration. The output shows the value of `x` being updated at each step: 1, 3, 0, 4, 1.

```
# Loops.py - C:\Users\ssavy\OneDrive\Documents\Technical Training\29-05-2024\exp3.py (3.12.3)
File Edit Format Run Options Window Help
L=int(input("Enter the value:"))
R=int(input("Enter the value:"))
for i in range(L,R+1):
    print(i)
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/29-05-2024/exp3.py
2
Enter the value:3
Enter the value:6
2
3
4
5
6
>>>
```

The screenshot shows the Python IDLE Shell 3.12.3 interface. The code defines two variables `L` and `R`, and then prints all integers from `L` to `R`. The output shows the numbers 2, 3, 4, 5, and 6 printed sequentially.

Programs related to List using Loops

1.

```

mypy -c C:\Users\sreya\OneDrive\Documents\Technical Training\29-05-2024\exp4.py (3.12.0)
File Edit Format Run Options Window Help
[12,14,20,'Raghupathi','Sreya Geervani','Swaroopa Rani']
L[1]=12
L[2]=14
L[3]=20
L[4]='Raghupathi'
L[5]='Sreya Geervani'
L[6]=20
L.append('Swaroopa Rani')
print(L)

```

```

File Edit Shell Options Window Help
Python 3.12.3 (tags/v3.12.3:f4edf5c9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:/Users/sreya/OneDrive/Documents/Technical Training/29-05-2024/exp4.py
py
[4, 'Raghupathi', 12, 'Raghupathi', 14, 'Sreya Geervani', 20, 'Swaroopa Rani']

```

2.

```

mypy -c C:\Users\sreya\OneDrive\Documents\Technical Training\29-05-2024\exp5.py (3.12.0)
File Edit Format Run Options Window Help
[1,2,3,4,5]
for i in range(0,4):
    print(i)
print(i,end="")

```

```

File Edit Shell Options Window Help
Python 3.12.3 (tags/v3.12.3:f4edf5c9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:/Users/sreya/OneDrive/Documents/Technical Training/29-05-2024/exp5.py
344

```

3.

```

mypy -c C:\Users\sreya\OneDrive\Documents\Technical Training\29-05-2024\exp5.py (3.12.0)
File Edit Format Run Options Window Help
[1,2,3,4,5]
for i in range(0,4):
    print(i)
print(i)

```

```

File Edit Shell Options Window Help
Python 3.12.3 (tags/v3.12.3:f4edf5c9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:/Users/sreya/OneDrive/Documents/Technical Training/29-05-2024/exp5.py
344
343
342
341

```

LIST METHODS:

- **Append():** Adding element at the end of the list.
- **Extend():** Adding no. of elements to the end of the list.
- **Insert():** Inserting an item at the define index. **[Syntax:{L.insert(2,'program')}]**
- **Remove():** Removing an item by its value from the list. **[Syntax:{L.remove()}]**
- **Pop():** Removes the last item from the list or removes an item by its index from the list.
- **Clear():** Removes all items from the list.

- **Index():** Returns the index of the first matched item.
[Syntax:{print(L.index())}]
- **Count():** Returns the count of number of items passed as an argument.
[Syntax:{print(L.count())}]
- **Copy():** Returns a copy of list. [Syntax:{L!=L.copy()}]
- **Sort():** Sort items in a list in ascending order. [Syntax:{L.sort()}]
- **Reverse():** Reverse the order of items in the list. [Syntax:{L>reverse}]

LIST BUILT-IN FUNCTIONS:

- **Sum():** Sums up the numbers in the list.
- **Max():** Return maximum element of given.
- **Min():** Return minimum element of given list.
- **All():** Returns true if all elements are true or if list is empty.
- **Any():** Returns true if any elements are true. If list is empty, return false.
- **Len():** Returns length of the list or size of the list.

4.

The screenshot shows a Windows desktop environment with the Python IDLE Shell 3.12.3 window open. The terminal window displays the following code and its execution:

```

4.1
print("C:/Users/sravy/OneDrive/Documents/Technical Training/29-05-2024/exp4.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650ef, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello, World!")
Hello, World!
>>>

```

OPERATIONS ON LIST:

1. Concatenation of list (+)
2. Repetition of list (*)
3. Membership on list (in, not in)

SLICING OF LIST:

Dt:30-05-2024

NESTED LIST

- List inside another list is Nested List.
 - Nested List is also called as Multidimensional List.

```
[file:///C:/Users/avasy/OneDrive/Documents/Technical Training/30-05-2024/onefile.py] (3.12.0)

File Edit Format Run Options Window Help

L=([12,25,4,"abc",197,"xyz"],1,[12,49])
print(L)
[[12,25,4,"abc",197,"xyz"],1,[12,49]]
L=[12,25,4,"abc",197,"xyz"]
print(L)
[12,25,4,"abc",197,"xyz"]
print([12])
[12]
>>> print([12,25,4,"abc",197,"xyz"])
[12,25,4,"abc",197,"xyz"]
>>>
>>> print([12,25,4,"abc",197,"xyz"],1,[12,49])
[12,25,4,"abc",197,"xyz"],1,[12,49]
>>>
```

List Comprehension

- List comprehension is an elegant way to define and create list of python .

```
[> wsl:1:~] mpc@mpc-OptiPlex-5090:~/OneDrive/Documents/Technical Training/30-05-2024/expl1
```

```
File Edit Format Run Options Window Help
```

```
Ipython Shell:3123
```

```
File Edit Shell Debug Options Window Help
```

```
Python 3.11.3 (tags/v3.11.3:166552f, Apr  9 2024, 14:02:15) [MSC v.1938 64 bit (AMD64)]
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>> print([x ** 2 for x in range(1,11) if x % 2 ==0])
```

```
List Comprehension contains Output Expression - Input Sequence and Variable
```

```
[4, 16, 36, 64, 100]
```

```
>>> print([x for x in range(1,11) if x % 2 ==0])
```

```
List Comprehension contains Output Expression - Input Sequence and Variable
```

```
[1, 3, 5, 7, 9]
```

```
>>> print([x for x in range(1,11) if x % 2 ==0])
```

```
List Comprehension contains Output Expression - Input Sequence and Variable
```

```
[1, 3, 5, 7, 9]
```

```
>>>
```

```
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/30-05-2024/expl1/list_comprehension.py
```

```
[4, 16, 36, 64, 100]
```

```
[4, 16, 36, 64, 100]
```

Take two different lists, merge them and print a new sorted list.

```
#> mrgmerging lists - C:\Users\sayav\OneDrive\Documents\Technical Training\30-05-2024\exp\merging lists.py (3.12.0)
File Edit Shell Debug Options Window Help
#Take 2 different lists , merge them and print a new sorted list.
l1=[1,2,3,4]
l2=[4,5,6,7]
l1.extend(l2)
l1.sort()
print(l1)
>>>
[1, 2, 3, 4, 5, 6, 7]
```

Input a list and sort the elements in the list based on its length.

```
#> sort based in length - C:\Users\sayav\OneDrive\Documents\Technical Training\30-05-2024\exp\sort based in length.py (3.12.0)
File Edit Shell Debug Options Window Help
#> sort based in length
l = ["1*", "11", "425", "2104"]
l.sort(key=len)
print(l)
>>>
['1*', '11', '425', '2104']
```

Print the list after deleting the duplicate elements in it.

```
#> mrgdeleting duplicate list - C:\Users\sayav\OneDrive\Documents\Technical Training\30-05-2024\exp\deleting duplicate list.py (3.12.0)
File Edit Shell Debug Options Window Help
#Print the list after deleting the duplicate elements in it.
l = [1, 2, 3, 2, 3, 55, 13, 14, 1, 8, 1, 4]
l1 = []
for i in l:
    if i not in l1:
        l1.append(i)
print(l1)
>>>
[1, 2, 3, 55, 13, 14, 8, 4]
```

Print the elements in the list which has occurred odd number of times.

```
#exp6odd or even count.py : C:\Users\sayav\OneDrive\Documents\Technical Training\30-05-2024\exp6odd or even count.py (3.12.0)
File Edit Format Run Options Window Help
#Print the elements in the list which has occurred odd number of times.
def printOddOrEvenCount():
    b=[]
    for i in a:
        if a.count(i)%2 !=0 and i not in b:
            b.append(i)
    print(b)

elements = [4,7,5,2,2,5,5,7,3,4]
counts = {}
for element in elements:
    if element in counts:
        counts[element] += 1
    else:
        counts[element] = 1
for element, count in counts.items():
    if count % 2 != 0:
        print(element)

#RESTART: C:/Users/sayav/OneDrive/Documents/Technical Training/30-05-2024/exp6odd or even count.py
Python 3.12.3 (appenv1.12.3:f665f5f9, Apr 9 2024, 14:05:25) | [GCC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/sayav/OneDrive/Documents/Technical Training/30-05-2024/exp6odd or even count.py
1 5 6 4 2 13 15 66
[1, 5, 6, 4, 2, 13, 15, 66]
>>> 3
3
>>> 3
```

Read a list and print sum of three minimum elements in the list.

A screenshot of a Windows desktop environment. In the foreground, a Notepad window displays a Python script named `sum_of_three_minimum_elements.py`. The script contains code to read a list of integers from input, sort it, and then calculate the sum of the first three minimum elements. The Notepad window has a title bar, menu bar, and status bar indicating line 7 and column 0. In the background, an IDLE Shell window titled "IDLE Shell 3.12.3" is running Python 3.12.3. It shows the script's execution and output. The output includes the command, the sorted list [4, 2, 3, 1], and the calculated sum 6. The IDLE window also has a title bar, menu bar, and status bar.

```
# Read a list and print sum of three minimum elements in the list.
a=list(map(int,input().split()))
a.sort()
print(sum(a[:3]))
```

```
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f66650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.

>>> - RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/30-05-2024/exp7/
sum of three minimum elements.py
4 2 3 1 5
6
>>>
```

Segregate the given list as even elements first in descending order and then odd elements next in ascending order

```
C:\Users\Avinash\OneDrive\Documents\Technical Training\10-05-2024\Ascending order(3.12.3)
File Edit Shell Debug Options Window Help
# This program takes a list of integers as input and prints them such that the first half of the list contains elements in descending order and the second half contains elements in ascending order.
# Example: If the given list is [1, 2, 3, 4, 5], the output will be [5, 4, 3, 2, 1].
# Input: Enter the list of integers separated by commas: 1, 2, 3, 4, 5
# Output: [5, 4, 3, 2, 1]
# Author: Avinash Kulkarni
# Date: 10-05-2024

def print_descending_order(input):
    n = len(input)
    a = []
    b = []
    for i in range(n):
        if i < n // 2:
            a.append(input[i])
        else:
            b.append(input[i])
    a.sort(reverse=True)
    b.sort()
    return a + b

input = list(map(int, input().split()))
print(print_descending_order(input))

# Python 3.12.3 (tags/v3.12.3:f499f79, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
# Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:/Users/avinash/OneDrive/Documents/Technical Training/10-05-2024/ascending_order.py
ascending_order.py
[5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]
>>>
```

print product of the elements in the list which are within the given range

```

# product of elements in given range.py
# File Edit Format Run Options Window Help
#>>> print(product of elements in given range)
#>>> print product of the elements in the list which are within the given range
#>>> a=list(map(int,input().split()))
#>>> m=int(input())
#>>> n=max(int,a)
#>>> b=[]
#>>> for i in range(m,n+1):
#>>>     if i in a:
#>>>         b.append(i)
#>>> for i in b:
#>>>     p*=i
#>>> print(p)
#>>>

```

```

1 2 3 4 5 6 7 8 9 10

```

Given lists in a list, find the maximum sum of elements of the list in a list of lists and print its index number .

```

# exp10/index number.py - C:\Users\ssavvy\OneDrive\Documents\Technical Training\30-05-2024\exp10\index number.py (3.12.3)
File Edit Format Run Options Window Help
# Given lists in a list, find the maximum sum of elements of the list in a list of lists and print its index number.
#>>> print(max([sum(i) for i in range(10)]))
#>>> a=[map(int,input().split()) for i in range(10)]
#>>> for i in a:
#>>>     g=sum(i)
#>>>     if g>s:
#>>>         s=g
#>>>         ind=a.index(i)
#>>> print(ind)
#>>>

```

```

1 2 3 4 5 2
1 2 3
2 7 4
2

```

TUPLE

- In tuple we can't update the elements just as list.

TUPLE BUILT-IN FUNCTIONS:

- **Sum():** Sums up the numbers in the tuple.
- **Max():** Return maximum element of given.
- **Min():** Return minimum element of given tuple.
- **All():** Returns true if all elements are true or if tuple is empty.
- **Any():** Returns true if any elements are true. If tuple is empty, return false.
- **Len():** Returns length of the list or size of the tuple.

OPERATIONS ON TUPLE:

1. Concatenation of tuple (+)
2. Repetition of tuple (*)
3. Membership on tuple (in, not in)

SLICING IN TUPLE:

- slice--> () using colon
- T[:]-->print elements from tuple
- T[2:]-->print all elements from 2 index in tuple
- T[:5]--> print all elements from tuple till ending index 4
- T[2:6]--> print elements from index 2 till 5
- [T[::2]]--> 2(index+2elements)

NESTED TUPLE:

```
# In [1]: C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\expt1.py (3.12.3)
# File Edit Kernel Run Options Window Help
# In [1]: NameError: name 'another' tuple is needed tuple
# In [2]: 2,3,(4,5),("hi","ssavy"),10
print(T)
print(T[1])
print(T[1][1])
print(T[1][1][1])
T=(4,5,6,(3,4,6),(6,3,8),("hi",))
print(T)
# Out[1]:
# (2, 3, (4, 5), ('hi', 'ssavy'), 10)
# (4, 5)
# 6
# ((4, 5, 6, (3, 4, 6), (6, 3, 8), ('hi',)), 1)
# In [2]:
```

Remove an element from an specified index in tuple.

```
# In [1]: C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\expt2.py (3.12.3)
# File Edit Format Run Options Window Help
# Remove an element from an specified index in tuple.
tupple=(input().split(','))
for i in tupple:
    print(i)
i=int(input())
tupple.pop(i)
print(tupple)
# Out[1]:
# 1,2,3,4,5,6,7,8,9,10
# 1
# 2,3,4,5,6,7,8,9,10
# In [2]:
```

Write a python program to reverse a tuple.

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp3.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to reverse a tuple
t = tuple(input().split(','))
t = t[::-1]
print("Original tuple:", t)
print("Reversed tuple:", t)
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp3.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp3.py
Original tuple: ('1', '2', '3', '4', '5', '6', '7')
Reversed tuple: ('7', '6', '5', '4', '3', '2', '1')

In 5 Col 0
```

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp4.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to find the repeated items of a tuple.
t = tuple(input().split(','))
x = {}
for item in t:
    if item in x:
        x[item] += 1
    else:
        x[item] = 1
y = {item for item, count in x.items() if count > 1}
print("y:", y)

my_tuple=(1,2,3,2,4,2,5,2,6)
max_count=max(my_tuple.count(item) for item in my_tuple)
print("Maximum count:{max_count}")
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp4.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp4.py
{'2': 2, '1': 2, '5': 2, '4': 2}
y: {'1', '2', '5'}
Maximum count:4

In 17 Col 0
```

Write a python program to find the repeated items of a tuple.

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp4.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to find the repeated items of a tuple.
t = tuple(input().split(','))
x = {}
for item in t:
    if item in x:
        x[item] += 1
    else:
        x[item] = 1
y = {item for item, count in x.items() if count > 1}
print("y:", y)

my_tuple=(1,2,3,2,4,2,5,2,6)
max_count=max(my_tuple.count(item) for item in my_tuple)
print("Maximum count:{max_count}")
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp5.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp5.py
{'2': 2, '1': 2, '5': 2, '4': 2}
y: {'1', '2', '5'}
Maximum count:4

In 17 Col 0
```

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp5.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to remove an empty tuple(s) from a list of tuples.
a = [(1,2),(),(1,2),(1,2,3),(5)]
b = [t for t in a if t]
print("a:", a)
print("b:", b)
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp5.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp5.py
a: [(1, 2), (), (1, 2), (1, 2, 3), 5]
b: [(1, 2), (1, 2), (1, 2, 3), 5]

In 2 Col 0
```

Write a python program to remove an empty tuple(s) from a list of tuples.

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp5.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to remove an empty tuple(s) from a list of tuples.
a = [(1,2),(),(1,2),(1,2,3),(5)]
b = [t for t in a if t]
print("a:", a)
print("b:", b)
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp6.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp6.py
a: [(1, 2), (), (1, 2), (1, 2, 3), 5]
b: [(1, 2), (1, 2), (1, 2, 3), 5]

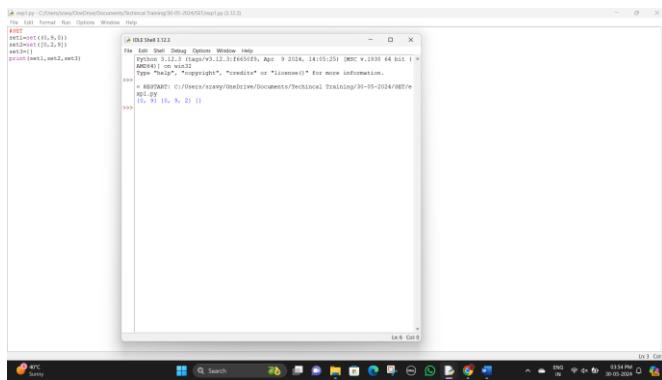
In 2 Col 0
```

```
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp6.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to find the maximum value in a list.
l = [1,2,3,4,5,6,7,8,9,10]
max_val = max(l)
print("Max value is:", max_val)
>>>
#> repl.py -C:\Users\ssavy\OneDrive\Documents\Technical Training\30-05-2024\TUPLE\Exmp6.py (3.12.3)
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ssavy/OneDrive/Documents/Technical Training/30-05-2024/TUPLE/Exmp6.py
Max value is: 10

In 2 Col 0
```

SET

- A set is an unordered collection data type that is iterable, mutable and has no duplicate elements.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.



The screenshot shows an IDLE Python shell window. The code entered is:

```
set1={1,2,3}
set2={3,4,5}
set3=set1.union(set2)
print(set3)
```

The output displayed is:

```
{1, 2, 3, 4, 5}
```

SET METHODS:

- **Add():** Adds an elements to the set. [Syntax: `s.add()`]
- **Update():** Updates the set with the union of itself and others. [Syntax: `s.update()`]
- **Copy():** Returns a copy of the set. [Syntax: `s.copy()`]
- **Clear():** Removes all elements from the ser. [Syntax: `s.clear()`]
- **Remove():** Removes an element from the set. If the element is not a member, raise a KeyError. [Syntax: `s.remove()`]
- **Pop():** Removes and returns an arbitrary set element. Raise KeyError if the set is empty. [Syntax: `print(s.pop())`]
- **Discard():** Removes an element from the set if it is a member.(Do nothing if not in set) [Syntax: `s.discard()`]
- **Union():** Returns the union of sets in a new set. [Syntax: `print(a.union(b))`]
- **Intersection():** Retuen the intersection of two sets as a new set.
[Syntax: `print(a.intersection(b))`]
- **Intersection_update():** Updates the set with the intersection of itself and another.[Syntax: `a.intersection_update*b)`]
- **Difference():** Returns the difference of two or more sets as a new set.
[Syntax: `print(a.difference(b))`]
- **Difference_update():** Removes all elements of another set from this set.
[Syntax: `a.difference_update(b)`]
- **Symmetric_difference():** Returns the symmetric difference of two sets as a new set.
[Syntax: `print(a.symmetric_difference(b))`]
- **Symmetric_difference_update():** Updates a set with the symmetric difference of itself and another. [Syntax: `a.Symmetric_difference_update(b)`]
- **Isdisjoint():** Returns True if two sets have a null intersection. [Syntax:
`print(a.isdisjoint(b))`]
- **Issubset():** Returns True if another set contains this set. [Syntax: `print(a.issubset(a))`]

- **issuperset():** Returns True if this set contains another set. [Syntax: print(b.issuperset(a))]

```

explorer C:\Users\scavy\OneDrive\Documents\Technical Training\30-05-2024\SET-Pset.py (3.12.3)
File Edit Format Run Options Window Help
#SET METHODS
s=set([1,2,3,4,5,6,'Strava'])
s|={1,2,3,4}
s.update(t)
t=[1,2,3,4]
s.add(14)
s.remove(5)
s.discard('hi')
s.pop()
print(s)
print(s.union(p))
print(s.intersection(p))
print(s.difference(p))
print(s.symmetric_difference(p))
print(s|p)
print(s.union_symmetric_difference(p))
print(s|p)
print(s.isdisjoint(p))
print(s.issubset(p))
print(s.issuperset(s))

```

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f665c0f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/scavy/OneDrive/Documents/Technical Training/30-05-2024/SET-1
venv2\py
[1, 2, 3, 4, 6, 10, 'Strava', 14]
[1, 2, 3, 4, 6, 10, 'Strava', 14]
[1, 2, 3, 4]
[2, 3, 4]
[2, 3, 4]
set()
set()
set()
set()
set()
set()
set()
False
True
True
True
True

```

Dt:31-05-2024

SET IN-BUILT FUNCTIONS:

- **Sum():** Sums of all elements in the set. [Syntax: print(sum())]
- **Max():** Return largest item of given. [Syntax: print(max())]
- **Min():** Return smallest item of given set. [Syntax: print(min())]
- **All():** Returns true if all elements of the set are true. [Syntax: print(all())]
- **Any():** Returns true if any elements are true. If set is empty, return false. [Syntax: print(any())]
- **Len():** Returns the length in the set. [Syntax: print(len())]
- **Sorted():** Return a new sorted list from elements in the set. [Syntax: print(sorted())]

```

explorer C:\Users\scavy\OneDrive\Documents\Technical Training\31-05-2024\SET-2\exp1.py (3.12.3)
File Edit Format Run Options Window Help
#SET BUILT-IN FUNCTIONS
s={1,0,3,4,5,2,7,6,8,9}
sum(s)
print(max(s))
print(min(s))
print(all(s))
print(any(s))
print(sorted(s))

```

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f665c0f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/scavy/OneDrive/Documents/Technical Training/31-05-2024/SET-2
/expli-py
45
9
0
False
True
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```

OPERATIONS ON SET:

- Membership on tuple(in, not in).
- Equal(==), Not Equal(!=).
- Subset(<=), Proper Subset(<).
- Superset(>=), Proper Superset(>).
- Union(|), Intersection(&).
- Difference(-), Symmetric_Difference(^).

The screenshot shows the Python IDLE Shell window. The code in the editor is:

```
#OPERATIONS ON SET
set1={1,2,3,4}
set2={4,5,6,7,8,9,10}
print(a < b)
print(a == b)
print(a >= b)
print(a <= b)
print(a|b)
```

The output in the shell is:

```
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/31-05-2024/SET-2/esp2.py
False
False
True
True
True
```

The status bar at the bottom right shows the date and time: 11-05-2024 10:15 AM.

FROZEN SET:

- They are immutable sets.
- They can be created using function frozenset().
- This datatype supports methods like copy(), difference(),etc

Code:

The screenshot shows the Python IDLE Shell window. The code in the editor is:

```
set1={10,20,30,40,50}
set2=frozenset({40,70,10,30,40,80,20,50})
print(set1)
print(set2.issuperset(set1))
```

The output in the shell is:

```
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/31-05-2024/SET-2/esp3.py
{10, 20, 30, 40, 50}
True
```

The status bar at the bottom right shows the date and time: 11-05-2024 10:21 AM.

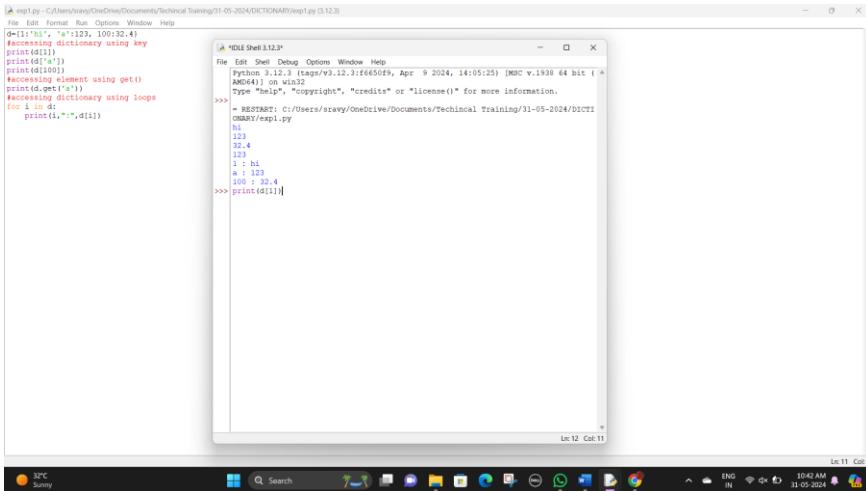
SET COMPREHENSION:

- Returns a set based on existing iterable.
- Syntax: {expression(variable) for variable in input_set[predicate][,...]}

DICTIONARY

- ❖ A dictionary is a collection which is unordered , changeable and indexed.
- ❖ In python dictionaries are written in curly brackets.

ACCESSING ELEMENTS IN DICTIONARY:



```
exp1.py - C:\Users\sravya\OneDrive\Documents\Technical Training\31-05-2024\DICTIONARY\exp1.py (3.12.0)
File Edit Format Run Options Window Help
d={"hi": "123", "a":123, 100:32,4}
#adding an element to dictionary
print(d)
print(d['a'])
print(d.get('a'))
#accessing element using get()
print(d.get("a"))
#deleting an element from dictionary using loops
for i in d:
    print(i,"-",d[i])
>>> print(d['hi'])

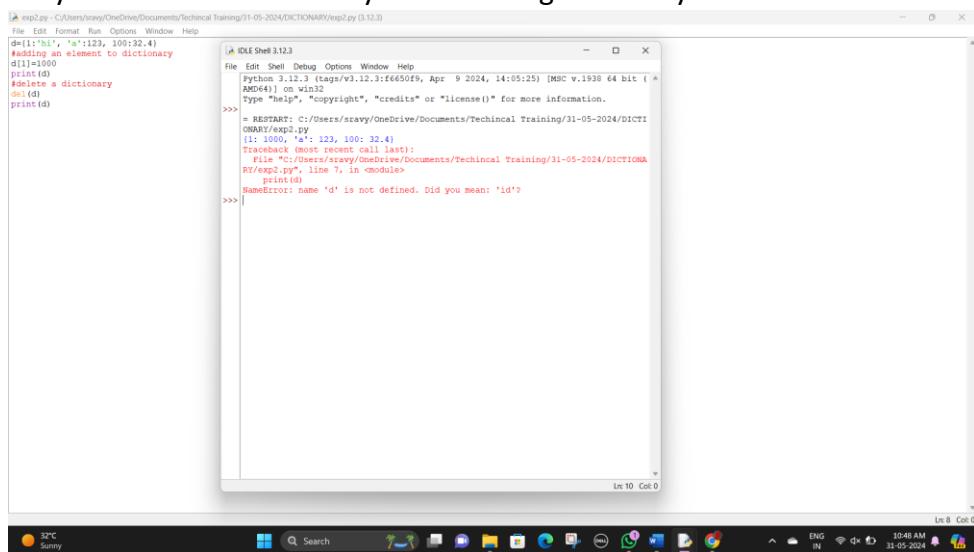
Ln 12 Col 0
```

```
IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f16650f9, Apr 9 2024, 14:05:25) [MSC v.1930 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp1.py
hi
123
32.4
100
i : hi
a : 123
100 : 32,4
>>> print(d['hi'])

Ln 11 Col 0
```

MODIFY AND DELETE A DICTIONARY:

- ❖ Only values in the dictionary can be changed but keys should be immutable.



```
exp2.py - C:\Users\sravya\OneDrive\Documents\Technical Training\31-05-2024\DICTIONARY\exp2.py (3.12.0)
File Edit Format Run Options Window Help
d={"hi": "123", "a":123, 100:32,4}
#adding an element to dictionary
print(d)
#delete a dictionary
del(d)
print(d)
>>>
= RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp2.py
{'hi': '123', 'a': 123, 100: 32, 4}
Traceback (most recent call last):
  File "C:/Users/sravya/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp2.py", line 7, in <module>
    print(d)
NameError: name 'd' is not defined. Did you mean: 'id'?

Ln 10 Col 0
```

```
IDLE Shell 3.12.0
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f16650f9, Apr 9 2024, 14:05:25) [MSC v.1930 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp2.py
{'hi': '123', 'a': 123, 100: 32, 4}
Traceback (most recent call last):
  File "C:/Users/sravya/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp2.py", line 7, in <module>
    print(d)
NameError: name 'd' is not defined. Did you mean: 'id'?

Ln 8 Col 0
```

DICTIONARY METHODS:

- ❖ **Get():**
- ❖ **Update():** Adds two or more dictionaries.
- ❖ **Pop():** Removes and returns an element from a dictionary having the given key.
- ❖ **Popitem():** Removes the arbitrary key value pair from the dictionary and returns it as tuple.
- ❖ **Clear():** The clear() method removes all items from the dictionary.
- ❖ **Keys():** returns list of dictionary dictionaries keys.
- ❖ **Values():** returns a list of all the values available in a given dictionary.
- ❖ **Items():** returns a list of dictionaries(key, value) tuple pairs.

- ❖ **Fromkeys():** Create a new dictionary with keys from sequence and values set to value.
- ❖ **Setdefault():** Set dictionary[key]=default if key is not already in dictionary.

NESTED DICTIONARY:

- ❖ Dictionary inside another dictionary is nested dictionary.

```

exp4.py - C:/Users/sravy/OneDrive/Documents/Techincal Training/31-05-2024/DICTIONARY/exp4.py (3.12.3)
File Edit Shell Options Window Help
File Edit Format Run Options Window Help
IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/sravy/OneDrive/Documents/Techincal Training/31-05-2024/DICTIONARY/exp4.py
d1:
d2:
d3:
d4:
d5:
d6:
d7:
d8:
d9:
d10:
d11:
d12:
d13:
d14:
d15:
d16:
d17:
d18:
d19:
d20:
d21:
d22:
d23:
d24:
d25:
d26:
d27:
d28:
d29:
d30:
d31:
d32:
d33:
d34:
d35:
d36:
d37:
d38:
d39:
d40:
d41:
d42:
d43:
d44:
d45:
d46:
d47:
d48:
d49:
d50:
d51:
d52:
d53:
d54:
d55:
d56:
d57:
d58:
d59:
d60:
d61:
d62:
d63:
d64:
d65:
d66:
d67:
d68:
d69:
d70:
d71:
d72:
d73:
d74:
d75:
d76:
d77:
d78:
d79:
d80:
d81:
d82:
d83:
d84:
d85:
d86:
d87:
d88:
d89:
d90:
d91:
d92:
d93:
d94:
d95:
d96:
d97:
d98:
d99:
d100:
d101:
d102:
d103:
d104:
d105:
d106:
d107:
d108:
d109:
d110:
d111:
d112:
d113:
d114:
d115:
d116:
d117:
d118:
d119:
d120:
d121:
d122:
d123:
d124:
d125:
d126:
d127:
d128:
d129:
d130:
d131:
d132:
d133:
d134:
d135:
d136:
d137:
d138:
d139:
d140:
d141:
d142:
d143:
d144:
d145:
d146:
d147:
d148:
d149:
d150:
d151:
d152:
d153:
d154:
d155:
d156:
d157:
d158:
d159:
d160:
d161:
d162:
d163:
d164:
d165:
d166:
d167:
d168:
d169:
d170:
d171:
d172:
d173:
d174:
d175:
d176:
d177:
d178:
d179:
d180:
d181:
d182:
d183:
d184:
d185:
d186:
d187:
d188:
d189:
d190:
d191:
d192:
d193:
d194:
d195:
d196:
d197:
d198:
d199:
d200:
d201:
d202:
d203:
d204:
d205:
d206:
d207:
d208:
d209:
d210:
d211:
d212:
d213:
d214:
d215:
d216:
d217:
d218:
d219:
d220:
d221:
d222:
d223:
d224:
d225:
d226:
d227:
d228:
d229:
d229:
d230:
d231:
d232:
d233:
d234:
d235:
d236:
d237:
d238:
d239:
d239:
d240:
d241:
d242:
d243:
d244:
d245:
d246:
d247:
d248:
d249:
d249:
d250:
d251:
d252:
d253:
d254:
d255:
d256:
d257:
d258:
d259:
d259:
d260:
d261:
d262:
d263:
d264:
d265:
d266:
d267:
d268:
d269:
d269:
d270:
d271:
d272:
d273:
d274:
d275:
d276:
d277:
d278:
d279:
d279:
d280:
d281:
d282:
d283:
d284:
d285:
d286:
d287:
d288:
d289:
d289:
d290:
d291:
d292:
d293:
d294:
d295:
d296:
d297:
d298:
d299:
d299:
d300:
d301:
d302:
d303:
d304:
d305:
d306:
d307:
d308:
d309:
d309:
d310:
d311:
d312:
d313:
d314:
d315:
d316:
d317:
d318:
d319:
d319:
d320:
d321:
d322:
d323:
d324:
d325:
d326:
d327:
d328:
d329:
d329:
d330:
d331:
d332:
d333:
d334:
d335:
d336:
d337:
d338:
d339:
d339:
d340:
d341:
d342:
d343:
d344:
d345:
d346:
d347:
d348:
d349:
d349:
d350:
d351:
d352:
d353:
d354:
d355:
d356:
d357:
d358:
d359:
d359:
d360:
d361:
d362:
d363:
d364:
d365:
d366:
d367:
d368:
d369:
d369:
d370:
d371:
d372:
d373:
d374:
d375:
d376:
d377:
d378:
d379:
d379:
d380:
d381:
d382:
d383:
d384:
d385:
d386:
d387:
d388:
d389:
d389:
d390:
d391:
d392:
d393:
d394:
d395:
d396:
d397:
d398:
d399:
d399:
d400:
d401:
d402:
d403:
d404:
d405:
d406:
d407:
d408:
d409:
d409:
d410:
d411:
d412:
d413:
d414:
d415:
d416:
d417:
d418:
d419:
d419:
d420:
d421:
d422:
d423:
d424:
d425:
d426:
d427:
d428:
d429:
d429:
d430:
d431:
d432:
d433:
d434:
d435:
d436:
d437:
d438:
d439:
d439:
d440:
d441:
d442:
d443:
d444:
d445:
d446:
d447:
d448:
d449:
d449:
d450:
d451:
d452:
d453:
d454:
d455:
d456:
d457:
d458:
d459:
d459:
d460:
d461:
d462:
d463:
d464:
d465:
d466:
d467:
d468:
d469:
d469:
d470:
d471:
d472:
d473:
d474:
d475:
d476:
d477:
d478:
d479:
d479:
d480:
d481:
d482:
d483:
d484:
d485:
d486:
d487:
d488:
d489:
d489:
d490:
d491:
d492:
d493:
d494:
d495:
d496:
d497:
d498:
d499:
d500:
d501:
d502:
d503:
d504:
d505:
d506:
d507:
d508:
d509:
d509:
d510:
d511:
d512:
d513:
d514:
d515:
d516:
d517:
d518:
d519:
d519:
d520:
d521:
d522:
d523:
d524:
d525:
d526:
d527:
d528:
d529:
d529:
d530:
d531:
d532:
d533:
d534:
d535:
d536:
d537:
d538:
d539:
d539:
d540:
d541:
d542:
d543:
d544:
d545:
d546:
d547:
d548:
d549:
d549:
d550:
d551:
d552:
d553:
d554:
d555:
d556:
d557:
d558:
d559:
d559:
d560:
d561:
d562:
d563:
d564:
d565:
d566:
d567:
d568:
d569:
d569:
d570:
d571:
d572:
d573:
d574:
d575:
d576:
d577:
d578:
d579:
d579:
d580:
d581:
d582:
d583:
d584:
d585:
d586:
d587:
d588:
d589:
d589:
d590:
d591:
d592:
d593:
d594:
d595:
d596:
d597:
d598:
d599:
d599:
d600:
d601:
d602:
d603:
d604:
d605:
d606:
d607:
d608:
d609:
d609:
d610:
d611:
d612:
d613:
d614:
d615:
d616:
d617:
d618:
d619:
d619:
d620:
d621:
d622:
d623:
d624:
d625:
d626:
d627:
d628:
d629:
d629:
d630:
d631:
d632:
d633:
d634:
d635:
d636:
d637:
d638:
d639:
d639:
d640:
d641:
d642:
d643:
d644:
d645:
d646:
d647:
d648:
d649:
d649:
d650:
d651:
d652:
d653:
d654:
d655:
d656:
d657:
d658:
d659:
d659:
d660:
d661:
d662:
d663:
d664:
d665:
d666:
d667:
d668:
d669:
d669:
d670:
d671:
d672:
d673:
d674:
d675:
d676:
d677:
d678:
d679:
d679:
d680:
d681:
d682:
d683:
d684:
d685:
d686:
d687:
d688:
d689:
d689:
d690:
d691:
d692:
d693:
d694:
d695:
d696:
d697:
d698:
d699:
d699:
d700:
d701:
d702:
d703:
d704:
d705:
d706:
d707:
d708:
d709:
d709:
d710:
d711:
d712:
d713:
d714:
d715:
d716:
d717:
d718:
d719:
d719:
d720:
d721:
d722:
d723:
d724:
d725:
d726:
d727:
d728:
d729:
d729:
d730:
d731:
d732:
d733:
d734:
d735:
d736:
d737:
d738:
d739:
d739:
d740:
d741:
d742:
d743:
d744:
d745:
d746:
d747:
d748:
d749:
d749:
d750:
d751:
d752:
d753:
d754:
d755:
d756:
d757:
d758:
d759:
d759:
d760:
d761:
d762:
d763:
d764:
d765:
d766:
d767:
d768:
d769:
d769:
d770:
d771:
d772:
d773:
d774:
d775:
d776:
d777:
d778:
d779:
d779:
d780:
d781:
d782:
d783:
d784:
d785:
d786:
d787:
d788:
d789:
d789:
d790:
d791:
d792:
d793:
d794:
d795:
d796:
d797:
d798:
d799:
d799:
d800:
d801:
d802:
d803:
d804:
d805:
d806:
d807:
d808:
d809:
d809:
d810:
d811:
d812:
d813:
d814:
d815:
d816:
d817:
d818:
d819:
d819:
d820:
d821:
d822:
d823:
d824:
d825:
d826:
d827:
d828:
d829:
d829:
d830:
d831:
d832:
d833:
d834:
d835:
d836:
d837:
d838:
d839:
d839:
d840:
d841:
d842:
d843:
d844:
d845:
d846:
d847:
d848:
d849:
d849:
d850:
d851:
d852:
d853:
d854:
d855:
d856:
d857:
d858:
d859:
d859:
d860:
d861:
d862:
d863:
d864:
d865:
d866:
d867:
d868:
d869:
d869:
d870:
d871:
d872:
d873:
d874:
d875:
d876:
d877:
d878:
d879:
d879:
d880:
d881:
d882:
d883:
d884:
d885:
d886:
d887:
d888:
d889:
d889:
d890:
d891:
d892:
d893:
d894:
d895:
d896:
d897:
d898:
d899:
d899:
d900:
d901:
d902:
d903:
d904:
d905:
d906:
d907:
d908:
d909:
d909:
d910:
d911:
d912:
d913:
d914:
d915:
d916:
d917:
d918:
d919:
d919:
d920:
d921:
d922:
d923:
d924:
d925:
d926:
d927:
d928:
d929:
d929:
d930:
d931:
d932:
d933:
d934:
d935:
d936:
d937:
d938:
d939:
d939:
d940:
d941:
d942:
d943:
d944:
d945:
d946:
d947:
d948:
d949:
d949:
d950:
d951:
d952:
d953:
d954:
d955:
d956:
d957:
d958:
d959:
d959:
d960:
d961:
d962:
d963:
d964:
d965:
d966:
d967:
d968:
d969:
d969:
d970:
d971:
d972:
d973:
d974:
d975:
d976:
d977:
d978:
d979:
d979:
d980:
d981:
d982:
d983:
d984:
d985:
d986:
d987:
d988:
d989:
d989:
d990:
d991:
d992:
d993:
d994:
d995:
d996:
d997:
d998:
d999:
d999:
d1000:
d1001:
d1002:
d1003:
d1004:
d1005:
d1006:
d1007:
d1008:
d1009:
d1009:
d1010:
d1011:
d1012:
d1013:
d1014:
d1015:
d1016:
d1017:
d1018:
d1019:
d1019:
d1020:
d1021:
d1022:
d1023:
d1024:
d1025:
d1026:
d1027:
d1028:
d1029:
d1029:
d1030:
d1031:
d1032:
d1033:
d1034:
d1035:
d1036:
d1037:
d1038:
d1039:
d1039:
d1040:
d1041:
d1042:
d1043:
d1044:
d1045:
d1046:
d1047:
d1048:
d1049:
d1049:
d1050:
d1051:
d1052:
d1053:
d1054:
d1055:
d1056:
d1057:
d1058:
d1059:
d1059:
d1060:
d1061:
d1062:
d1063:
d1064:
d1065:
d1066:
d1067:
d1068:
d1069:
d1069:
d1070:
d1071:
d1072:
d1073:
d1074:
d1075:
d1076:
d1077:
d1078:
d1079:
d1079:
d1080:
d1081:
d1082:
d1083:
d1084:
d1085:
d1086:
d1087:
d1088:
d1089:
d1089:
d1090:
d1091:
d1092:
d1093:
d1094:
d1095:
d1096:
d1097:
d1098:
d1099:
d1099:
d1100:
d1101:
d1102:
d1103:
d1104:
d1105:
d1106:
d1107:
d1108:
d1109:
d1109:
d1110:
d1111:
d1112:
d1113:
d1114:
d1115:
d1116:
d1117:
d1118:
d1119:
d1119:
d1120:
d1121:
d1122:
d1123:
d1124:
d1125:
d1126:
d1127:
d1128:
d1129:
d1129:
d1130:
d1131:
d1132:
d1133:
d1134:
d1135:
d1136:
d1137:
d1138:
d1139:
d1139:
d1140:
d1141:
d1142:
d1143:
d1144:
d1145:
d1146:
d1147:
d1148:
d1149:
d1149:
d1150:
d1151:
d1152:
d1153:
d1154:
d1155:
d1156:
d1157:
d1158:
d1159:
d1159:
d1160:
d1161:
d1162:
d1163:
d1164:
d1165:
d1166:
d1167:
d1168:
d1169:
d1169:
d1170:
d1171:
d1172:
d1173:
d1174:
d1175:
d1176:
d1177:
d1178:
d1179:
d1179:
d1180:
d1181:
d1182:
d1183:
d1184:
d1185:
d1186:
d1187:
d1188:
d1189:
d1189:
d1190:
d1191:
d1192:
d1193:
d1194:
d1195:
d1196:
d1197:
d1198:
d1198:
d1199:
d1199:
d1200:
d1201:
d1202:
d1203:
d1204:
d1205:
d1206:
d1207:
d1208:
d1209:
d1209:
d1210:
d1211:
d1212:
d1213:
d1214:
d1215:
d1216:
d1217:
d1218:
d1219:
d1219:
d1220:
d1221:
d1222:
d1223:
d1224:
d1225:
d1226:
d1227:
d1228:
d1229:
d1229:
d1230:
d1231:
d1232:
d1233:
d1234:
d1235:
d1236:
d1237:
d1238:
d1239:
d1239:
d1240:
d1241:
d1242:
d1243:
d1244:
d1245:
d1246:
d1247:
d1248:
d1249:
d1249:
d1250:
d1251:
d1252:
d1253:
d1254:
d1255:
d1256:
d1257:
d1258:
d1259:
d1259:
d1260:
d1261:
d1262:
d1263:
d1264:
d1265:
d1266:
d1267:
d1268:
d1269:
d1269:
d1270:
d1271:
d1272:
d1273:
d1274:
d1275:
d1276:
d1277:
d1278:
d1279:
d1279:
d1280:
d1281:
d1282:
d1283:
d1284:
d1285:
d1286:
d1287:
d1288:
d1289:
d1289:
d1290:
d1291:
d1292:
d1293:
d1294:
d1295:
d1296:
d1297:
d1298:
d1298:
d1299:
d1299:
d1300:
d1301:
d1302:
d1303:
d1304:
d1305:
d1306:
d1307:
d1308:
d1309:
d1309:
d1310:
d1311:
d1312:
d1313:
d1314:
d1315:
d1316:
d1317:
d1318:
d1319:
d1319:
d1320:
d1321:
d1322:
d1323:
d1324:
d1325:
d1326:
d1327:
d1328:
d1329:
d1329:
d1330:
d1331:
d1332:
d1333:
d1334:
d1335:
d1336:
d1337:
d1338:
d1339:
d1339:
d1340:
d1341:
d1342:
d1343:
d1344:
d1345:
d1346:
d1347:
d1348:
d1349:
d1349:
d1350:
d1351:
d1352:
d1353:
d1354:
d1355:
d1356:
d1357:
d1358:
d1359:
d1359:
d1360:
d1361:
d1362:
d1363:
d1364:
d1365:
d1366:
d1367:
d1368:
d1369:
d1369:
d1370:
d1371:
d1372:
d1373:
d1374:
d1375:
d1376:
d1377:
d1378:
d1379:
d1379:
d1380:
d1381:
d1382:
d1383:
d1384:
d1385:
d1386:
d1387:
d1388:
d1389:
d1389:
d1390:
d1391:
d1392:
d1393:
d1394:
d1395:
d1396:
d1397:
d1398:
d1398:
d1399:
d1399:
d1400:
d1401:
d1402:
d1403:
d1404:
d1405:
d1406:
d1407:
d1408:
d1409:
d1409:
d1410:
d1411:
d1412:
d1413:
d1414:
d1415:
d1416:
d1417:
d1418:
d1419:
d1419:
d1420:
d1421:
d1422:
d1423:
d1424:
d1425:
d1426:
d1427:
d1428:
d1429:
d1429:
d1430:
d1431:
d1432:
d1433:
d1434:
d1435:
d1436:
d1437:
d1438:
d1439:
d1439:
d1440:
d1441:
d1442:
d1443:
d1444:
d1445:
d1446:
d1447:
d1448:
d1449:
d1449:
d1450:
d1451:
d1452:
d1453:
d1454:
d1455:
d1456:
d1457:
d1458:
d1459:
d1459:
d1460:
d1461:
d1462:
d1463:
d1464:
d1465:
d1466:
d1467:
d1468:
d1469:
d1469:
d1470:
d1471:
d1472:
d1473:
d1474:
d1475:
d1476:
d1477:
d1478:
d1479:
d1479:
d1480:
d1481:
d1482:
d1483:
d1484:
d1485:
d1486:
d1487:
d1488:
d1489:
d1489:
d1490:
d1491:
d1492:
d1493:
d1494:
d1495:
d1496:
d1497:
d1498:
d1498:
d1499:
d1499:
d1500:
d1501:
d1502:
d1503:
d1504:
d1505:
d1506:
d1507:
d1508:
d1509:
d1509:
d1510:
d1511:
d1512:
d1513:
d1514:
d1515:
d1516:
d1517:
d1518:
d1519:
d1519:
d1520:
d1521:
d1522:
d1523:
d1524:
d1525:
d1526:
d1527:
d1528:
d1529:
d1529:
d1530:
d1531:
d1532:
d1533:
d1534:
d1535:
d1536:
d1537:
d1538:
d1539:
d1539:
d1540:
d1541:
d1542:
d1543:
d1544:
d1545:
d1546:
d1547:
d1548:
d1549:
d1549:
d1550:
d1551:
d1552:
d1553:
d1554:
d1555:
d1556:
d1557:
d1558:
d1559:
d1559:
d1560:
d1561:
d1562:
d1563:
d1564:
d1565:
d1566:
d1567:
d1568:
d1569:
d1569:
d1570:
d1571:
d1572:
d1573:
d1574:
d1575:
d1576:
d1577:
d1578:
d1579:
d1579:
d1580:
d1581:
d1582:
d1583:
d1584:
d1585:
d1586:
d1587:
d1588:
d1589:
d1589:
d1590:
d1591:
d1592:
d1593:
d1594:
d1595:
d1596:
d1597:
d1598:
d1598:
d1599:
d1599:
d1600:
d1601:
d1602:
d1603:
d1604:
d1605:
d1606:
d1607:
d1608:
d1609:
d1609:
d1610:
d1611:
d1612:
d1613:
d1614:
d1615:
d1616:
d1617:
d1618:
d1619:
d1619:
d1620:
d1621:
d1622:
d1623:
d1624:
d1625:
d1626:
d1627:
d1628:
d1629:
d1629:
d1630:
d1631:
d1632:
d1633:
d1634:
d1635:
d1636:
d1637:
d1638:
d1639:
d1639:
d1640:
d1641:
d1642:
d1643:
d1644:
d1645:
d1646:
d1647:
d1648:
d1649:
d1649:
d1650:
d1651:
d1652:
d1653:
d1654:
d1655:
d1656:
d1657:
d1658:
d1659:
d1659:
d1660:
d1661:
d1662:
d1663:
d1664:
d1665:
d1666:
d1667:
d1668:
d1669:
d1669:
d1670:
d1671:
d1672:
d1673:
d1674:
d1675:
d1676:
d1677:
d1678:
d1679:
d1679:
d1680:
d1681:
d1682:
d1683:
d1684:
d1685:
d1686:
d1687:
d1688:
d1689:
d1689:
d1690:
d1691:
d1692:
d1693:
d1694:
d1695:
d1696:
d1697:
d1698:
d1698:
d1699:
d1699:
d1700:
d1701:
d1702:
d1703:
d1704:
d1705:
d1706:
d1707:
d1708:
d1709:
d1709:
d1710:
d1711:
d1712:
d1713:
d1714:
d1715:
d1716:
d1717:
d1718:
d1719:
d1719:
d1720:
d1721:
d1722:
d1723:
d1724:
d1725:
d1726:
d1727:
d1728:
d1729:
d1729:
d1730:
d1731:
d1732:
d1733:
d1734:
d1735:
d1736:
d1737:
d1738:
d1739:
d1739:
d1740:
d1741:
d1742:
d1743:
d1744:
d1745:
d1746:
d1747:
d1748:
d1749:
d1749:
d1750:
d1751:
d1752:
d1753:
d1754:
d1755:
d1756:
d1757:
d1758:
d1759:
d1759:
d1760:
d1761:
d1762:
d1763:
d1764:
d1765:
d1766:
d1767:
d1768:
d1769:
d1769:
d1770:
d1771:
d1772:
d1773:
d1774:
d1775:
d1776:
d1777:
d1778:
d1779:
d1779:
d1780:
d1781:
d1782:
d1783:
d1784:
d1785:
d1786:
d1787:
d1788:
d1789:
d1789:
d1790:
d1791:
d1792:
d1793:
d1794:
d1795:
d1796:
d1797:
d1798:
d1798:
d1799:
d1799:
d1800:
d1801:
d1802:
d1803:
d1804:
d1805:
d1806:
d1807:
d1808:
d1809:
d1809:
d1810:
d1811:
d1812:
d1813:
d1814:
d1815:
d1816:
d1817:
d1818
```

Write a python program to check whether a given key already exists in a dictionary.

```
#exp4.py - C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp4.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to check whether a given key already exists in a dictionary.
import operator
def dict1():
n=int(input("no. of elements:"))
for i in range(n):
    k=input("enter key for first dictionary:")
    v=input("enter its value:")
    d[k]=v
key=input()
if key in d:
    print('Key is present in the dictionary')
else:
    print('Key is not present in the dictionary')

>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp4.py
no. of elements:5
enter key for first dictionary:1
enter key for first dictionary:2
enter key for first dictionary:3
enter key for first dictionary:4
enter key for first dictionary:5
enter its value:20
enter its value:30
enter its value:40
enter its value:50
enter its value:60
2
Key is present in the dictionary
>>>
```

Write a python program to generate and print a dictionary that contains a number (between 1 in n) in the form (x,x*x)..

```
#exp5.py - C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp5.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to generate and print a dictionary that contains a number (between 1 in n) in the form (x,x*x).. .
print("enter a number")
n=int(input())
d={}
for x in range(1,n+1):
    d[x]=x*x
print(d)

>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp5.py
enter a number
5
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
>>>
```

Write a python program to merge two python dictionaries.

```
#exp6.py - C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp6.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to merge two python dictionaries.
d1={1:1,2:2,3:3}
d2={}
d3=d1.copy()
d3.update(d2)
print(d3)

d={}
d1={}
d2={}
n=int(input("no. of elements"))
for i in range(n):
    k=input("enter key for dict1:")
    v=input("enter its value:")
    d1[k]=v
for i in range(n):
    k=input("enter key for dict2:")
    v=input("enter its value:")
    d2[k]=v
d3=d1.copy()
d3.update(d2)
print(d3)

>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/31-05-2024/DICTIONARY/exp6.py
{'1': 1, '2': 2, '3': 3}
no. of elements2
enter key for dict1:
enter its value:2
enter key for dict1:
enter its value:3
enter key for dict2:
enter its value:4
enter key for dict2:
enter its value:5
enter its value:6
{'1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6}
>>>
```

Write a python program to remove a key from a dictionary.

```
[*] replpy - C:\Users\savvy\OneDrive\Documents\Technical Training\31-05-2024\DICTIONARY\replpy (3.12.2)
File Edit Shell Debug Options Window Help
#Write a python program to remove a key from a dictionary.
def remove_key_from_dict():
    print("Input the number of elements:")
    n = int(input())
    dict1 = {}
    for i in range(n):
        key = input("Enter key for dict1:")
        value = input("Enter its value:")
        dict1[key] = value
    print("Key to be removed:")
    key_to_remove = input()
    del dict1[key_to_remove]
    print(dict1)

remove_key_from_dict()

# Output:
# Enter the number of elements:5
# Enter key for dict1:1
# Enter its value:1
# Enter key for dict1:2
# Enter its value:2
# Enter key for dict1:3
# Enter its value:3
# Enter key for dict1:4
# Enter its value:4
# Enter key for dict1:5
# Enter its value:5
# Enter key to be removed:3
# {1: 1, 2: 2, 4: 4, 5: 5}
```

Write a python program to map two lists into a dictionary.

```
[*] replpy - C:\Users\savvy\OneDrive\Documents\Technical Training\31-05-2024\DICTIONARY\replpy (3.12.2)
File Edit Shell Debug Options Window Help
#Write a python program to map two lists into a dictionary.
d = [(1,2,3,4,5,6,7,8)]
a = [1,2,3,4,5,6,7,8]
b = [1,2,3,4,5,6,7,8]
c = [1,2,3,4,5,6,7,8]
d = [1,2,3,4,5,6,7,8]
a = [1,2,3,4,5,6,7,8]
b = [1,2,3,4,5,6,7,8]
c = [1,2,3,4,5,6,7,8]
print("Mapped Dictionary:", a)

key=input().split()
values=input().split()
dict1={key[i]:values[i] for i in range(len(key))}

print(dict1)

# Output:
# Mapped Dictionary: [1, 2, 3, 4, 5, 6, 7, 8]
# Enter key:1
# Enter value:1
# Enter key:2
# Enter value:2
# Enter key:3
# Enter value:3
# Enter key:4
# Enter value:4
# Enter key:5
# Enter value:5
# Enter key:6
# Enter value:6
# Enter key:7
# Enter value:7
# Enter key:8
# Enter value:8
# Mapped Dictionary: {1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8'}
```

Write a python program to combine two dictionary adding values for common keys.

```
[*] replpy - C:\Users\savvy\OneDrive\Documents\Technical Training\31-05-2024\DICTIONARY\replpy (3.12.2)
File Edit Format Run Options Window Help
#Write a python program to combine two dictionary adding values for common keys.
dict1={'a':10,'b':20,'c':30}
dict2={'b':10,'c':20,'d':30}
combined_dict={}
for key in dict1:
    if key in dict2:
        combined_dict[key]=dict1[key]+dict2[key]
    else:
        combined_dict[key]=dict1[key]
for key in dict2:
    if key not in combined_dict:
        combined_dict[key]=dict2[key]
print("combined dictionary:",combined_dict)

# Output:
# combined dictionary: {'a': 10, 'b': 30, 'c': 50, 'd': 30}
```

STRINGS

- ⊕ Sequences of characters.

MODIFY A STRING:

- ⊕ Strings are immutable, hence elements of a string cannot be changed once it has been assigned.
- ⊕ Only new strings can be reassigned to same name.

DELETING A STRING:

- ⊕ We can directly delete a string using del keyword.

STRING METHODS AND FUNCTIONS:

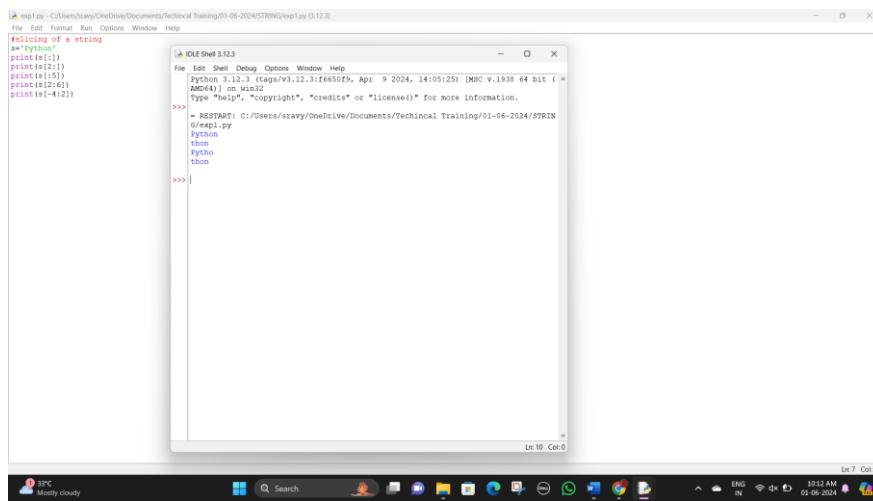
- ⊕ **String.ascii_letters**: Concatenation of the ascii_lowercase and ascii_uppercase constants. [Syntax: print(string.ascii_letters)]
- ⊕ **String.ascii_lowercase**: Concatenation of lowercase letters. [Syntax: print(string.ascii_lowercase)]
- ⊕ **String.ascii_uppercase**: Concatenation of uppercase letters. [Syntax: print(string.ascii_uppercase)]
- ⊕ **String.ascii_punctuation**: ASCII characters having punctuation character. [Syntax: print(string.ascii_punctuation)]
- ⊕ **String.digits**: Digit in strings. [Syntax: print(string.digits)]
- ⊕ **String.hexdigits**: Hexadigit in strings. [Syntax: print(string.hexdigits)]
- ⊕ **String.octdigits**: Octadigit in strings. [Syntax: print(string.octdigits)]
- ⊕ **String.endswith()**: Returns True if a string ends with the given suffix otherwise return False. [Syntax: print('hello'.endswith('o'))]
- ⊕ **String.startswith()**: Returns True if a string ends with the given prefix otherwise return False. [Syntax: print('hello'.startswith('h'))]
- ⊕ **replace()**: Returns a copy of the string where all occurrences of a substring is replaced with another substring. [Syntax: print('hello'.replace('l','t'))]
- ⊕ **String.isdigit()**: Returns True if all characters in the string are digits, Otherwise, It return False. [Syntax: print('100'.isdigit())]
- ⊕ **String.isalpha()**: Returns True if all characters in the string are alphabets, Otherwise, It return False. [Syntax: print('abc'.isalpha())]
- ⊕ **String.isdecimal()**: Returns True if all characters in the string are decimal. [Syntax: print('72'.isdecimal())]
- ⊕ **String.isalnum**: Returns true if all the characters in a given string are alphanumeric. [Syntax: print('ab12'.isalnum())]
- ⊕ **String.istitle**: Returns True if the string is a titlecased string. [Syntax: print('Hello World'.istitle())]
- ⊕ **String.upper()**: Returns the string with all uppcases. [Syntax: print('hello'.upper())]
- ⊕ **String.lower()**: Returns the string with all lowercases. [Syntax: print('hello'.lower())]

- **String.swapcase():** Returns the string with all upercases to lowercases and vice versa. [Syntax: print('hello'.swapcase())]
- **String.partition():** Splits the string at the first occurrence of the separator and returns a tuple. [Syntax: print(str.partition())]
- **String.index():** Returns the position at the first occurrence of substring in a string. [Syntax: print(str.index())]
- **String.rindex():** Returns the highest index of the substring inside the string if substring is found. [Syntax: print(str.rindex())]
- **String.splitlines():** Returns a list of lines in the string. [Syntax: print(str.splitlines())]

Dt:01-06-2024

- **String.capitalize:** Return a word with its first character capitalized.
[Syntax: print('python programming'.capitalize())]
- **String.find:** Return the lowest index in a sub string.
[Syntax: print('python cython'.find('th'))]
- **String.rfind:** Find the highest index. [Syntax: print('python cython'.rfind('th'))]
- **String.count:** Return the number of (non-overlapping) occurrences of substring sub in string. [Syntax: print('python cython'.count('th'))]
- **Len():** Returns the length of the string. [Syntax: print(len())]
- **Max():** Returns the highest alphabetical character in a string. [Syntax: print(max())]
- **Min():** Returns the minimum alphabetical character in a string. [Syntax: print(min())]

SLICING A STRING:



```

Hello! slicing of a string
s="Python"
print(s[1])
print(s[1:3])
print(s[1:5])
print(s[2:6])
print(s[-4:-2])
>>> 
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRIN
Python
ello
ello
ello
ello
ello
>>> 

```

```
# exp2.py - C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp2.py (3.12.5)
File Edit Format Run Options Window Help
#1
str1="this is 2020"
str2="2020 is now"
print(str1.isdigit())
print(str2.isdigit())
#2
str1= "sravya"
str2= "Coder"
print(str1[8] + 'Coder')
>>>

```

```
[idle] IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/G/exp2.py
False
True
sravyaCoder
>>>
```

Write a python program to calculate the length of a string.

```
# exp3.Length of string.py - C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp3.Length of string.py (3.12.3)
File Edit Format Run Options Window Help
# write a python program to calculate the length of a string.
str="code123"
print(len(str))
def len1(str1):
    count=0
    for char in str1:
        count += 1
    return count
str1=input()
print(len1(str1))
>>>

```

```
[idle] IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/G/exp3.Length of string.py
code123
8
>>>
```

Write a python script that takes input from the user and displays that input back in upper and lower cases.

```
# exp4(uppercase and lowercase).py - C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp4(uppercase and lowercase).py (3.12.3)
File Edit Format Run Options Window Help
#Write a python script that takes input from the user and displays that input back in upper and lower cases.
s1="my cat is kaw"
s2="xyz"
s1=s1.replace(s1,s1.upper())
print(s2)
>>>

```

```
[idle] IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/STRING/G/exp4(uppercase and lowercase).py
G/exp4(uppercase and lowercase).py
my cat is kaw
>>>
```

Write a python program to swap a string.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled "expSwapping the string.py" with the following Python code:

```
# swapping the string
def swap(str1):
    for i in str1:
        if i.isupper():
            res+=i.lower()
        else:
            res+=i.upper()
    return res
str=input()
print(swap(str))
```

Below it is a Python IDLE Shell window titled "IDLE Shell 3.12.3" with the following output:

```
>>> = RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp5(swapping the string).py
my name is SRAYA
>>>
```

The taskbar at the bottom shows various pinned icons and the system tray indicates the date and time as 01-06-2024 10:59 AM.

Write a python program to remove all consecutive duplicates from a given string.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled "exp!removing duplicates.py" with the following Python code:

```
# Write a python program to remove all consecutive duplicates from a given string.
str = input()
result = str[0]
for i in range(1, len(str)):
    if str[i] != result[-1]:
        result += str[i]
print(result)
```

Below it is a Python IDLE Shell window titled "IDLE Shell 3.12.3" with the following output:

```
>>> = RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp!removing duplicates.py
amulababulu
amulababulu
>>>
```

The taskbar at the bottom shows various pinned icons and the system tray indicates the date and time as 01-06-2024 11:36 AM.

Write a python program to do a word count of a paragraph.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled "exp!count of paragraph.py" with the following Python code:

```
# Write a python program to do a word count of a paragraph.
s = "I am a student of vtu"
count = 1
for char in s:
    if char == ' ':
        count += 1
print("Word count:", count)
```

Below it is a Python IDLE Shell window titled "IDLE Shell 3.12.3" with the following output:

```
>>> = RESTART: C:/Users/sravya/OneDrive/Documents/Technical Training/01-06-2024/STRING/exp!count of paragraph.py
Enter a paragraph: I am p sravya geervani from kazipet. i have completed my diploma in vnu polytechnic college.
Word count: 14
>>>
```

The taskbar at the bottom shows various pinned icons and the system tray indicates the date and time as 01-06-2024 12:01 PM.

Write a python program to move all spaces to the front of a given string in single traversal.

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled "moving spaces to front.py". It contains Python code to move all spaces in a string to the front. In the bottom-right corner, there is an IDLE Shell window titled "IDLE Shell 3.12.3". The shell window shows the execution of the script and its output, which is a string where all spaces have been moved to the beginning.

```
#!/usr/bin/moving spaces to front.py - C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/moving spaces to front.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to move all spaces to the front of a given string in single traversal.
spaces=" "
result=" "
for char in s:
    if char==" ":
        spaces+=char
    else:
        result+=char
final_string=spaces+result
print(final_string)

>>> = RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/moving spaces to front.py
move all spaces to the front of a given string in single traversal.
spacesallspacetosthedrontofagivestringiningsingletraversal.
>>>
```

Write a python program to create a string from two given strings concatenating uncommon characters of the said strings.

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled "concatenating uncommon characters.py". It contains Python code to find uncommon characters between two strings. In the bottom-right corner, there is an IDLE Shell window titled "IDLE Shell 3.12.3". The shell window shows the execution of the script and its output, which is a string containing only the uncommon characters from both input strings.

```
#!/usr/bin(concatenating uncommon characters.py - C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/concatenating uncommon characters.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to create a string from two given strings concatenating uncommon characters of the said strings.
s1 = input("Enter the first string: ")
s2 = input("Enter the second string: ")
s = ""
for char in s1:
    if char not in s2 and char not in s:
        s += char
for char in s2:
    if char not in s1 and char not in s:
        s += char
print("String with uncommon characters are : ", s)

s1=input()
s2=input()
uncommon_chars= ''.join(set(s1)^set(s2))
print(uncommon_chars)

>>> = RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/concatenating uncommon characters.py
Enter the first string: python
Enter the second string: program
String with uncommon characters are : ythargam
hello
world
e a d h r
```

Write a python program to find the maximum occurring character in a given string.

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled "maximum occurring character.py - C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/10/maximum occurring character.py (3.12.3)". It contains Python code to find the character with the highest frequency in a string. In the bottom-right corner, there is an IDLE Shell window titled "IDLE Shell 3.12.3". The shell window shows the execution of the script and its output, which is the character 'a' as it appears 10 times in the input string.

```
#!/usr/bin(maximum occurring character.py - C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/10/maximum occurring character.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to find the maximum occurring character in a given string.
s=input()
max_char=max(s, key=s.count)
print("Maximum occurring character: ", max_char)

>>> = RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/STRING/leap/10/maximum occurring character.py
aaaaaaaaaaaaaaabbcc
Maximum occurring character: a
```

FUNCTIONS

- The syntax to declare a function is: def function_name(parameters):
- Def: keyword, function_name: Function name and parameters: parameter

Def function_name(parameters):

#statement→ body of statement

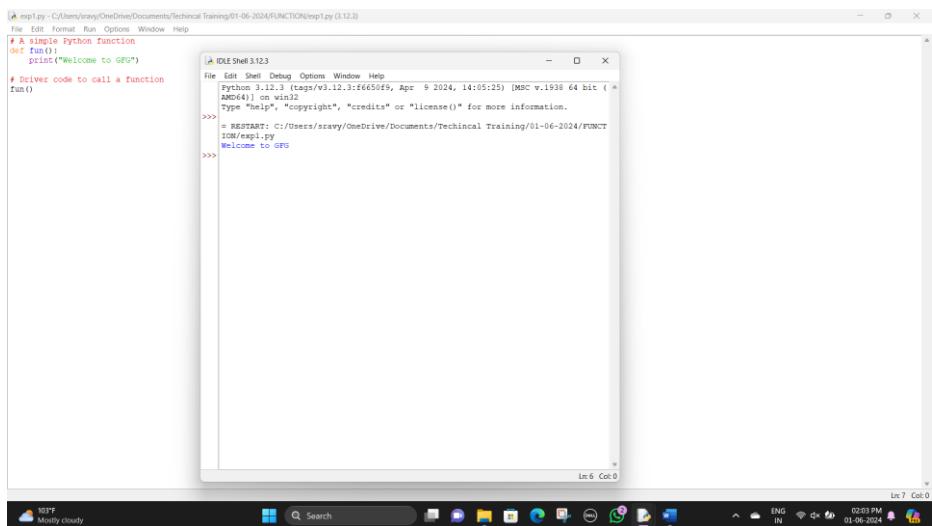
Function return<----Return expression

- **Parameters:** A parameter is the variable defined within the parameters during function definition. Simply they are written when we declare a function.

TYPES OF FUNCTIONS:

- **Built-in Function:** These are standard functions in Python that are available to use.
- **User-defined Function:** We can create our own functions based on our requirements.

CREATING AND CALL A FUNCTION:



```
# exp1.py - C:\Users\sravy\OneDrive\Documents\Technical Training\01-06-2024\FUNCTION\exp1.py (3.12.0)
File Edit Format Run Options Window Help
# A simple Python function
def fun():
    print("Welcome to GFG")
# Driver code to call a function
fun()
>>>
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:f4665d9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/FUNCT
ION/exp1.py
Welcome to GFG
>>>
```

PYTHON FUNCTION WITH PARAMETERS

Addition of two numbers

```
# Adding of two numbers
def add(num1: int, num2: int) -> int:
    """Add two numbers"""
    num = num1 + num2
    return num
# Driver Code
num1, num2 = 5, 15
ans = add(num1, num2)
print("The addition of (num1) and (num2) results (ans).")
>>>
```

```
File Edit Shell Debug Options Window Help
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTION/expl/expl/adding of two numbers.py
The addition of 5 and 15 results 20.
>>>
```

Write a function to check whether a number is prime or not.

```
# Write a function to check whether a number is prime or not.
# some more functions
def check_prime():
    num=int(input("Enter a number:"))
    if num<1:
        for i in range(2,int(num**0.5)+1):
            if num%i==0:
                print(num,"is not prime number")
                break
            else:
                print(num,"is a prime number")
        else:
            print(num,"is not a prime number")
    check_prime()
```

```
File Edit Shell Debug Options Window Help
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTION/expl/expl/prime or not.py
Enter a number:3
3 is a prime number
>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTION/expl/expl/factorial.py
Enter a number:79
79 is a prime number
>>>
```

ARGUMENTS IN FUNCTIONS

DEFAULT ARGUMENT

```
#defualt argument
def myFun(x, y=50):
    print("x: ", x)
    print("y: ", y)
myFun(10)
```

```
File Edit Shell Debug Options Window Help
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/savvy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTION/expl/expl/default argument.py
x:  10
y:  50
>>>
```

KEY ARGUMENT

```
exp5Key argument.py - C:\Users\sravya\OneDrive\Documents\Technical Training\01-06-2024\FUNCTIONS\exp5Key argument.py (3.12.3)

File Edit Format Run Options Window Help
#Key Argument
def student(firstname, lastname):
    print(firstname, lastname)
student(firstname='sravya', lastname='hi')
student(firstname='hi', lastname='sravya')

>>>
```

The screenshot shows a Windows desktop environment. At the top, there's a taskbar with icons for weather, search, and various system status indicators. Two windows are open: a file explorer window showing a folder structure and a Python IDLE Shell window. The shell window displays the execution of a script named 'exp5Key argument.py'. It defines a function 'student' that takes 'firstname' and 'lastname' as parameters. When called with keyword arguments, it prints 'sravya hi' and 'hi sravya' respectively. The Python version is 3.12.3.

POSITIONAL ARGUMENT

```
exp6Positional argument.py - C:\Users\sravya\OneDrive\Documents\Technical Training\01-06-2024\FUNCTIONS\exp6Positional argument.py (3.12.3)

File Edit Format Run Options Window Help
#POSITIONAL ARGUMENT
def nameAge(name, age):
    print("hi, i am", name)
print("case-1:", "you will get correct output bcz argument is in order")
nameAge("Sravya", 20)
print("case-2:", "you will get incorrect output bcz argument is not in order")
nameAge(20, "sravya")

>>>
```

The screenshot shows a Windows desktop environment. A file explorer window is visible in the background. In the foreground, a Python IDLE Shell window is open. It runs a script named 'exp6Positional argument.py'. The script contains a function 'nameAge' that prints a message with a name and age. It then demonstrates two cases: 'case-1' where the arguments are in the correct order (name followed by age), and 'case-2' where the arguments are in the wrong order (age followed by name). The Python version is 3.12.3.

USING THE PYTHON args VARIABLE IN FUNCTION

```
exp7args FUNCTION.py - C:\Users\sravya\OneDrive\Documents\Technical Training\01-06-2024\FUNCTIONS\exp7args FUNCTION.py (3.12.3)

File Edit Format Run Options Window Help
#args FUNCTION
def my_sum(my_integers):
    result = 0
    for x in my_integers:
        result += x
    return result

list_of_integers = [1, 2, 3]
print(my_sum(list_of_integers))

>>>
```

The screenshot shows a Windows desktop environment. A file explorer window is visible in the background. In the foreground, a Python IDLE Shell window is open. It runs a script named 'exp7args FUNCTION.py'. The script defines a function 'my_sum' that takes a list of integers and returns their sum. It then calls this function with a list containing the integers 1, 2, and 3. The Python version is 3.12.3.

USING THE PYTHON kwargs VARIABLE IN FUNCTION

The screenshot shows a Windows desktop environment. In the center, there are two windows: a code editor titled 'EXP8kwargs FUNCTION.py' and an IDLE Shell window. The code in the editor is:

```
#kwargs FUNCTION
def kwargs_FUNCTION(**kwargs):
    result = ""
    # Iterating over the Python kwargs dictionary
    for arg in kwargs.values():
        result += arg
    return result

print(concatenate(a="Real", b="Python", c="Is", d="Great", e="!"))
```

The IDLE Shell window shows the output of running the code:

```
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTIONS/EXP8kwargs FUNCTION.py
RealPythonIsGreat
>>>
```

The desktop taskbar at the bottom shows various icons for system status and applications.

RECURSION

- The process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

Factorial of a number recursively

The screenshot shows a Windows desktop environment. In the center, there are two windows: a code editor titled 'exp9Recursion FUNCTION.py' and an IDLE Shell window. The code in the editor is:

```
# exp9Recursion FUNCTION.py - C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTIONS/exp9Recursion FUNCTION.py (3.12.3)
File Edit Format Run Options Window Help
# Program to print Factorial of a number recursively.
def recursive_factorial(n):
    if n == 0:
        return n
    else:
        return n * recursive_factorial(n-1)

# user input
num = 6

# check if the input is valid or not
if num < 0:
    print("Invalid Input ! Please enter a positive number.")
elif num == 0:
    print("Factorial of number 0 is 1")
else:
    print("Factorial of number", num, "=", recursive_factorial(num))
```

The IDLE Shell window shows the output of running the code:

```
>>> = RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTIONS/exp9Recursion FUNCTION.py
Factorial of number 6 = 720
>>>
```

The desktop taskbar at the bottom shows various icons for system status and applications.

Reverse of numbers using recursion

```

# ex01/Reverse of number.py - C:/Users/sravy/OneDrive/Documents/Technical Training/01-06-2024/FUNCTIONS/exp10/reverse of number.py (3.12.3)
File Edit Shell Debug Options Window Help
#Reverse of numbers
def print_reverse(n, i=none):
    if i < 0:
        i = len(n) - 1
    if i < 0:
        return
    print(n[i])
    print_reverse(n, i - 1)

# Sample Input
n = [1, 2, 3, 4, 5]
# Call the recursive function
print_reverse(n)

```

The output shows the numbers 5, 4, 3, 2, 1 printed sequentially.

Dt:03-06-2024

Write a python program to print Fibonacci series using recursion function.

```

# ex02.py - C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/RECURSION FUNCTION/exp1.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to print Fibonacci series using recursion function.
def fibonacci_recursive(i):
    if i <= 0:
        return 0
    elif i == 1:
        return 1
    else:
        return fibonacci_recursive(i-1) + fibonacci_recursive(i-2)
n = int(input("Enter a number: "))
for i in range(n):
    print(fibonacci_recursive(i), end=" ")

```

The output shows the Fibonacci sequence from 0 to 144.

Write a python program to print n natural numbers in ascending and descending order.

```

# ex03.py - C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/RECURSION FUNCTION/exp2.py (3.12.3)
File Edit Format Run Options Window Help
#Write a python program to print n natural numbers in ascending order.
def print_numbers(n):
    if n > 0:
        print_numbers(n - 1)
        print(n)
print_numbers(5)

#Write a python program to print n natural numbers in descending order.
def print_numbers(n):
    if n > 0:
        print(n)
        print_numbers(n - 1)
print_numbers(5)

```

The output shows the natural numbers from 1 to 5 in ascending order, and from 5 down to 1 in descending order.

Tree recursion function of Fibonacci series.

```

expipy: C:\Users\sravy\OneDrive\Documents\Technical Training\03-06-2024\RECURSION FUNCTION\expipy 3.12.3
File Edit Shell Options Window Help
# Recursive function of Fibonacci series
def fibonacci(n):
    if n<1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
num_terms = int(input("Enter num of terms:"))
for i in range(num_terms):
    print(f"fibonacci{i} = {fibonacci(i)}")

```

```

File Edit Shell Options Window Help
Python 3.12.3 (tags/v3.12.3:3fe6550f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (A
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
> RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/RECUR
SION FUNCTION/expipy 3.12.3
enter num of terms:
fibonacci0 = fibonacci0
fibonacci1 = fibonacci1
fibonacci2 = fibonacci2
fibonacci3 = fibonacci3
fibonacci4 = fibonacci4

```

PYTHON AND DATA ANALYSIS

- Python is the leading language of choice for many data scientists.
- Python supports Object-Oriented Programming.

BASIC TERMINOLOGIES OF OOPS

- Class
- Objects
- Methods
- Constructors

1. CLASS:

- Class is an abstract as well user defined data type,
- The primary purpose of class is to store data and information.
- The members of the class defines the class.
- Class cannot be seen to real world but an object can be seen.
- A class is basically an object constructor or a blueprint for an object.

2. OBJECTS:

- Objects are an encapsulation of variables and functions into a single entity.
- Objects get their variables and function from classes.
- The implementation of class is an object.
- The class is not visible to the outside world but object does.

OBJECT CREATION using __init__ method

- The __init__ method is similar to constructors in C++ and Java.
- It is run as soon as an object of a class is **instantiated**.
- The method is useful to do any **initialization** you want to do with your **object**.

```

exp1.py: C:\Users\sravy\OneDrive\Documents\Technical Training\03-06-2024\OBJECT\exp1.py (3.12.3)
File Edit Format Run Options Window Help
Creating object using __init__ method
>>> class Person:
...     def __init__(self, name): #constructor
...         self.name = name
...     def say_hi(self):
...         print("Hello, my name is", self.name)
...
p=Person('ammu') #object
p.say_hi() #accessing method
>>>

```

```

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f4edf60f9, Apr  9 2024, 14:05:12) [MSC V.1930 64 bit (A
READEONLY]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/OBJEC
T/exp1.py
Hello, my name is ammu

```

OOPS CONCEPT

- Inheritance
- Polymorphism
- Data Encapsulation
- Data Abstraction
- Except Handling

INHERITANCE

TYPES OF INHERITENCE:

- Single Inheritance: When a child class inherits from one parent class, it is single inheritance.
- Multiple Inheritance: When a child class inherits from more than one parent class, it is single inheritance.
- Multilevel Inheritance: When a child class inherits from other class, it is multi-level inheritance.
- Hierachal Inheritance: Hierachal inheritance involves multiple inheritance from the same base or parent class.
- Hybrid Inheritance: Hybrid inheritance involves multiple inheritance taking place in single program.

SUPER METHOD

```

exp3.py - C:\Users\sravy\OneDrive\Documents\Technical Training\03-06-2024\OBJECT/exp3.py (3.12.3)
File Edit Format Run Options Window Help
#Single Parent
class Parent:
    def func1(self):
        print("this is function 1")
class Child(Parent):
    def func1(self):
        super().func1() #It is used to call the parent class
        print("this is function 2")
obj=Child()
obj.func2()

```

```

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/OBJEC
>>> exp3.py
>>> func1
this is function 1
>>> func2
this is function 2

```

Zoo management system using Python Inheritance

```

cop4.py - C:\Users\sravy\OneDrive\Documents\Technical Training\03-06-2024\OBJECT/cop4.py (3.12.3)
File Edit Format Run Options Window Help
#Zoo management system using Python Inheritance
class Animal:
    def __init__(self, name, species, sound):
        self.name = name
        self.species = species
        self.sound = sound
    def make_sound(self):
        print(f"The {self.species} named {self.name} goes '{self.sound}'")
class Lion(Animal):
    def __init__(self, name):
        super().__init__(name, "Lion", "roar")
    def get_info(self):
        print("Lions are the kings of the jungle.")
class Elephant(Animal):
    def __init__(self, name):
        super().__init__(name, "Elephant", "trumpet")
    def get_info(self):
        print("Elephants are the largest land animals.")
class Snake(Animal):
    def __init__(self, name):
        super().__init__(name, "Snake", "hiss")
    def get_info(self):
        print("Snakes can be found on every continent except Antarctica.")
leo = Lion("Leo")
ellie = Elephant("Ellie")
slyther = Snake("Slyther")
leo.make_sound()
leo.get_info()
print()
ellie.make_sound()
ellie.get_info()
print()
slyther.make_sound()
slyther.get_info()
print()

```

```

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/OBJEC
>>> cop4.py
The Lion named Leo goes 'roar'.
Lions are the kings of the jungle.

The Elephant named Ellie goes 'trumpet'.
Elephants are the largest land animals.

The Snake named Slyther goes 'hiss'.
Snakes can be found on every continent except Antarctica.

```

Anton likes to play chess, and so does his friend Danik.

Once they have played n games in a row. For each game it's known who was the winner — Anton or Danik. None of the games ended with a tie.

Now Anton wonders, who won more games, he or Danik? Help him determine this.

Input

The first line of the input contains a single integer n (1 ≤ n ≤ 100 000) — the number of games played.

The second line contains a string s, consisting of n uppercase English letters 'A' and 'D' — the outcome of each of the games. The i-th character of the string is equal to 'A' if the Anton won the i-th game and 'D' if Danik won the i-th game.

Output

If Anton won more games than Danik, print "Anton" (without quotes) in the only line of the output.

If Danik won more games than Anton, print "Danik" (without quotes) in the only line of the output.

If Anton and Danik won the same number of games, print "Friendship" (without quotes).

The screenshot shows a Windows desktop environment. In the foreground, there is a Python IDLE Shell window titled 'IDLE Shell 3.12.3'. It displays the following code and its execution:

```
exp5.py - C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/OBJC/CT/exp5.py (3.12.3)
File Edit Shell Debug Options Window Help
anton and danik
n = int(input())
s = input()
anton_wins = s.count('A')
danik_wins = s.count('D')
if anton_wins > danik_wins:
    print("Anton")
elif danik_wins > anton_wins:
    print("Danik")
else:
    print("Friendship")
```

The shell then prints the output:

```
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/03-06-2024/OBJC/CT/exp5.py
6
AAADDA
Anton
```

In the background, there is a code editor window titled 'exp5.py' showing the same code. The taskbar at the bottom of the screen shows various pinned icons and the date '03-06-2024'.

Dt:04-06-2024

Own code for animal using super class

The screenshot shows a Windows desktop environment. In the foreground, there is a Python IDLE Shell window titled 'IDLE Shell 3.12.3'. It displays the following code and its execution:

```
exp1.py - C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp1.py (3.12.3)
File Edit Shell Debug Options Window Help
own code for animal using super class
class Animal:
    def __init__(self,name,location,age):
        self.name=name
        self.location=location
        self.age=age
    def description(self):
        print(f"This is {self.name} and iam from {self.location} and iam {self.age}")
class Cat(Animal):
    def __init__(self,name,breed,age,location):
        super().__init__(name,age,location)
        self.breed=breed
    def description(self):
        print(f"This is {self.name} she is so cute and she is {self.age} her breed is {self.breed} and she is {self.location}")
cat=Cat(name="bisshu", age=9, breed="persian", location="Rhemaram")
animal=Animal(name="suji", location="hmk", age=10)
animal.description()
cat.description()
```

The shell then prints the output:

```
>>>
= RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp1.py
The cutie bisshu she is so cute and she is hemaram her breed is persian and s
he is 9
This is suji and iam from hmk and iam 10
```

In the background, there is a code editor window titled 'exp1.py' showing the same code. The taskbar at the bottom of the screen shows various pinned icons and the date '03-06-2024'.

Creating a class make sure to print capital "p" while defining.

```

exp2.py : C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp2.py (3.12.3)
File Edit Format Run Options Window Help
#class Person:
#    def __init__(self,name,college,group):
#        self.name=name #attributes
#        self.college=college
#        self.group=group
#    def introduce(self,lang):
#        print(f"Hi iam {self.name} and my college is {self.college} and my group is {self.group} and my mothertounge is {lang}")
#creating a object
P1=Person(name="Pogu Sravya Geervani", college="SR University", group="ECE")
P2=Person(name="Rohith", college="Vagdhevi", group="CSE")
P1.introduce("Telugu") #calling a function

```

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f66650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
> RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp2.py
P1
Hi iam Pogu Sravya Geervani and my college is SR University and my group is ECE
and my mothertounge is Telugu
>>>

```

To print our own code for introduction.

```

exp3.py : C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp3.py (3.12.3)
File Edit Format Run Options Window Help
#introduction of yourself
class Person:
    def __init__(self,name,college,group):
        self.name=name #attributes
        self.college=college
        self.group=group
    def introduce(self):
        #calling the methods
        print(f"Hi iam {self.name} and my college is {self.college} and my group is {self.group}")
    def study(self):
        print(f"{self.name} is studying in the {self.group} group at {self.college}")
    print()
#creating a object
P1=Person(name="Pogu Sravya Geervani", college="SR University", group="ECE")
P2=Person(name="Rohith", college="Vagdhevi", group="CSE")
P1.introduce() #calling functions
P2.introduce()
P2.study()

```

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f66650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
> RESTART: C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/exp3.py
P1
Hi iam Pogu Sravya Geervani and my college is SR University and my group is ECE
Pogu Sravya Geervani is studying in the ECE group at SR University
Rohith
Hi iam Rohith and my college is Vagdhevi and my group is CSE
Rohith is studying in the CSE group at Vagdhevi
>>>

```

SEARCH METHODS USING PYTHON

1. Linear Search:

- We can give the input as ascending or descending order.

```
# exp/linear search.py - C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/New folder/exp/linear search.py (3.12.3)
File Edit Format Run Options Window Help
#Linear Search
arr=input("Enter the list of numbers separated by spaces: ").split()
arr=list(map(int,arr))
target=int(input("Enter the values to search"))
found=False
for i in range(len(arr)):
    if arr[i]==target:
        print(f"Target({target}) found at index ({i}).")
        found=True
if not found:
    print(f"Target ({target}) not found in the array.")
>>>
```

2. Binary Search:

- The input should be given in ascending order then the output will be appeared otherwise it shows as not found in array.

```
# exp/binary search.py - C:/Users/sravy/OneDrive/Documents/Technical Training/04-06-2024/New folder/exp/binary search.py (3.12.3)
File Edit Format Run Options Window Help
#Binary Search
arr=[int(x) for x in input("Enter the list of numbers separated by spaces: ").split()]
target=int(input("Enter the values to search"))
start=0; end=len(arr)-1
found=False
while start <= end:
    mid=(start+end)//2
    if arr[mid]==target:
        print(f"Target({target}) found at index ({mid}).")
        found=True
    elif arr[mid] < target:
        start=mid+1
    else:
        end=mid-1
if not found:
    print(f"Target ({target}) not found in the array.")
>>>
```

Dt:05-06-2024

VECHILE DESCRIPTION

Main program:

```
Main.py - C:\Users\srujan\OneDrive\Documents\Technical Training\05-06-2024\VEHICLE>Main.py @ 12:29
File Edit View Run Tools Help
Main.py Vehicle.py AbstractVehicle.py exp1(password creation).py Main.py Password.py AbstractPassword.py <untitled> *
Assistant - 

1 from Vehicle import Bike, Car, Scooty
2
3 def main():
4     Xpulse=Bike("Black",2,5)
5     XUV700=Car("Navy Blue", 4, 6)
6     Activa=Scooty("White",2, 0)
7
8     XUV700.display_details()
9     print("\n")
10    Xpulse.display_details()
11    print("\n")
12    Activa.display_details()
13    print("\n")
14
15 if __name__ == "__main__":
16     main()

Shell -
>>> %Run -c $EDITOR_CONTENT
Name (Public): Pogu Sravya Geervani
Roll Number (Protected): 2205A41L15
Local Python 3 • Thomy's Python
```

Vehicle program:

```
Main.py - C:\Users\srujan\OneDrive\Documents\Technical Training\05-06-2024\VEHICLE\Vehicle.py @ 9:37
File Edit Run Tools Help
Main.py Vehicle.py AbstractVehicle.py exp1(password creation).py Main.py Password.py AbstractPassword.py <untitled> *
Assistant - 

1 from AbstractVehicle import AbstractVehicle
2
3 class Bike(AbstractVehicle):
4     def display_details(self):
5         print("Bike Details:")
6         print(f"Colour:{self.colour}")
7         print(f"Number of tyres:{self.num_tyre}")
8         print(f"Gears:{self.gears}")
9
10 class Car(AbstractVehicle):
11     def display_details(self):
12         print("Car Details:")
13         print(f"Colour:{self.colour}")
14         print(f"Number of tyres:{self.num_tyre}")
15         print(f"Gears:{self.gears}")
16
17 class Scooty(AbstractVehicle):
18     def display_details(self):
19         print("Scooty Details:")
20         print(f"Colour:{self.colour}")
21         print(f"Number of tyres:{self.num_tyre}")
22         print(f"Gears:{self.gears}")

Shell -
>>> %Run -c $EDITOR_CONTENT
Name (Public): Pogu Sravya Geervani
Roll Number (Protected): 2205A41L15
Name: Pogu Sravya Geervani
Local Python 3 • Thomy's Python
```

AbstractVechile:

```
Main.py - C:\Users\srujan\OneDrive\Documents\Technical Training\05-06-2024\VEHICLE\AbstractVehicle.py @ 1:1
File Edit View Run Tools Help
Main.py Vehicle.py AbstractVehicle.py exp1(password creation).py Main.py Password.py AbstractPassword.py <untitled> *
Assistant - 

1 from abc import ABC, abstractmethod
2
3 class AbstractVehicle(ABC):
4     def __init__(self, colour, num_tyre, gears):
5         self.colour=colour
6         self.num_tyre=num_tyre
7         self.gears=gears
8
9     @abstractmethod
10     def display_details(self):
11         pass

Shell -
>>> %Run -c $EDITOR_CONTENT
Name (Public): Pogu Sravya Geervani
Roll Number (Protected): 2205A41L15
Local Python 3 • Thomy's Python
```

Output:

Thonny - C:\Users\gravy\OneDrive\Documents\Technical Training\05-06-2024\VEHICLE>Main.py @ 12:29

```

1 from Vehicle import Bike, Car, Scooty
2
3 def main():
4     Xpulse= Bike("Black", 2, 5)
5     XUV700=Car("Navy Blue", 4, 6)

Shell:
Car Details:
Colour:Navy Blue
Number of tyres:4
Gears:6

Bike Details:
Colour:Black
Number of tyres:2
Gears:5

Scooty Details:
Colour:White
Number of tyres:2
Gears:0

```

Local Python 3 • Thonny's Python

SORT

BUBBLE SORT

explorable sort.py - C:\Users\gravy\OneDrive\Documents\Technical Training\05-06-2024/SORT\expl\bubble sort.py (3.12.3)

```

#BUBBLE SORTING
def bubble_sort(arr):
    n=len(arr)

    #traverse through all elements in the array
    for i in range(n):
        #if elements are already sorted, so we don't need to check them
        for j in range(0, n-i-1):
            #swap if the element found is greater than the next element
            if arr[i]>arr[j+1]:
                arr[i], arr[j+1]= arr[j+1], arr[i]

#example usage
my_list=[64, 34, 25, 12, 22, 11, 90]
print("Original list:", my_list)
bubble_sort(my_list)
print("Sorted list:", my_list)

```

IDLE Shell 3.12.3

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f1fe650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64) on win32]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
> RESTART: C:/Users/gravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl\bubble sort.py
Original List: [64, 34, 25, 12, 22, 11, 90]
Sorted List: [11, 12, 22, 25, 34, 64, 90]

```

Ln 7 Col 0 Ln 18 Col 13

SELECTION SORT

selection sort.py - C:\Users\gravy\OneDrive\Documents\Technical Training\05-06-2024/SORT\expl\selection sort.py (3.12.3)

```

#SELECTION SORT
def selection_sort(arr):
    n=len(arr)

    #traverse through all elements in the array
    for i in range(n):
        #find the minimum element in the unsorted part
        min_index=i
        for j in range(i+1,n):
            if arr[j]<arr[min_index]:
                min_index=j

        #swap the found minimum element with the first element of the unsorted part
        arr[i], arr[min_index]=arr[min_index], arr[i]

#example usage
my_list=[64, 34, 25, 12, 22, 11, 90]
print("Original list:", my_list)
selection_sort(my_list)
print("Sorted list:", my_list)

```

IDLE Shell 3.12.3

```

File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f1fe650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64) on win32]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
> RESTART: C:/Users/gravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl\selection sort.py
Original List: [64, 34, 25, 12, 22, 11, 90]
Sorted List: [11, 12, 22, 25, 34, 64, 90]

```

Ln 14 Col 33

INSERTION SORT

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled "expl3insertion sort.py - C:/Users/sravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl3insertion sort.py (3.12.3)". The code implements the insertion sort algorithm. In the top-right corner, there is an IDLE Shell window titled "IDLE Shell 3.12.3". The shell shows the execution of the script, starting with a RESTART message, followed by the original list [64, 34, 25, 12, 22, 11, 90], and the sorted list [11, 12, 22, 25, 34, 44, 60]. The desktop taskbar at the bottom shows various icons for system functions like battery status, network, and date/time.

```
#expl3insertion sort.py - C:/Users/sravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl3insertion sort.py (3.12.3)
#INSERTION SORT
def insertion_sort(arr):
    n= len(arr)
    #Traverse through all elements in the array starting from the second element
    for i in range(1,n):
        key = arr[i]
        #Move elements of arr[0..i-1] that are greater than key to one position ahead of their current position
        j=i-1
        while j>0 and key < arr[j]:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=key
#example usage:
my_list= [64, 34, 25, 12, 22, 11, 90]
print("Original List:", my_list)
insertion_sort(my_list)
print("Sorted List:", my_list)
>>>
```

MERGE SORT

The screenshot shows a Windows desktop environment. In the top-left corner, there is a code editor window titled "expl4merge sort.py - C:/Users/sravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl4merge sort.py (3.12.3)". The code implements the merge sort algorithm. In the top-right corner, there is an IDLE Shell window titled "IDLE Shell 3.12.3". The shell shows the execution of the script, starting with a RESTART message, followed by the original array [12, 11, 13, 5, 6, 7], and the sorted array [5, 6, 7, 11, 12, 13]. The desktop taskbar at the bottom shows various icons for system functions like battery status, network, and date/time.

```
#expl4merge sort.py - C:/Users/sravy/OneDrive/Documents/Technical Training/05-06-2024/SORT/expl4merge sort.py (3.12.3)
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2 # Find the middle point
        left_half = arr[:mid] # Dividing the array elements into 2 halves
        right_half = arr[mid:]

        merge_sort(left_half) # Sorting the first half
        merge_sort(right_half) # Sorting the second half

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

def print_array(arr):
    for i in arr:
        print(i, end=" ")
    print()

if __name__ == "__main__":
    arr = [12, 11, 13, 5, 6, 7]
    print("Original array:")
    print_array(arr)

    merge_sort(arr)

    print("Sorted array:")
    print_array(arr)
```

Dt:06-06-2024

METHOD OVERRIDING

The screenshot shows a Windows desktop environment. In the center, there are two windows: a code editor and an IDLE shell.

The code editor window (top left) contains the following Python script:

```
#METHOD OVERRIDING
class Animal:
    def speak(self):
        return "Animal makes a sound"

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

class Cow(Animal):
    pass

dog=Dog()
cat=Cat()
cow=Cow()

print("Dog says:", dog.speak())
print("Cat says:", cat.speak())
print("Cow says:", cow.speak())
```

The IDLE shell window (bottom right) shows the output of running the script:

```
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6f650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sraavy\OneDrive\Documents\Technical Training\06-06-2024\exp1\method overriding.py
Dog says: Woof!
Cat says: Meow!
Cow says: Animal makes a sound
```

The desktop taskbar at the bottom shows various icons for system functions like battery status, network, and date/time.