

B. K. BIRLA COLLEGE OF ARTS, SCIENCE & COMMERCE (AUTONOMOUS), KALYAN

DEPARTMENT OF INFORMATION TECHNOLOGY



Student Name:	Adarsh Gupta
Student ID:	4824559
Class:	MSC IT (CC)
Subject:	Data Science Using Python

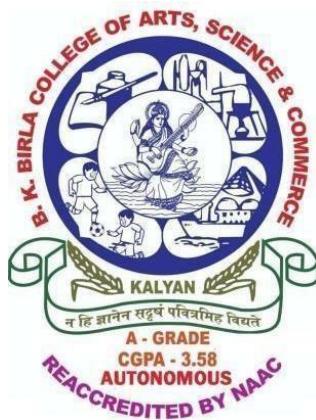
B. K. BIRLA COLLEGE OF ARTS, SCIENCE & COMMERCE

(AUTONOMOUS), KALYAN

(Affiliated to University of Mumbai)

KALYAN-MAHARASHTRA-421301

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that Mr Adarsh Gupta bearing Seat. No: (4824559), in class _____ has successfully completed practical of the subject Data Science Using Python

Teacher's Signature: _____

Place: Birla College

Date: 4th Jan 2024

College Seal

INDEX

SR. NO	PRACTICAL NAME	SIGNATURE
1.	Practical on Basics of Python (Operators)	
2.	Practical on Control and For Loop Statements	
3.	Practical on While loop and Break Statements	
4.	Practical on List, Tuples, Dictionary and Functions	
5.	Operations on Dataset	
6.	Practical on Sets	
7.	Practical on Functions	
8.	Practical on Exception Handling and file handling	
9.	Practical on Regular Expression	

Practical No. 1

Aim: Practical on basics of python

❖ Write a python program to print employee name, id, salary. Code & Output:

The screenshot shows two windows from the Python IDLE environment. The top window is a code editor with the file path 'demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)'. It contains the following code:#Prg to print employee name, id, salary.
emp_name="Smita Parale"
emp_id=1
salary=20000
print("Employee name is : ", emp_name)
print("Employee id is : ", emp_id)
print("Employee salary is : ", salary)The bottom window is an 'IDLE Shell 3.11.6' window titled 'RESTART: C:/Users/admin/Desktop/demo.py'. It displays the output of the program:=====
Employee name is : Smita Parale
Employee id is : 1
Employee salary is : 20000
>>>Ln: 160 Col: 0

❖ Write a python program for arithmetic operators and logical operators.

Arithmetic operators Code

& Output:

The screenshot shows two windows from the Python IDLE environment. The top window is a code editor with the file path 'demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)'. It contains the following code:#Prg for arithmetic operators and loical operators
#Arithmetic Operators
x=5
y=5
print ("Addition is:", x+y)
print ("Subtraction is:", x-y)
print ("Multiplication is:", x*y)
print ("Division is:", x/y)
print ("Modulus is:", x%y)
print ("Exponent is:", x**y)
print ("Floor division is:", x//y)The bottom window is an 'IDLE Shell 3.11.6' window titled 'RESTART: C:/Users/admin/Desktop/demo.py'. It displays the output of the program:=====
Addition is: 10
Subtraction is: 0
Multiplication is: 25
Division is: 1.0
Modulus is: 0
Exponent is: 3125
Floor division is: 1
>>>Ln: 169 Col: 0

#Logical operators Code

& Output:

The screenshot shows two windows from the Python IDLE environment. The top window is a code editor with the file path 'demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)'. It contains the following code:#Prg for arithmetic operators and loical operators
#Logical Operators
x=5
y=5
print(x==y and x>y)
print(x==y or x>y)
print(not(x>y and x==y))The bottom window is an 'IDLE Shell 3.11.6' window titled 'RESTART: C:/Users/admin/Desktop/demo.py'. It displays the output of the program:=====
False
True
True
>>>Ln: 174 Col: 0

❖ Write a python to solve the following equation: $(10*2+40/9-60+2)$. Code & Output:

```

demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)
File Edit Format Run Options Window Help
#solve the following (10*2+40/9-60+2)
print(10*2+40/9-60+2)

IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
>>> ===== RESTART: C:/Users/admin/Desktop/demo.py =====
-33.555555555555556
>>> | Ln: 177 Col: 0

```

- ❖ Write a python program to accept value from user like name, salary then calculate the annual income of the employee. Code & Output:

```

demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)
File Edit Format Run Options Window Help
#prg to accept value from user like name and salary then calculate annual income of the employee
emp_name=input("Enter the name:")
salary=int(input("Enter the salary:"))
print("Name of the employee is: "+emp_name)
print("Salary of the employee is: ", salary)
print("Annual income of the employee is: ", salary*12)

IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
>>> ===== RESTART: C:/Users/admin/Desktop/demo.py =====
Enter the name:Smita
Enter the salary:20000
Name of the employee is: Smita
salary of the employee is: 20000
annual income of the employee is: 240000
>>> | Ln: 206 Col: 0

```

- ❖ Write a python program to accept values from user like amount, time period and rate if interest and then calculate the simple interest.

Code & Output:

```

demo.py - C:/Users/admin/Desktop/demo.py (3.11.6)
File Edit Format Run Options Window Help
#prg to accept value from user like amount, time period and rate of interest then calculate te simple interest
A=int(input("Enter the amount:"))
T=int(input("Enter the Time period:"))
I=int(input("Enter the rate of ineterest:"))
print("Principal Amount is: ",A)
print("Time Periods is: ",T, "yrs")
print("Rate of Interest is: ",I)
print("Simple Interest= ",A*T*I/100)

IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
>>> ===== RESTART: C:/Users/admin/Desktop/demo.py =====
Enter the amount:3000
Enter the Time period:2
Enter the rate of ineterest:3
Principal Amount is: 3000
Time Perios is: 2 yrs
Rate of Interest is: 3
Simple Interest= 180.0
>>> | Ln: 233 Col: 0

```

Practical No. 2

Aim: Practical on control and for loop statements

- ❖ Find the final ticket price with the following conditions:
a) If men and senior citizen, 70% of fare is applicable

b) If female and senior citizen, 80% of fare is applicable

a) If female and normal citizen, 70% of fare is applicable

a) If men and normal citizen, 100% of fare is applicable Code

& Output:

```
[1] C:\Users\admin\Desktop\dspy\d.py (3.11.6)
File Edit Forms Run Options Window Help
gender = input("Enter Gender (M/F):")
senior_citizen = input("Are you a senior citizen? (Y/N):")
fare = float(input("Enter the fare:"))

if gender == "M" and senior_citizen == "Y":
    price= 0.7*fare

elif gender == "F" and senior_citizen == "Y":
    price= 0.8*fare

elif gender == "F" and senior_citizen == "N":
    price= 0.7*fare

elif gender == "M" and senior_citizen == "N":
    price= fare

else:
    print("Invalid input. Please enter the valid input")

print(f"Final ticket price: ${price:.2f}")

[2] IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
=====
----- RESTART: C:/Users/admin/Desktop/dspy/d.py -----
Enter Gender (M/F):F
Are you a senior citizen? (Y/N):N
Enter the fare:2000
Final ticket price: $1400.00
>>> |
```

❖ Take a dataset for diabetes, Enter the 50 patient data and use for, if else statement.
Code & Output:

P

Practical No. 3

Aim: practical on while loop and break statements

- ❖ Write a python program to find number series by adding +5. Code & Output:

The screenshot shows two windows from the Python IDLE environment. On the left is a code editor window titled 'd.py - C:\Users\admin\Desktop\d.py (3.11.6)' containing the following Python code:n=0
number = int(input("Enter the number: "))

while n < number:
 print(n+5)
 n+=5

print()On the right is an 'IDLE Shell 3.11.6' window showing the output of running the script. It displays the following text:

```
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> = RESTART: C:\Users\admin\Desktop\d.py  
Enter the number: 60  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50  
55  
60  
>>> |
```

The shell window has status bars at the bottom indicating 'Ln: 19 Col: 0' on the left and 'Ln: 1 Col: 0' on the right.

- ❖ Print a string “Hello world” except o using python while loop Code & Output:

The screenshot shows two windows from the Python IDLE environment. On the left is a code editor window titled 'd.py - C:\Users\admin\Desktop\d.py (3.11.6)' containing the following Python code:text="HELLO WORLD!"
length=len(text)
i=0
while i < length:
 if text[i]!='o':
 print(text[i], end=" ")
 i+=1
>>>On the right is an 'IDLE Shell 3.11.6' window showing the output of running the script. It displays the following text:

```
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> = RESTART: C:\Users\admin\Desktop\d.py  
H E L L   W R L D !  
>>> |
```

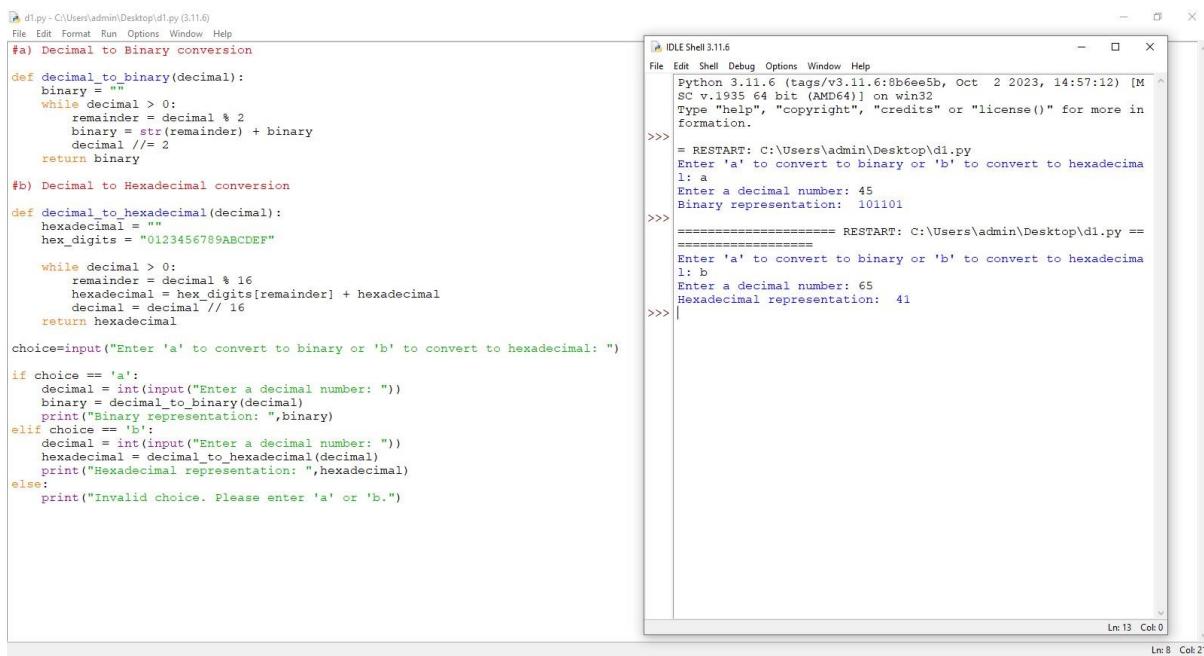
The shell window has status bars at the bottom indicating 'Ln: 6 Col: 0' on the left and 'Ln: 5 Col: 15' on the right.

❖ Convert a number

a) decimal to binary

b) decimal to hexadecimal

Code & Output:



The code in the editor (dl1.py) is as follows:

```
#a) Decimal to Binary conversion
def decimal_to_binary(decimal):
    binary = ""
    while decimal > 0:
        remainder = decimal % 2
        binary = str(remainder) + binary
        decimal //= 2
    return binary

#b) Decimal to Hexadecimal conversion
def decimal_to_hexadecimal(decimal):
    hexdecimal = ""
    hex_digits = "0123456789ABCDEF"
    while decimal > 0:
        remainder = decimal % 16
        hexdecimal = hex_digits[remainder] + hexdecimal
        decimal = decimal // 16
    return hexdecimal

choice=input("Enter 'a' to convert to binary or 'b' to convert to hexadecimal: ")

if choice == 'a':
    decimal = int(input("Enter a decimal number: "))
    binary = decimal_to_binary(decimal)
    print("Binary representation: ",binary)
elif choice == 'b':
    decimal = int(input("Enter a decimal number: "))
    hexdecimal = decimal_to_hexadecimal(decimal)
    print("Hexadecimal representation: ",hexdecimal)
else:
    print("Invalid choice. Please enter 'a' or 'b.'")
```

The output in the IDLE shell shows:

```
IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:\Users\admin\Desktop\dl1.py
Enter 'a' to convert to binary or 'b' to convert to hexadecimal:
l: a
Enter a decimal number: 45
Binary representation: 101101
>>> ===== RESTART: C:\Users\admin\Desktop\dl1.py ==
=====
Enter 'a' to convert to binary or 'b' to convert to hexadecimal:
l: b
Enter a decimal number: 65
Hexadecimal representation: 41
>>>
```

❖ Find the average of given number from user after finding average reverse

the number. Code & Output:

P

The screenshot shows two windows from the Python IDLE environment. The top window is titled 'd.py - C:\Users\admin\Desktop\d.py (3.11.6)' and contains the following code:

```
def reverse_number(num):
    reversed_num = 0
    while num > 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    return reversed_num

numbers = []

n = int(input("Enter the number of values: "))

for i in range(n):
    num = float(input(f"Enter value {i + 1}: "))
    numbers.append(num)

average = sum(numbers) / n
print("The average of the", n, "numbers is:", average)

reversed_average = reverse_number(int(average))
print(f"Reversed average: {reversed_average}")
```

The bottom window is titled 'IDLE Shell 3.11.6' and shows the execution of the script:

```
>>> ===== RESTART: C:\Users\admin\Desktop\d.py =====
Enter the number of values: 4
Enter value 1: 10
Enter value 2: 20
Enter value 3: 65
Enter value 4: 45
The average of the 4 numbers is: 35.0
Reversed average: 53
```

Practical No. 4 Aim:

practical on list, tuples, dictionary and functions

❖ Function in Python:

❖ Write a program to calculate the addition of two numbers.

Code & Output:

The screenshot shows two windows from the Python IDLE environment. The top window is titled 'list.py - C:\Users\admin\Desktop\list.py (3.11.6)' and contains the following code:

```
def addition(a,b):
    c=a+b
    return c

print("Addition = ",addition(30,40))
```

The bottom window is titled 'IDLE Shell 3.11.6' and shows the execution of the script:

```
>>> ===== RESTART: C:\Users\admin\Desktop\list.py =====
Addition = 70
```

❖ Write a program to find the annual salary of an employee. Code & Output:

The screenshot shows two windows from the Python IDLE environment. The top window is titled 'list.py - C:\Users\admin\Desktop\list.py (3.11.6)' and contains the following code:

```
def annual_salary(salary):
    annual_sal=salary*12
    return annual_sal

print("Annual Salary =", annual_salary(50000))
```

The bottom window is titled 'IDLE Shell 3.11.6' and shows the execution of the script:

```
>>> ===== RESTART: C:\Users\admin\Desktop\list.py =====
Annual Salary = 600000
```

❖ Write a program to find the rate of interest for car loan, home loan and education.

Code:

```
list.py - C:\Users\admin\Desktop\list.py (3.11.6)
File Edit Format Run Options Window Help
def loan(total_interest, principal, yrs):
    rate = total_interest*100 / principal*yrs
    return rate

print("Car Loan \nHome Loan \nEducation Loan")
choice=int(input("Enter your choice:"))

if choice==1:
    total_interest=int(input("Enter the total interest paid :"))
    principal=int(input("Enter the principal amount:"))
    yrs=int(input("Enter the number of years:"))
    print("Rate of interest of your car is:", loan(total_interest, principal, yrs), "%")

if choice==2:
    total_interest=int(input("Enter the total interest paid :"))
    principal=int(input("Enter the principal amount:"))
    yrs=int(input("Enter the number of years:"))
    print("Rate of interest of your home is:", loan(total_interest, principal, yrs), "%")

if choice==3:
    total_interest=int(input("Enter the total interest paid :"))
    principal=int(input("Enter the principal amount:"))
    yrs=int(input("Enter the number of years:"))
    print("Rate of interest of your education is:", loan(total_interest, principal, yrs), "%")

else:
    print("You have made wrong choice")
```

Output:

```
IDLE Shell 3.11.6
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:\Users\admin\Desktop\list.py
Car Loan
Home Loan
Education Loan
Enter your choice:3
Enter the total interest paid :5000
Enter the principal amount:200000
Enter the number of years:2
Rate of interest of your education is: 5.0 %
```

Practical No. 6

Aim: practical on sets

❖ Sets operation on Agriculture Dataset in Python:

a. Creating Set:

Code.

```

Code.                                     File Edit Format Run Options Window Help
[("Farm_ID", "24"), ("Crop", "Sugar Beets"), ("Acres_Planted", 75), ("Yield_Per_Acre", 40), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 14000), ("Profit", 22000)),
(("Farm_ID", "7"), ("Crop", "Apples"), ("Acres_Planted", 30), ("Yield_Per_Acre", 100), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 15000), ("Profit", 21000)),
(("Farm_ID", "19"), ("Crop", "Potatoes"), ("Acres_Planted", 50), ("Yield_Per_Acre", 80), ("Total_Production", 3600), ("Market_Price", 10), ("Revenue", 36000), ("Expense", 16000), ("Profit", 27200)),
(("Farm_ID", "11"), ("Crop", "Tomatoes"), ("Acres_Planted", 25), ("Yield_Per_Acre", 120), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 18000), ("Profit", 42000)),
(("Farm_ID", "10"), ("Crop", "Carrots"), ("Acres_Planted", 70), ("Yield_Per_Acre", 60), ("Total_Production", 4200), ("Market_Price", 9), ("Revenue", 37800), ("Expense", 15000), ("Profit", 22800)),
(("Farm_ID", "16"), ("Crop", "Onions"), ("Acres_Planted", 95), ("Yield_Per_Acre", 20), ("Total_Production", 2850), ("Market_Price", 7), ("Revenue", 19950), ("Expense", 9500), ("Profit", 10450)),
(("Farm_ID", "28"), ("Crop", "Hops"), ("Acres_Planted", 35), ("Yield_Per_Acre", 100), ("Total_Production", 3500), ("Market_Price", 40), ("Revenue", 140000), ("Expense", 35000), ("Profit", 105000)),
(("Farm_ID", "6"), ("Crop", "Potatoes"), ("Acres_Planted", 50), ("Yield_Per_Acre", 50), ("Total_Production", 2500), ("Market_Price", 10), ("Revenue", 25000), ("Expense", 12000), ("Profit", 13000)),
(("Farm_ID", "22"), ("Crop", "Broccoli"), ("Acres_Planted", 40), ("Yield_Per_Acre", 60), ("Total_Production", 2400), ("Market_Price", 14), ("Revenue", 33600), ("Expense", 15000), ("Profit", 18600)),
(("Farm_ID", "14"), ("Crop", "Sunflowers"), ("Acres_Planted", 75), ("Yield_Per_Acre", 25), ("Total_Production", 1875), ("Market_Price", 22), ("Revenue", 41250), ("Expense", 14000), ("Profit", 27250)),
(("Farm_ID", "13"), ("Crop", "Peppers"), ("Acres_Planted", 35), ("Yield_Per_Acre", 90), ("Total_Production", 3150), ("Market_Price", 14), ("Revenue", 44100), ("Expense", 17000), ("Profit", 27100)),
(("Farm_ID", "3"), ("Crop", "Rice"), ("Acres_Planted", 80), ("Yield_Per_Acre", 40), ("Total_Production", 3200), ("Market_Price", 8), ("Revenue", 25600), ("Expense", 10000), ("Profit", 15600)),
(("Farm_ID", "29"), ("Crop", "Blueberries"), ("Acres_Planted", 50), ("Yield_Per_Acre", 120), ("Total_Production", 6000), ("Market_Price", 18), ("Revenue", 108000), ("Expense", 28000), ("Profit", 80000)),
(("Farm_ID", "30"), ("Crop", "Avocado"), ("Acres_Planted", 20), ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 40), ("Revenue", 120000), ("Expense", 40000), ("Profit", 80000)),
(("Farm_ID", "19"), ("Crop", "Peanuts"), ("Acres_Planted", 65), ("Yield_Per_Acre", 35), ("Total_Production", 2275), ("Market_Price", 18), ("Revenue", 40950), ("Expense", 12000), ("Profit", 28950)),
(("Farm_ID", "9"), ("Crop", "Grapes"), ("Acres_Planted", 60), ("Yield_Per_Acre", 70), ("Total_Production", 4200), ("Market_Price", 18), ("Revenue", 75600), ("Expense", 25000), ("Profit", 50600)),
(("Farm_ID", "2"), ("Crop", "Corn"), ("Acres_Planted", 150), ("Yield_Per_Acre", 30), ("Total_Production", 3750), ("Market_Price", 4), ("Revenue", 15000), ("Expense", 9000), ("Profit", 6000)),
(("Farm_ID", "27"), ("Crop", "Tomatoes"), ("Acres_Planted", 90), ("Yield_Per_Acre", 30), ("Total_Production", 2700), ("Market_Price", 12), ("Revenue", 32400), ("Expense", 16500), ("Profit", 47500)),
(("Farm_ID", "17"), ("Crop", "Strawberries"), ("Acres_Planted", 20), ("Yield_Per_Acre", 150), ("Total_Production", 3150), ("Market_Price", 6), ("Revenue", 18900), ("Expense", 8500), ("Profit", 10400)),
(("Farm_ID", "17"), ("Crop", "Strawberries"), ("Acres_Planted", 20), ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 5), ("Revenue", 90000), ("Expense", 25000), ("Profit", 65000)),
(("Farm_ID", "21"), ("Crop", "Sorghum"), ("Acres_Planted", 85), ("Yield_Per_Acre", 25), ("Total_Production", 2125), ("Market_Price", 9), ("Revenue", 19125), ("Expense", 8000), ("Profit", 11125)),
(("Farm_ID", "4"), ("Crop", "Soybeans"), ("Acres_Planted", 120), ("Yield_Per_Acre", 20), ("Total_Production", 2400), ("Market_Price", 7), ("Revenue", 16800), ("Expense", 7500), ("Profit", 9300)),
(("Farm_ID", "1"), ("Crop", "Wheat"), ("Acres_Planted", 100), ("Yield_Per_Acre", 30), ("Total_Production", 3000), ("Market_Price", 5), ("Revenue", 15000), ("Expense", 8000), ("Profit", 7000)),
(("Farm_ID", "20"), ("Crop", "Lettuce"), ("Acres_Planted", 30), ("Yield_Per_Acre", 70), ("Total_Production", 2100), ("Market_Price", 15), ("Revenue", 31500), ("Expense", 12000), ("Profit", 19500)),
(("Farm_ID", "12"), ("Crop", "Cotton"), ("Acres_Planted", 110), ("Yield_Per_Acre", 15), ("Total_Production", 1650), ("Market_Price", 25), ("Revenue", 41250), ("Expense", 12000), ("Profit", 29250)),
(("Farm_ID", "18"), ("Crop", "Beans"), ("Acres_Planted", 55), ("Yield_Per_Acre", 40), ("Total_Production", 2200), ("Market_Price", 10), ("Revenue", 22000), ("Expense", 10000), ("Profit", 12000))]

agricultural_dataset2 = [
    ("Farm_ID", "24"), ("Crop", "Sugar Beets"), ("Acres_Planted", 75), ("Yield_Per_Acre", 40), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 14000), ("Profit", 22000)),
    ("Farm_ID", "31"), ("Crop", "Spinach"), ("Acres_Planted", 15), ("Yield_Per_Acre", 90), ("Total_Production", 1350), ("Market_Price", 18), ("Revenue", 24300), ("Expense", 10000), ("Profit", 14300)),
    ("Farm_ID", "33"), ("Crop", "Kale"), ("Acres_Planted", 35), ("Yield_Per_Acre", 80), ("Total_Production", 2800), ("Market_Price", 15), ("Revenue", 42000), ("Expense", 15000), ("Profit", 27000)),
    ("Farm_ID", "7"), ("Crop", "Apples"), ("Acres_Planted", 30), ("Yield_Per_Acre", 100), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 15000), ("Profit", 21000)),
]

print(agricultural_data.intersection(agricultural_dataset2))

```

Output:

```

= RESTART: D:/SMITA/MSci IT CC P1/DSPY/Programs/Agriculture_dataset_sets.py
((Farm_ID_, 18), ('Crop', 'Beans'), ('Acres_Planted', 55), ('Yield_Per_Acre', 40), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000), ((Farm_ID_, 27), ('Crop', 'Tobacco'), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500), ((Farm_ID_, 20), ('Crop', 'Lettuce'), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500), ((Farm_ID_, 9), ('Crop', 'Grapes'), ('Acres_Planted', 100), ('Yield_Per_Acre', 70), ('Total_Production', 4200), ('Market_Price', 18), ('Revenue', 75600), ('Expense', 25000), ('Profit', 50600)), ((Farm_ID_, 1), ('Crop', 'Wheat'), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000), ((Farm_ID_, 29), ('Crop', 'Blueberries'), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000), ((Farm_ID_, 24), ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000), ((Farm_ID_, 28), ('Crop', 'Hops'), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 140000), ('Expense', 50000), ('Profit', 90000), ((Farm_ID_, 22), ('Crop', 'Mango'), ('Acres_Planted', 25), ('Yield_Per_Acre', 80), ('Total_Production', 2000), ('Market_Price', 10), ('Revenue', 20000), ('Expense', 8000), ('Profit', 12000), ((Farm_ID_, 21), ('Crop', 'Broccoli'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 10), ('Revenue', 24000), ('Expense', 10000), ('Profit', 14000), ((Farm_ID_, 19), ('Crop', 'Papaya'), ('Acres_Planted', 30), ('Yield_Per_Acre', 90), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 31500), ('Expense', 15000), ('Profit', 16500), ((Farm_ID_, 23), ('Crop', 'Pumpkin'), ('Acres_Planted', 30), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100), ((Farm_ID_, 5), ('Crop', 'Banana'), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 15), ('Revenue', 10500), ('Expense', 8500), ('Profit', 10000), ((Farm_ID_, 17), ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 25), ('Total_Production', 2735), ('Market_Price', 4), ('Revenue', 18000), ('Expense', 12000), ('Profit', 6000), ((Farm_ID_, 19), ('Crop', 'Peanuts'), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950), ((Farm_ID_, 12), ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 30), ('Revenue', 45000), ('Expense', 25000), ('Profit', 65000)), ((Farm_ID_, 4), ('Crop', 'Soybeans'), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300), ((Farm_ID_, 21), ('Crop', 'Coffee'), ('Acres_Planted', 15), ('Yield_Per_Acre', 200), ('Total_Production', 3000), ('Market_Price', 35), ('Revenue', 105000), ('Expense', 105000), ('Profit', 95000), ('Profit', 104500), ((Farm_ID_, 7), ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000), ((Farm_ID_, 10), ('Crop', 'Carrots'), ('Acres_Planted', 7), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 9), ('Revenue', 37800), ('Expense', 15000), ('Profit', 22800), ((Farm_ID_, 23), ('Crop', 'Tomatoes'), ('Acres_Planted', 25), ('Yield_Per_Acre', 120), ('Total_Production', 3000), ('Market_Price', 20), ('Revenue', 60000), ('Expense', 18000), ('Profit', 42000), ((Farm_ID_, 3), ('Crop', 'Rice'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 15000), ('Profit', 15600), ((Farm_ID_, 21), ('Crop', 'Cucumbers'), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200), ((Farm_ID_, 15), ('Crop', 'Sunflowers'), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 2750), ((Farm_ID_, 30), ('Crop', 'Avocado'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000), ((Farm_ID_, 20), ('Crop', 'Oranges'), ('Acres_Planted', 40), ('Yield_Per_Acre', 80), ('Total_Production', 3200), ('Market_Price', 15), ('Revenue', 48000), ('Expense', 20000), ('Profit', 28000)))
>>>

```

b. Access the Dataset items:

Code:

Output..

```
IDE Shell 3.11.6
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 

== RESTART: D:/SMTC/MSI IT CC P1/DSPY/Programs/Agriculture_dataset_sets.py
([Farm_ID, 18], ('Crop', 'Beans'), ('Acres_Planted', 55), ('Yield_Per_Acre', 40), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000))
([Farm_ID, 22], ('Crop', 'Broccoli'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 14), ('Revenue', 33600), ('Expense', 15000), ('Profit', 18600))
([Farm_ID, 3], ('Crop', 'Rice'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 10000), ('Profit', 15600))
([Farm_ID, 25], ('Crop', 'Cabbage'), ('Acres_Planted', 25), ('Yield_Per_Acre', 80), ('Total_Production', 2000), ('Market_Price', 10), ('Revenue', 20000), ('Expense', 9000), ('Profit', 11000))
([Farm_ID, 2], ('Crop', 'Corn'), ('Acres_Planted', 150), ('Yield_Per_Acre', 25), ('Total_Production', 3750), ('Market_Price', 4), ('Revenue', 15000), ('Expense', 9000), ('Profit', 6000))
([Farm_ID, 12], ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 25), ('Revenue', 41250), ('Expense', 12000), ('Profit', 29250))
([Farm_ID, 4], ('Crop', 'Soybeans'), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300))
([Farm_ID, 6], ('Crop', 'Potatoes'), ('Acres_Planted', 50), ('Yield_Per_Acre', 50), ('Total_Production', 2500), ('Market_Price', 10), ('Revenue', 25000), ('Expense', 12000), ('Profit', 13000))
([Farm_ID, 21], ('Crop', 'Sorghum'), ('Acres_Planted', 85), ('Yield_Per_Acre', 25), ('Total_Production', 2125), ('Market_Price', 9), ('Revenue', 19125), ('Expense', 8000), ('Profit', 11125))
([Farm_ID, 11], ('Crop', 'Wheat'), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000))
([Farm_ID, 17], ('Crop', 'Carrots'), ('Acres_Planted', 60), ('Yield_Per_Acre', 40), ('Total_Production', 2400), ('Market_Price', 12), ('Revenue', 28800), ('Expense', 15000), ('Profit', 5000))
([Farm_ID, 3], ('Crop', 'Onions'), ('Acres_Planted', 55), ('Yield_Per_Acre', 50), ('Total_Production', 2750), ('Market_Price', 0), ('Revenue', 22000), ('Expense', 11000), ('Profit', 11000))
([Farm_ID, 19], ('Crop', 'Peanuts'), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950))
([Farm_ID, 7], ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000))
([Farm_ID, 27], ('Crop', 'Tobacco'), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500))
([Farm_ID, 28], ('Crop', 'Hops'), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 140000), ('Expense', 35000), ('Profit', 105000))
([Farm_ID, 8], ('Crop', 'Oranges'), ('Acres_Planted', 40), ('Yield_Per_Acre', 80), ('Total_Production', 3200), ('Market_Price', 15), ('Revenue', 48000), ('Expense', 20000), ('Profit', 28000))
([Farm_ID, 20], ('Crop', 'Lettuce'), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500))
([Farm_ID, 24], ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000))
([Farm_ID, 13], ('Crop', 'Peppers'), ('Acres_Planted', 35), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100))
([Farm_ID, 26], ('Crop', 'Coffee'), ('Acres_Planted', 15), ('Yield_Per_Acre', 200), ('Total_Production', 3000), ('Market_Price', 35), ('Revenue', 105000), ('Expense', 30000), ('Profit', 75000))
([Farm_ID, 5], ('Crop', 'Barley'), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 6), ('Revenue', 18900), ('Expense', 8500), ('Profit', 10400))
([Farm_ID, 29], ('Crop', 'Blueberries'), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000))
([Farm_ID, 30], ('Crop', 'Avocado'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000))
([Farm_ID, 11], ('Crop', 'Tomatoes'), ('Acres_Planted', 60), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 20), ('Revenue', 77800), ('Expense', 15000), ('Profit', 22800))
([Farm_ID, 17], ('Crop', 'Strawberries'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 30), ('Revenue', 90000), ('Expense', 25000), ('Profit', 65000))
([Farm_ID, 15], ('Crop', 'Cucumbers'), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200))
([Farm_ID, 14], ('Crop', 'Sunflowers'), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 27250))
([Farm_ID, 16], ('Crop', 'Oats'), ('Acres_Planted', 95), ('Yield_Per_Acre', 30), ('Total_Production', 2850), ('Market_Price', 7), ('Revenue', 19950), ('Expense', 9500), ('Profit', 10450))
>>>
```

c. Add items in Dataset using add():

Code:

```
Agriculture_dataset_sets.py -o /DMITA/MSK_IT_CC_Pl/DSPV/Programs/Agriculture_dataset_sets.py (3.11.6)
File Edit Formulas Run Options Window Help
agricultural_data = (
    # Farm ID 23, Crop: Onions, Acres Planted: 55, Yield Per Acre: 50, Total Production: 2750, Market Price: 8, Revenue: 22000, Expense: 11000, Profit: 11000
    # Farm ID 8, Crop: Oranges, Acres Planted: 40, Yield Per Acre: 80, Total Production: 3200, Market Price: 15, Revenue: 48000, Expense: 20000, Profit: 28000
    # Farm ID 25, Crop: Cabbage, Acres Planted: 25, Yield Per Acre: 80, Total Production: 2000, Market Price: 10, Revenue: 20000, Expense: 9000, Profit: 11000
    # Farm ID 26, Crop: Coffee, Acres Planted: 15, Yield Per Acre: 200, Total Production: 3000, Market Price: 35, Revenue: 105000, Expense: 30000, Profit: 75000
    # Farm ID 24, Crop: Sugar Beets, Acres Planted: 75, Yield Per Acre: 40, Total Production: 3000, Market Price: 12, Revenue: 36000, Expense: 14000, Profit: 22000
    # Farm ID 7, Crop: Apples, Acres Planted: 30, Yield Per Acre: 100, Total Production: 3000, Market Price: 12, Revenue: 36000, Expense: 15000, Profit: 21000
    # Farm ID 15, Crop: Cucumbers, Acres Planted: 45, Yield Per Acre: 80, Total Production: 3600, Market Price: 12, Revenue: 43200, Expense: 16000, Profit: 27200
    # Farm ID 11, Crop: Tomatoes, Acres Planted: 25, Yield Per Acre: 120, Total Production: 3000, Market Price: 20, Revenue: 60000, Expense: 18000, Profit: 42000
    # Farm ID 10, Crop: Carrots, Acres Planted: 70, Yield Per Acre: 60, Total Production: 4200, Market Price: 9, Revenue: 37800, Expense: 15000, Profit: 22800
    # Farm ID 16, Crop: Oats, Acres Planted: 95, Yield Per Acre: 30, Total Production: 2850, Market Price: 7, Revenue: 19950, Expense: 9500, Profit: 10450
    # Farm ID 28, Crop: Hops, Acres Planted: 35, Yield Per Acre: 100, Total Production: 3500, Market Price: 40, Revenue: 140000, Expense: 35000, Profit: 105000
    # Farm ID 6, Crop: Potatoes, Acres Planted: 50, Yield Per Acre: 50, Total Production: 2500, Market Price: 10, Revenue: 25000, Expense: 12000, Profit: 13000
    # Farm ID 22, Crop: Broccoli, Acres Planted: 40, Yield Per Acre: 60, Total Production: 2400, Market Price: 14, Revenue: 33600, Expense: 15000, Profit: 18600
    # Farm ID 14, Crop: Sunflowers, Acres Planted: 75, Yield Per Acre: 25, Total Production: 1875, Market Price: 22, Revenue: 41250, Expense: 14000, Profit: 27250
    # Farm ID 13, Crop: Peppers, Acres Planted: 35, Yield Per Acre: 90, Total Production: 3150, Market Price: 14, Revenue: 44100, Expense: 17000, Profit: 27100
    # Farm ID 29, Crop: Blueberries, Acres Planted: 50, Yield Per Acre: 120, Total Production: 6000, Market Price: 18, Revenue: 108000, Expense: 28000, Profit: 80000
    # Farm ID 19, Crop: Peas, Acres Planted: 80, Yield Per Acre: 40, Total Production: 3200, Market Price: 8, Revenue: 25600, Expense: 10000, Profit: 15600
    # Farm ID 9, Crop: Peaches, Acres Planted: 65, Yield Per Acre: 35, Total Production: 2275, Market Price: 18, Revenue: 40950, Expense: 12000, Profit: 28950
    # Farm ID 20, Crop: Grapes, Acres Planted: 60, Yield Per Acre: 70, Total Production: 4200, Market Price: 18, Revenue: 75600, Expense: 25000, Profit: 50600
    # Farm ID 2, Crop: Corn, Acres Planted: 150, Yield Per Acre: 25, Total Production: 3750, Market Price: 4, Revenue: 15000, Expense: 9000, Profit: 6000
    # Farm ID 27, Crop: Tobacco, Acres Planted: 90, Yield Per Acre: 30, Total Production: 3700, Market Price: 25, Revenue: 67500, Expense: 20000, Profit: 47500
    # Farm ID 5, Crop: Barley, Acres Planted: 90, Yield Per Acre: 35, Total Production: 3150, Market Price: 6, Revenue: 18900, Expense: 8500, Profit: 10400
    # Farm ID 17, Crop: Strawberries, Acres Planted: 20, Yield Per Acre: 150, Total Production: 3000, Market Price: 30, Revenue: 90000, Expense: 25000, Profit: 65000
    # Farm ID 21, Crop: Sorghum, Acres Planted: 85, Yield Per Acre: 25, Total Production: 2125, Market Price: 9, Revenue: 19125, Expense: 8000, Profit: 11125
    # Farm ID 4, Crop: Soybeans, Acres Planted: 120, Yield Per Acre: 20, Total Production: 2400, Market Price: 7, Revenue: 16800, Expense: 7500, Profit: 9300
    # Farm ID 1, Crop: Wheat, Acres Planted: 100, Yield Per Acre: 30, Total Production: 3000, Market Price: 5, Revenue: 15000, Expense: 8000, Profit: 7000
    # Farm ID 20, Crop: Lettuce, Acres Planted: 30, Yield Per Acre: 70, Total Production: 2100, Market Price: 15, Revenue: 31500, Expense: 12000, Profit: 19500
    # Farm ID 12, Crop: Cotton, Acres Planted: 110, Yield Per Acre: 15, Total Production: 1650, Market Price: 25, Revenue: 41250, Expense: 20000, Profit: 29250
    # Farm ID 18, Crop: Beans, Acres Planted: 55, Yield Per Acre: 40, Total Production: 2200, Market Price: 10, Revenue: 22000, Expense: 10000, Profit: 12000
)

agricultural_data.addd((Farm_ID, 31), (Crop, 'Peas'), (Acres_Planted, 40), (Yield_Per_Acre, 60), (Total_Production, 2400), (Market_Price, 10), (Revenue, 24000), (Expense, 9000), (Profit, 15000))

for data in agricultural_data:
    print(data)

print(len(agricultural_data))
```

Output:

```

File Edit Shell Options Window Help
Python 3.11.6 (tags/v3.11.6b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
= RESTART: D:/SMITA/MSIC It CC P1/DSPY/Programs/Agriculture_dataset_sets.py
([Farm_ID: 29], ('Crop', 'Blueberries'), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000))
([Farm_ID: 20], ('Crop', 'Lettuce'), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500))
([Farm_ID: 15], ('Crop', 'Cucumbers'), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200))
([Farm_ID: 26], ('Crop', 'Fruit'), ('Acres_Planted', 15), ('Yield_Per_Acre', 200), ('Total_Production', 3000), ('Market_Price', 35), ('Revenue', 105000), ('Expense', 30000), ('Profit', 75000))
([Farm_ID: 41], ('Crop', 'Soybeans'), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300))
([Farm_ID: 17], ('Crop', 'Strawberries'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 30), ('Revenue', 90000), ('Expense', 25000), ('Profit', 65000))
([Farm_ID: 29], ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000))
([Farm_ID: 13], ('Crop', 'Peppers'), ('Acres_Planted', 35), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100))
([Farm_ID: 25], ('Crop', 'Oats'), ('Acres_Planted', 95), ('Yield_Per_Acre', 100), ('Total_Production', 2850), ('Market_Price', 7), ('Revenue', 19550), ('Expense', 9500), ('Profit', 10450))
([Farm_ID: 25], ('Crop', 'Cabbage'), ('Acres_Planted', 25), ('Yield_Per_Acre', 80), ('Total_Production', 2000), ('Market_Price', 10), ('Revenue', 20000), ('Expense', 9000), ('Profit', 11000))
([Farm_ID: 8], ('Crop', 'Oranges'), ('Acres_Planted', 40), ('Yield_Per_Acre', 80), ('Total_Production', 3200), ('Market_Price', 15), ('Revenue', 48000), ('Expense', 20000), ('Profit', 28000))
([Farm_ID: 21], ('Crop', 'Sorghum'), ('Acres_Planted', 85), ('Yield_Per_Acre', 25), ('Total_Production', 2125), ('Market_Price', 9), ('Revenue', 19125), ('Expense', 8000), ('Profit', 11125))
([Farm_ID: 19], ('Crop', 'Peanuts'), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950))
([Farm_ID: 10], ('Crop', 'Carrots'), ('Acres_Planted', 70), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 9), ('Revenue', 37800), ('Expense', 15000), ('Profit', 22800))
([Farm_ID: 27], ('Crop', 'Tobacco'), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500))
([Farm_ID: 18], ('Crop', 'Beans'), ('Acres_Planted', 55), ('Yield_Per_Acre', 100), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000))
([Farm_ID: 22], ('Crop', 'Broccoli'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 14), ('Revenue', 33600), ('Expense', 15000), ('Profit', 18600))
([Farm_ID: 30], ('Crop', 'Avocado'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000))
([Farm_ID: 7], ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000))
([Farm_ID: 2], ('Crop', 'Corn'), ('Acres_Planted', 150), ('Yield_Per_Acre', 25), ('Total_Production', 3750), ('Market_Price', 4), ('Revenue', 15000), ('Expense', 9000), ('Profit', 6000))
([Farm_ID: 1], ('Crop', 'Wheat'), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000))
([Farm_ID: 11], ('Crop', 'Tomatoes'), ('Acres_Planted', 25), ('Yield_Per_Acre', 120), ('Total_Production', 3000), ('Market_Price', 20), ('Revenue', 60000), ('Expense', 18000), ('Profit', 42000))
([Farm_ID: 28], ('Crop', 'Hops'), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 140000), ('Expense', 35000), ('Profit', 105000))
([Farm_ID: 5], ('Crop', 'Barley'), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 6), ('Revenue', 18900), ('Expense', 8500), ('Profit', 10400))
([Farm_ID: 31], ('Crop', 'Peaches'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 10), ('Revenue', 24000), ('Expense', 9000), ('Profit', 15000))
([Farm_ID: 9], ('Crop', 'Grapes'), ('Acres_Planted', 60), ('Yield_Per_Acre', 70), ('Total_Production', 4200), ('Market_Price', 18), ('Revenue', 75600), ('Expense', 25000), ('Profit', 50600))
([Farm_ID: 6], ('Crop', 'Potatoes'), ('Acres_Planted', 50), ('Yield_Per_Acre', 30), ('Total_Production', 2500), ('Market_Price', 10), ('Revenue', 25000), ('Expense', 12000), ('Profit', 13000))
([Farm_ID: 3], ('Crop', 'Rice'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 10000), ('Profit', 15600))
([Farm_ID: 14], ('Crop', 'Sunflowers'), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 27250))
([Farm_ID: 12], ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 25), ('Revenue', 41250), ('Expense', 12000), ('Profit', 29250))
([Farm_ID: 23], ('Crop', 'Onions'), ('Acres_Planted', 55), ('Yield_Per_Acre', 50), ('Total_Production', 2750), ('Market_Price', 8), ('Revenue', 22000), ('Expense', 11000), ('Profit', 11000))
31

```


f. pop():

Agriculture_dataset_sets.py - D:\SMITA\MSIT CC P1\DSPP\Programs\Agriculture_dataset_sets.py (3.11.6)

```

agricultural_data = [
    {"Farm_ID": 23, ("Crop", "Onions"), ("Acres_Planted", 55), ("Yield_Per_Acre", 50), ("Total_Production", 2750), ("Market_Price", 8), ("Revenue", 22000), ("Expense", 11000), ("Profit", 11000)},
    {"Farm_ID": 8, ("Crop", "Oranges"), ("Acres_Planted", 40), ("Yield_Per_Acre", 80), ("Total_Production", 3200), ("Market_Price", 15), ("Revenue", 48000), ("Expense", 20000), ("Profit", 28000)},
    {"Farm_ID": 25, ("Crop", "Cabbage"), ("Acres_Planted", 25), ("Yield_Per_Acre", 80), ("Total_Production", 2000), ("Market_Price", 10), ("Revenue", 20000), ("Expense", 9000), ("Profit", 11000)},
    {"Farm_ID": 26, ("Crop", "Coffee"), ("Acres_Planted", 15), ("Yield_Per_Acre", 200), ("Total_Production", 3000), ("Market_Price", 35), ("Revenue", 105000), ("Expense", 30000), ("Profit", 75000)},
    {"Farm_ID": 24, ("Crop", "Sugar Beets"), ("Acres_Planted", 75), ("Yield_Per_Acre", 40), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 14000), ("Profit", 22000)},
    {"Farm_ID": 7, ("Crop", "Apples"), ("Acres_Planted", 30), ("Yield_Per_Acre", 100), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 15000), ("Profit", 21000)},
    {"Farm_ID": 15, ("Crop", "Cucumbers"), ("Acres_Planted", 45), ("Yield_Per_Acre", 80), ("Total_Production", 3600), ("Market_Price", 12), ("Revenue", 43200), ("Expense", 16000), ("Profit", 27200)},
    {"Farm_ID": 11, ("Crop", "Tomatoes"), ("Acres_Planted", 25), ("Yield_Per_Acre", 120), ("Total_Production", 3000), ("Market_Price", 20), ("Revenue", 60000), ("Expense", 18000), ("Profit", 42000)},
    {"Farm_ID": 10, ("Crop", "Carrots"), ("Acres_Planted", 70), ("Yield_Per_Acre", 60), ("Total_Production", 4200), ("Market_Price", 9), ("Revenue", 37800), ("Expense", 15000), ("Profit", 22800)},
    {"Farm_ID": 16, ("Crop", "Oats"), ("Acres_Planted", 95), ("Yield_Per_Acre", 30), ("Total_Production", 2850), ("Market_Price", 7), ("Revenue", 19950), ("Expense", 9500), ("Profit", 10450)},
    {"Farm_ID": 28, ("Crop", "Hops"), ("Acres_Planted", 35), ("Yield_Per_Acre", 100), ("Total_Production", 3500), ("Market_Price", 40), ("Revenue", 14000), ("Expense", 35000), ("Profit", 105000)},
    {"Farm_ID": 6, ("Crop", "Potatoes"), ("Acres_Planted", 50), ("Yield_Per_Acre", 50), ("Total_Production", 2500), ("Market_Price", 10), ("Revenue", 25000), ("Expense", 12000), ("Profit", 13000)},
    {"Farm_ID": 22, ("Crop", "Broccoli"), ("Acres_Planted", 40), ("Yield_Per_Acre", 60), ("Total_Production", 2400), ("Market_Price", 14), ("Revenue", 33600), ("Expense", 15000), ("Profit", 18600)},
    {"Farm_ID": 14, ("Crop", "Sunflowers"), ("Acres_Planted", 75), ("Yield_Per_Acre", 25), ("Total_Production", 1875), ("Market_Price", 22), ("Revenue", 41250), ("Expense", 14000), ("Profit", 27500)},
    {"Farm_ID": 13, ("Crop", "Peppers"), ("Acres_Planted", 35), ("Yield_Per_Acre", 90), ("Total_Production", 3150), ("Market_Price", 14), ("Revenue", 44100), ("Expense", 17000), ("Profit", 27100)},
    {"Farm_ID": 3, ("Crop", "Rice"), ("Acres_Planted", 80), ("Yield_Per_Acre", 40), ("Total_Production", 3200), ("Market_Price", 8), ("Revenue", 25600), ("Expense", 10000), ("Profit", 15600)},
    {"Farm_ID": 29, ("Crop", "Blueberries"), ("Acres_Planted", 50), ("Yield_Per_Acre", 120), ("Total_Production", 6000), ("Market_Price", 18), ("Revenue", 108000), ("Expense", 28000), ("Profit", 80000)},
    {"Farm_ID": 30, ("Crop", "Avocado"), ("Acres_Planted", 20), ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 40), ("Revenue", 120000), ("Expense", 40000), ("Profit", 80000)},
    {"Farm_ID": 19, ("Crop", "Peanuts"), ("Acres_Planted", 65), ("Yield_Per_Acre", 35), ("Total_Production", 2275), ("Market_Price", 18), ("Revenue", 40950), ("Expense", 12000), ("Profit", 28950)},
    {"Farm_ID": 9, ("Crop", "Grapes"), ("Acres_Planted", 60), ("Yield_Per_Acre", 70), ("Total_Production", 4200), ("Market_Price", 18), ("Revenue", 75600), ("Expense", 25000), ("Profit", 50600)},
    {"Farm_ID": 2, ("Crop", "Corn"), ("Acres_Planted", 150), ("Yield_Per_Acre", 25), ("Total_Production", 3750), ("Market_Price", 4), ("Revenue", 15000), ("Expense", 9000), ("Profit", 6000)},
    {"Farm_ID": 27, ("Crop", "Tobacco"), ("Acres_Planted", 90), ("Yield_Per_Acre", 30), ("Total_Production", 2700), ("Market_Price", 25), ("Revenue", 67500), ("Expense", 20000), ("Profit", 47500)},
    {"Farm_ID": 5, ("Crop", "Barley"), ("Acres_Planted", 90), ("Yield_Per_Acre", 35), ("Total_Production", 3150), ("Market_Price", 6), ("Revenue", 18900), ("Expense", 8500), ("Profit", 10400)},
    {"Farm_ID": 17, ("Crop", "Strawberries"), ("Acres_Planted", 20), ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 30), ("Revenue", 90000), ("Expense", 25000), ("Profit", 65000)},
    {"Farm_ID": 21, ("Crop", "Sorghum"), ("Acres_Planted", 85), ("Yield_Per_Acre", 25), ("Total_Production", 2125), ("Market_Price", 9), ("Revenue", 19125), ("Expense", 8000), ("Profit", 11125)},
    {"Farm_ID": 4, ("Crop", "Soybeans"), ("Acres_Planted", 120), ("Yield_Per_Acre", 20), ("Total_Production", 2400), ("Market_Price", 7), ("Revenue", 16800), ("Expense", 7500), ("Profit", 9300)},
    {"Farm_ID": 11, ("Crop", "Wheat"), ("Acres_Planted", 100), ("Yield_Per_Acre", 30), ("Total_Production", 3000), ("Market_Price", 5), ("Revenue", 15000), ("Expense", 8000), ("Profit", 7000)},
    {"Farm_ID": 20, ("Crop", "Lettuce"), ("Acres_Planted", 30), ("Yield_Per_Acre", 70), ("Total_Production", 2100), ("Market_Price", 15), ("Revenue", 31500), ("Expense", 12000), ("Profit", 19500)},
    {"Farm_ID": 12, ("Crop", "Cotton"), ("Acres_Planted", 110), ("Yield_Per_Acre", 15), ("Total_Production", 1650), ("Market_Price", 25), ("Revenue", 41250), ("Expense", 12000), ("Profit", 29250)},
    {"Farm_ID": 18, ("Crop", "Beans"), ("Acres_Planted", 55), ("Yield_Per_Acre", 40), ("Total_Production", 2200), ("Market_Price", 10), ("Revenue", 22000), ("Expense", 10000), ("Profit", 12000))
]
```

```
agricultural_data.pop()  
print(len(agricultural_data))
```

```
>>> Output:  
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\Agriculture_dataset_sets.py  
29
```

g. **clear():**
Code:

Agriculture_dataset_sets.py - D:\SMITA\MSc IT CC P1\DSPY\Programs\Agriculture_dataset_sets.py (3.11.6)

File	Edit	Format	Run	Options	Window	Help
agricultural_data = {						
('Farm_ID', 23), ('Crop', 'Onions'), ('Acres_Planted', 55), ('Yield_Per_Acre', 50), ('Total_Production', 2750), ('Market_Price', 8), ('Revenue', 22000), ('Expense', 11000), ('Profit', 11000),						
('Farm_ID', 8), ('Crop', 'Oranges'), ('Acres_Planted', 40), ('Yield_Per_Acre', 80), ('Total_Production', 3200), ('Market_Price', 15), ('Revenue', 48000), ('Expense', 20000), ('Profit', 28000)),						
('Farm_ID', 25), ('Crop', 'Cabbage'), ('Acres_Planted', 25), ('Yield_Per_Acre', 80), ('Total_Production', 2000), ('Market_Price', 10), ('Revenue', 20000), ('Expense', 9000), ('Profit', 11000),						
('Farm_ID', 26), ('Crop', 'Coffee'), ('Acres_Planted', 15), ('Yield_Per_Acre', 200), ('Total_Production', 3000), ('Market_Price', 35), ('Revenue', 105000), ('Expense', 30000), ('Profit', 75000)),						
('Farm_ID', 24), ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000),						
('Farm_ID', 7), ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000)),						
('Farm_ID', 15), ('Crop', 'Cucumbers'), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200)),						
('Farm_ID', 11), ('Crop', 'Tomatoes'), ('Acres_Planted', 25), ('Yield_Per_Acre', 120), ('Total_Production', 3000), ('Market_Price', 20), ('Revenue', 60000), ('Expense', 18000), ('Profit', 42000)),						
('Farm_ID', 10), ('Crop', 'Carrots'), ('Acres_Planted', 70), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 9), ('Revenue', 37800), ('Expense', 15000), ('Profit', 22800)),						
('Farm_ID', 16), ('Crop', 'Oats'), ('Acres_Planted', 95), ('Yield_Per_Acre', 30), ('Total_Production', 2850), ('Market_Price', 7), ('Revenue', 19950), ('Expense', 9500), ('Profit', 10450)),						
('Farm_ID', 28), ('Crop', 'Hops'), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 14000), ('Expense', 35000), ('Profit', 105000)),						
('Farm_ID', 6), ('Crop', 'Potatoes'), ('Acres_Planted', 50), ('Yield_Per_Acre', 50), ('Total_Production', 2500), ('Market_Price', 10), ('Revenue', 25000), ('Expense', 12000), ('Profit', 13000)),						
('Farm_ID', 22), ('Crop', 'Broccoli'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 14), ('Revenue', 33600), ('Expense', 15000), ('Profit', 18600)),						
('Farm_ID', 14), ('Crop', 'Sunflowers'), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 27250)),						
('Farm_ID', 13), ('Crop', 'Peppers'), ('Acres_Planted', 35), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100)),						
('Farm_ID', 3), ('Crop', 'Rice'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 10000), ('Profit', 15600)),						
('Farm_ID', 29), ('Crop', 'Blueberries'), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000)),						
('Farm_ID', 30), ('Crop', 'Avocado'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000)),						
('Farm_ID', 19), ('Crop', 'Peanuts'), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950)),						
('Farm_ID', 9), ('Crop', 'Grapes'), ('Acres_Planted', 60), ('Yield_Per_Acre', 70), ('Total_Production', 4200), ('Market_Price', 7), ('Revenue', 75600), ('Expense', 25000), ('Profit', 50600)),						
('Farm_ID', 2), ('Crop', 'Corn'), ('Acres_Planted', 150), ('Yield_Per_Acre', 25), ('Total_Production', 3750), ('Market_Price', 4), ('Revenue', 15000), ('Expense', 9000), ('Profit', 6000)),						
('Farm_ID', 27), ('Crop', 'Tobacco'), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500)),						
('Farm_ID', 5), ('Crop', 'Barley'), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 6), ('Revenue', 18900), ('Expense', 8500), ('Profit', 10400)),						
('Farm_ID', 17), ('Crop', 'Strawberries'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 30), ('Revenue', 90000), ('Expense', 25000), ('Profit', 65000)),						
('Farm_ID', 21), ('Crop', 'Sorghum'), ('Acres_Planted', 85), ('Yield_Per_Acre', 25), ('Total_Production', 2125), ('Market_Price', 9), ('Revenue', 19125), ('Expense', 8000), ('Profit', 11125)),						
('Farm_ID', 4), ('Crop', 'Soybeans'), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300)),						
('Farm_ID', 11), ('Crop', 'Wheat'), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000)),						
('Farm_ID', 20), ('Crop', 'Lettuce'), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500)),						
('Farm_ID', 12), ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 25), ('Revenue', 41250), ('Expense', 12000), ('Profit', 29250)),						
('Farm_ID', 18), ('Crop', 'Beans'), ('Acres_Planted', 55), ('Yield_Per_Acre', 40), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000))						

```
agricultural_data.clear()  
print(len(agricultural_data))
```

Output:

```
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (A
2
Type "help", "copyright", "credits" or "license()" for more information.
> > =
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\Agriculture_dataset_sets.py
0
```

h. union():

Code:

```

Agriculture_dataset_sets.py - D:/SMITA/MSc IT CC P1/DSPY/Programs/Agriculture_dataset_sets.py (3.11.6)
File Edit Formatted Run Options Window Help
agricultural_data = [
    {"Crop": "Onions", "Acres_Planted": 55, "Yield_Per_Acre": 50, "Total_Production": 2750, "Market_Price": 8, "Revenue": 22000, "Expense": 11000, "Profit": 11000, ("Crop_ID": 23), ("Crop": "Oranges"), "Acres_Planted": 40, "Yield_Per_Acre": 80, "Total_Production": 3200, "Market_Price": 15, "Revenue": 48000, "Expense": 20000, ("Profit": 28000), ("Crop_ID": 25), ("Crop": "Cabbage"), "Acres_Planted": 25, "Yield_Per_Acre": 80, "Total_Production": 2000, "Market_Price": 10, "Revenue": 20000, "Expense": 9000, ("Profit": 11000), ("Crop_ID": 26), ("Crop": "Coffee"), "Acres_Planted": 15, "Yield_Per_Acre": 200, "Total_Production": 3000, "Market_Price": 35, "Revenue": 105000, "Expense": 30000, ("Profit": 75000), ("Crop_ID": 24), ("Crop": "Sugar Beets"), "Acres_Planted": 75, "Yield_Per_Acre": 40, "Total_Production": 3000, "Market_Price": 12, "Revenue": 36000, "Expense": 14000, ("Profit": 22000), ("Crop_ID": 7), ("Crop": "Apples"), "Acres_Planted": 30, "Yield_Per_Acre": 100, "Total_Production": 3000, "Market_Price": 12, "Revenue": 36000, "Expense": 15000, ("Profit": 21000), ("Crop_ID": 15), ("Crop": "Cucumbers"), "Acres_Planted": 45, "Yield_Per_Acre": 80, "Total_Production": 3600, "Market_Price": 12, "Revenue": 43200, "Expense": 18000, ("Profit": 27200), ("Crop_ID": 17), ("Crop": "Pumpkins"), "Acres_Planted": 30, "Yield_Per_Acre": 50, "Total_Production": 1500, "Market_Price": 10, "Revenue": 15000, "Expense": 12000, ("Profit": 3000), ("Crop_ID": 10), ("Crop": "Carrots"), "Acres_Planted": 70, "Yield_Per_Acre": 60, "Total_Production": 4200, "Market_Price": 8, "Revenue": 37800, "Expense": 18000, ("Profit": 22800), ("Crop_ID": 16), ("Crop": "Oats"), "Acres_Planted": 65, "Yield_Per_Acre": 30, "Total_Production": 2850, "Market_Price": 7, "Revenue": 19580, "Expense": 9500, ("Profit": 10450), ("Crop_ID": 20), ("Crop": "Hops"), "Acres_Planted": 35, "Yield_Per_Acre": 100, "Total_Production": 3500, "Market_Price": 40, "Revenue": 140000, "Expense": 35000, ("Profit": 105000), ("Crop_ID": 6), ("Crop": "Potatoes"), "Acres_Planted": 50, "Yield_Per_Acre": 50, "Total_Production": 2500, "Market_Price": 10, "Revenue": 25000, ("Expense": 12000, ("Profit": 13000)), ("Crop_ID": 22), ("Crop": "Broccoli"), "Acres_Planted": 40, "Yield_Per_Acre": 60, "Total_Production": 2400, "Market_Price": 14, "Revenue": 33600, ("Expense": 15000, ("Profit": 18600)), ("Crop_ID": 14), ("Crop": "Sunflowers"), "Acres_Planted": 75, "Yield_Per_Acre": 25, "Total_Production": 1875, "Market_Price": 22, "Revenue": 41250, "Expense": 14000, ("Profit": 27500), ("Crop_ID": 19), ("Crop": "Lettuce"), "Acres_Planted": 35, "Yield_Per_Acre": 90, "Total_Production": 3150, "Market_Price": 14, "Revenue": 44100, "Expense": 17000, ("Profit": 27100), ("Crop_ID": 21), ("Crop": "Rice"), "Acres_Planted": 80, "Yield_Per_Acre": 40, "Total_Production": 3200, "Market_Price": 8, "Revenue": 25600, "Expense": 10000, ("Profit": 15600), ("Crop_ID": 25), ("Crop": "Blueberries"), "Acres_Planted": 50, "Yield_Per_Acre": 120, "Total_Production": 6000, "Market_Price": 15, "Revenue": 108000, "Expense": 28000, ("Profit": 80000), ("Crop_ID": 30), ("Crop": "Avocado"), "Acres_Planted": 20, "Yield_Per_Acre": 150, "Total_Production": 3000, "Market_Price": 40, "Revenue": 120000, "Expense": 40000, ("Profit": 80000), ("Crop_ID": 18), ("Crop": "Peanuts"), "Acres_Planted": 65, "Yield_Per_Acre": 35, "Total_Production": 2275, "Market_Price": 18, "Revenue": 40950, "Expense": 12000, ("Profit": 28950), ("Crop_ID": 9), ("Crop": "Grapes"), "Acres_Planted": 60, "Yield_Per_Acre": 70, "Total_Production": 4200, "Market_Price": 18, "Revenue": 75600, "Expense": 25000, ("Profit": 50600), ("Crop_ID": 2), ("Crop": "Corn"), "Acres_Planted": 150, "Yield_Per_Acre": 25, "Total_Production": 3750, "Market_Price": 4, "Revenue": 15000, "Expense": 9000, ("Profit": 6000), ("Crop_ID": 12), ("Crop": "Barley"), "Acres_Planted": 100, "Yield_Per_Acre": 50, "Total_Production": 5000, "Market_Price": 6, "Revenue": 30000, "Expense": 18000, ("Profit": 12000), ("Crop_ID": 17), ("Crop": "Strawberries"), "Acres_Planted": 20, "Yield_Per_Acre": 150, "Total_Production": 3000, "Market_Price": 30, "Revenue": 90000, "Expense": 25000, ("Profit": 6500), ("Crop_ID": 21), ("Crop": "Sorghum"), "Acres_Planted": 85, "Yield_Per_Acre": 90, "Total_Production": 2125, "Market_Price": 9, "Revenue": 19125, "Expense": 8000, ("Profit": 11125), ("Crop_ID": 4), ("Crop": "Soybeans"), "Acres_Planted": 120, "Yield_Per_Acre": 20, "Total_Production": 2400, "Market_Price": 7, "Revenue": 16800, "Expense": 7500, ("Profit": 9300), ("Crop_ID": 1), ("Crop": "Wheat"), "Acres_Planted": 100, "Yield_Per_Acre": 30, "Total_Production": 3000, "Market_Price": 5, "Revenue": 15000, "Expense": 8000, ("Profit": 7000), ("Crop_ID": 20), ("Crop": "Lettuce"), "Acres_Planted": 30, "Yield_Per_Acre": 70, "Total_Production": 2100, "Market_Price": 15, "Revenue": 31500, "Expense": 12000, ("Profit": 19500), ("Crop_ID": 12), ("Crop": "Cotton"), "Acres_Planted": 110, "Yield_Per_Acre": 15, "Total_Production": 1650, "Market_Price": 25, "Revenue": 41250, "Expense": 12000, ("Profit": 29250), ("Crop_ID": 18), ("Crop": "Beans"), "Acres_Planted": 55, "Yield_Per_Acre": 40, "Total_Production": 2200, "Market_Price": 10, "Revenue": 22000, "Expense": 10000, ("Profit": 12000))
]

agricultural_dataset2 = [
    {"Crop": "Asparagus", "Acres_Planted": 30, "Yield_Per_Acre": 50, "Total_Production": 1500, "Market_Price": 25, "Revenue": 37500, "Expense": 15000, ("Profit": 22500), ("Crop_ID": 36), ("Crop": "Spinach"), "Acres_Planted": 15, "Yield_Per_Acre": 90, "Total_Production": 1350, "Market_Price": 18, "Revenue": 24300, "Expense": 10000, ("Profit": 14300), ("Crop_ID": 33), ("Crop": "Kale"), "Acres_Planted": 35, "Yield_Per_Acre": 80, "Total_Production": 2800, "Market_Price": 15, "Revenue": 42000, "Expense": 15000, ("Profit": 27000), ("Crop_ID": 38), ("Crop": "Arugula"), "Acres_Planted": 20, "Yield_Per_Acre": 50, "Total_Production": 2400, "Market_Price": 25, "Revenue": 36000, "Expense": 18000, ("Profit": 18000), ("Crop_ID": 39), ("Crop": "Arichokes"), "Acres_Planted": 25, "Yield_Per_Acre": 70, "Total_Production": 1750, "Market_Price": 20, "Revenue": 38500, "Expense": 15000, ("Profit": 23500), ("Crop_ID": 40), ("Crop": "Raspberries"), "Acres_Planted": 55, "Yield_Per_Acre": 100, "Total_Production": 5500, "Market_Price": 22, "Revenue": 121000, "Expense": 25000, ("Profit": 9600), ("Crop_ID": 39), ("Crop": "KiwiFruit"), "Acres_Planted": 45, "Yield_Per_Acre": 75, "Total_Production": 3375, "Market_Price": 15, "Revenue": 50625, "Expense": 20000, ("Profit": 30625), ("Crop_ID": 37), ("Crop": "Mangos"), "Acres_Planted": 80, "Yield_Per_Acre": 40, "Total_Production": 3200, "Market_Price": 30, "Revenue": 96000, "Expense": 30000, ("Profit": 66000), ("Crop_ID": 32), ("Crop": "Chickpeas"), "Acres_Planted": 50, "Yield_Per_Acre": 25, "Total_Production": 1250, "Market_Price": 12, "Revenue": 15000, "Expense": 8000, ("Profit": 7000), ("Crop_ID": 34), ("Crop": "Pears"), "Acres_Planted": 40, "Yield_Per_Acre": 60, "Total_Production": 2400, "Market_Price": 10, "Revenue": 24000, "Expense": 9000, ("Profit": 15000))
]

agricultural_dataset3 = agricultural_data.union(agricultural_dataset2)

for data in agricultural_dataset3:
    print(data)

```

Output:

IDE Shell 3.11.6

File Edit Shell Debug Options Window Help

Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32

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

>>> # RESTART: D:\SMITA\MSnC IT CC PI\DSPI\Programs\Agriculture_dataset_sets.py

(('Farm_ID', 7), ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000))

(('Farm_ID', 12), ('Crop', 'Cotton'), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 25), ('Revenue', 41250), ('Expense', 12000), ('Profit', 28250))

(('Farm_ID', 39), ('Crop', 'Kiwifruit'), ('Acres_Planted', 45), ('Yield_Per_Acre', 75), ('Total_Production', 3375), ('Market_Price', 15), ('Revenue', 50625), ('Expense', 20000), ('Profit', 30625))

(('Farm_ID', 14), ('Crop', 'Sunflowers'), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 27250))

(('Farm_ID', 10), ('Crop', 'Carrots'), ('Acres_Planted', 70), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 9), ('Revenue', 37800), ('Expense', 15000), ('Profit', 22800))

(('Farm_ID', 32), ('Crop', 'Chickpeas'), ('Acres_Planted', 50), ('Yield_Per_Acre', 25), ('Total_Production', 1250), ('Market_Price', 12), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000))

(('Farm_ID', 20), ('Crop', 'Lettuce'), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500))

(('Farm_ID', 17), ('Crop', 'Strawberries'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 30), ('Revenue', 90000), ('Expense', 25000), ('Profit', 65000))

(('Farm_ID', 1), ('Crop', 'Wheat'), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000))

(('Farm_ID', 35), ('Crop', 'Artichokes'), ('Acres_Planted', 25), ('Yield_Per_Acre', 70), ('Total_Production', 1750), ('Market_Price', 20), ('Revenue', 35000), ('Expense', 12000), ('Profit', 23000))

(('Farm_ID', 6), ('Crop', 'Potatoes'), ('Acres_Planted', 50), ('Yield_Per_Acre', 50), ('Total_Production', 2500), ('Market_Price', 10), ('Revenue', 25000), ('Expense', 12000), ('Profit', 13000))

(('Farm_ID', 23), ('Crop', 'Onions'), ('Acres_Planted', 55), ('Yield_Per_Acre', 50), ('Total_Production', 2750), ('Market_Price', 8), ('Revenue', 22000), ('Expense', 11000), ('Profit', 11000))

(('Farm_ID', 33), ('Crop', 'Arugula'), ('Acres_Planted', 35), ('Yield_Per_Acre', 80), ('Total_Production', 2800), ('Market_Price', 15), ('Revenue', 42000), ('Expense', 15000), ('Profit', 27000))

(('Farm_ID', 24), ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000))

(('Farm_ID', 15), ('Crop', 'Cucumbers'), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200))

(('Farm_ID', 36), ('Crop', 'Asparagus'), ('Acres_Planted', 30), ('Yield_Per_Acre', 50), ('Total_Production', 1500), ('Market_Price', 25), ('Revenue', 37500), ('Expense', 15000), ('Profit', 22500))

(('Farm_ID', 40), ('Crop', 'Raspberries'), ('Acres_Planted', 55), ('Yield_Per_Acre', 100), ('Total_Production', 5500), ('Market_Price', 22), ('Revenue', 121000), ('Expense', 25000), ('Profit', 96000))

(('Farm_ID', 5), ('Crop', 'Barley'), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 6), ('Revenue', 18900), ('Expense', 8800), ('Profit', 10400))

(('Farm_ID', 11), ('Crop', 'Tomatoes'), ('Acres_Planted', 25), ('Yield_Per_Acre', 120), ('Total_Production', 3000), ('Market_Price', 20), ('Revenue', 60000), ('Expense', 15000), ('Profit', 42000))

(('Farm_ID', 13), ('Crop', 'Peppers'), ('Acres_Planted', 35), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100))

(('Farm_ID', 37), ('Crop', 'Mangos'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 30), ('Revenue', 96000), ('Expense', 30000), ('Profit', 66000))

(('Farm_ID', 19), ('Crop', 'Peanuts'), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950))

(('Farm_ID', 29), ('Crop', 'Blueberries'), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000))

(('Farm_ID', 34), ('Crop', 'Pears'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 10), ('Revenue', 24000), ('Expense', 9000), ('Profit', 15000))

(('Farm_ID', 25), ('Crop', 'Cabbage'), ('Acres_Planted', 25), ('Yield_Per_Acre', 80), ('Total_Production', 2000), ('Market_Price', 10), ('Revenue', 20000), ('Expense', 9000), ('Profit', 11000))

((Farm_ID, 2), ('Crop', 'Corn'), ('Acres_Planted', 100), ('Yield_Per_Acre', 35), ('Total_Production', 3500), ('Market_Price', 4), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000))

((Farm_ID, 1), ('Crop', 'Rice'), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 10000), ('Profit', 15600))

((Farm_ID, 8), ('Crop', 'Oranges'), ('Acres_Planted', 40), ('Yield_Per_Acre', 80), ('Total_Production', 3200), ('Market_Price', 15), ('Revenue', 48000), ('Expense', 20000), ('Profit', 28000))

((Farm_ID, 22), ('Crop', 'Broccoli'), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 14), ('Revenue', 33600), ('Expense', 15000), ('Profit', 18600))

((Farm_ID, 9), ('Crop', 'Grapes'), ('Acres_Planted', 60), ('Yield_Per_Acre', 70), ('Total_Production', 4200), ('Market_Price', 18), ('Revenue', 75600), ('Expense', 25000), ('Profit', 50600))

((Farm_ID, 31), ('Crop', 'Spinach'), ('Acres_Planted', 15), ('Yield_Per_Acre', 80), ('Total_Production', 1350), ('Market_Price', 18), ('Revenue', 24300), ('Expense', 10000), ('Profit', 14300))

((Farm_ID, 38), ('Crop', 'Flours'), ('Acres_Planted', 20), ('Yield_Per_Acre', 120), ('Total_Production', 2400), ('Market_Price', 25), ('Revenue', 60000), ('Expense', 18000), ('Profit', 42000))

((Farm_ID, 28), ('Crop', 'Hops'), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 140000), ('Expense', 35000), ('Profit', 105000))

((Farm_ID, 16), ('Crop', 'Oats'), ('Acres_Planted', 95), ('Yield_Per_Acre', 30), ('Total_Production', 2850), ('Market_Price', 7), ('Revenue', 19850), ('Expense', 9500), ('Profit', 10450))

((Farm_ID, 30), ('Crop', 'Avocado'), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000))

((Farm_ID, 27), ('Crop', 'Tobacco'), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500))

((Farm_ID, 26), ('Crop', 'Coffee'), ('Acres_Planted', 15), ('Yield_Per_Acre', 200), ('Total_Production', 3000), ('Market_Price', 35), ('Revenue', 105000), ('Expense', 30000), ('Profit', 75000))

((Farm_ID, 18), ('Crop', 'Beans'), ('Acres_Planted', 55), ('Yield_Per_Acre', 40), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000))

((Farm_ID, 4), ('Crop', 'Soybeans'), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300))

((Farm_ID, 21), ('Crop', 'Sorghum'), ('Acres_Planted', 85), ('Yield_Per_Acre', 25), ('Total_Production', 2125), ('Market_Price', 9), ('Revenue', 19125), ('Expense', 8000), ('Profit', 11125))

i. intersection():

Code:

```

Agriculture_dataset_sets.py - D:\SMITA\MSc IT CC P1\DSPY\Programs\Agriculture_dataset_sets.py (3:11.6)
File Edit Format Run Options Window Help

(("Farm_ID", 24), ("Crop", "Sugar Beets"), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000)),
(("Farm_ID", 7), ("Crop", "Apples"), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000)),
(("Farm_ID", 15), ("Crop", "Cucumbers"), ('Acres_Planted', 45), ('Yield_Per_Acre', 80), ('Total_Production', 3600), ('Market_Price', 12), ('Revenue', 43200), ('Expense', 16000), ('Profit', 27200)),
(("Farm_ID", 11), ("Crop", "Tomatoes"), ('Acres_Planted', 25), ('Yield_Per_Acre', 120), ('Total_Production', 3000), ('Market_Price', 20), ('Revenue', 60000), ('Expense', 18000), ('Profit', 42000)),
(("Farm_ID", 10), ("Crop", "Carrots"), ('Acres_Planted', 70), ('Yield_Per_Acre', 60), ('Total_Production', 4200), ('Market_Price', 9), ('Revenue', 37800), ('Expense', 15000), ('Profit', 22800)),
(("Farm_ID", 16), ("Crop", "Oats"), ('Acres_Planted', 95), ('Yield_Per_Acre', 30), ('Total_Production', 2850), ('Market_Price', 7), ('Revenue', 19950), ('Expense', 9500), ('Profit', 10450)),
(("Farm_ID", 28), ("Crop", "Hops"), ('Acres_Planted', 35), ('Yield_Per_Acre', 100), ('Total_Production', 3500), ('Market_Price', 40), ('Revenue', 140000), ('Expense', 35000), ('Profit', 105000)),
(("Farm_ID", 6), ("Crop", "Potatoes"), ('Acres_Planted', 50), ('Yield_Per_Acre', 50), ('Total_Production', 2500), ('Market_Price', 10), ('Revenue', 25000), ('Expense', 12000), ('Profit', 13000)),
(("Farm_ID", 22), ("Crop", "Broccoli"), ('Acres_Planted', 40), ('Yield_Per_Acre', 60), ('Total_Production', 2400), ('Market_Price', 14), ('Revenue', 33600), ('Expense', 15000), ('Profit', 18600)),
(("Farm_ID", 14), ("Crop", "Sunflowers"), ('Acres_Planted', 75), ('Yield_Per_Acre', 25), ('Total_Production', 1875), ('Market_Price', 22), ('Revenue', 41250), ('Expense', 14000), ('Profit', 27250)),
(("Farm_ID", 13), ("Crop", "Peppers"), ('Acres_Planted', 35), ('Yield_Per_Acre', 90), ('Total_Production', 3150), ('Market_Price', 14), ('Revenue', 44100), ('Expense', 17000), ('Profit', 27100)),
(("Farm_ID", 3), ("Crop", "Onions"), ('Acres_Planted', 80), ('Yield_Per_Acre', 40), ('Total_Production', 3200), ('Market_Price', 8), ('Revenue', 25600), ('Expense', 10000), ('Profit', 15600)),
(("Farm_ID", 29), ("Crop", "Blueberries"), ('Acres_Planted', 50), ('Yield_Per_Acre', 120), ('Total_Production', 6000), ('Market_Price', 18), ('Revenue', 108000), ('Expense', 28000), ('Profit', 80000)),
(("Farm_ID", 30), ("Crop", "Avocado"), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 40), ('Revenue', 120000), ('Expense', 40000), ('Profit', 80000)),
(("Farm_ID", 19), ("Crop", "Peanuts"), ('Acres_Planted', 65), ('Yield_Per_Acre', 35), ('Total_Production', 2275), ('Market_Price', 18), ('Revenue', 40950), ('Expense', 12000), ('Profit', 28950)),
(("Farm_ID", 9), ("Crop", "Grapes"), ('Acres_Planted', 60), ('Yield_Per_Acre', 70), ('Total_Production', 4200), ('Market_Price', 18), ('Revenue', 75600), ('Expense', 25000), ('Profit', 50600)),
(("Farm_ID", 2), ("Crop", "Corn"), ('Acres_Planted', 150), ('Yield_Per_Acre', 25), ('Total_Production', 3750), ('Market_Price', 4), ('Revenue', 15000), ('Expense', 9000), ('Profit', 6000)),
(("Farm_ID", 27), ("Crop", "Tobacco"), ('Acres_Planted', 90), ('Yield_Per_Acre', 30), ('Total_Production', 2700), ('Market_Price', 25), ('Revenue', 67500), ('Expense', 20000), ('Profit', 47500)),
(("Farm_ID", 5), ("Crop", "Barley"), ('Acres_Planted', 90), ('Yield_Per_Acre', 35), ('Total_Production', 3150), ('Market_Price', 6), ('Revenue', 18900), ('Expense', 8500), ('Profit', 10400)),
(("Farm_ID", 17), ("Crop", "Strawberries"), ('Acres_Planted', 20), ('Yield_Per_Acre', 150), ('Total_Production', 3000), ('Market_Price', 30), ('Revenue', 90000), ('Expense', 25000), ('Profit', 65000)),
(("Farm_ID", 21), ("Crop", "Sorghum"), ('Acres_Planted', 85), ('Yield_Per_Acre', 25), ('Total_Production', 2125), ('Market_Price', 9), ('Revenue', 19125), ('Expense', 8000), ('Profit', 11125)),
(("Farm_ID", 4), ("Crop", "Soybeans"), ('Acres_Planted', 120), ('Yield_Per_Acre', 20), ('Total_Production', 2400), ('Market_Price', 7), ('Revenue', 16800), ('Expense', 7500), ('Profit', 9300)),
(("Farm_ID", 1), ("Crop", "Wheat"), ('Acres_Planted', 100), ('Yield_Per_Acre', 30), ('Total_Production', 3000), ('Market_Price', 5), ('Revenue', 15000), ('Expense', 8000), ('Profit', 7000)),
(("Farm_ID", 20), ("Crop", "Lettuce"), ('Acres_Planted', 30), ('Yield_Per_Acre', 70), ('Total_Production', 2100), ('Market_Price', 15), ('Revenue', 31500), ('Expense', 12000), ('Profit', 19500)),
(("Farm_ID", 12), ("Crop", "Cotton"), ('Acres_Planted', 110), ('Yield_Per_Acre', 15), ('Total_Production', 1650), ('Market_Price', 25), ('Revenue', 41250), ('Expense', 12000), ('Profit', 29250)),
(("Farm_ID", 18), ("Crop", "Beans"), ('Acres_Planted', 55), ('Yield_Per_Acre', 40), ('Total_Production', 2200), ('Market_Price', 10), ('Revenue', 22000), ('Expense', 10000), ('Profit', 12000)))
}

agricultural_dataset2 = (
    ("Farm_ID", 24), ("Crop", "Sugar Beets"), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000)),
    ("Farm_ID", 31), ("Crop", "Spinach"), ('Acres_Planted', 15), ('Yield_Per_Acre', 90), ('Total_Production', 1350), ('Market_Price', 18), ('Revenue', 24300), ('Expense', 10000), ('Profit', 14300)),
    ("Farm_ID", 33), ("Crop", "Kale"), ('Acres_Planted', 35), ('Yield_Per_Acre', 80), ('Total_Production', 2800), ('Market_Price', 15), ('Revenue', 42000), ('Expense', 15000), ('Profit', 27000)),
    ("Farm_ID", 7), ("Crop", "Apples"), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000)),
)

print(agricultural_data.intersection(agricultural_dataset2))

```

Output:

```
>>> = RESTART: D:/SMITA/MSc IT CC P1/DSPY/Programs/Agriculture_dataset_sets.py
((('Farm_ID', 7), ('Crop', 'Apples'), ('Acres_Planted', 30), ('Yield_Per_Acre', 100), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 15000), ('Profit', 21000)), ((('Farm_ID', 24), ('Crop', 'Sugar Beets'), ('Acres_Planted', 75), ('Yield_Per_Acre', 40), ('Total_Production', 3000), ('Market_Price', 12), ('Revenue', 36000), ('Expense', 14000), ('Profit', 22000)))
```

j. **symmetric_difference_update()**:

Code:

```
* Agriculture_dataset.sets.py - D:\SMITA\MSIT CC P1\DSPP\Programs\Agriculture_dataset.sets.py (3.11.6)*
File Edit Formulas Run Options Window Help

agricultural_data = [
    {"Farm_ID": 23, "Crop": "Onions", "Acres_Planted": 55, "Yield_Per_Acre": 50, ("Total_Production", 2750), ("Market_Price", 8), ("Revenue", 22000), ("Expense", 11000), ("Profit", 11000)},
    {"Farm_ID": 8, "Crop": "Oranges", "Acres_Planted": 40, "Yield_Per_Acre": 80, ("Total_Production", 3200), ("Market_Price", 15), ("Revenue", 48000), ("Expense", 20000), ("Profit", 28000)},
    {"Farm_ID": 25, "Crop": "Cabbage", "Acres_Planted": 25, "Yield_Per_Acre": 80, ("Total_Production", 2000), ("Market_Price", 10), ("Revenue", 20000), ("Expense", 9000), ("Profit", 11000)},
    {"Farm_ID": 26, "Crop": "Coffee", "Acres_Planted": 50, "Yield_Per_Acre": 200, ("Total_Production", 3000), ("Market_Price", 35), ("Revenue", 105000), ("Expense", 30000), ("Profit", 75000)},
    {"Farm_ID": 24, "Crop": "Sugar Beets", "Acres_Planted": 75, "Yield_Per_Acre": 40, ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 14000), ("Profit", 22000)},
    {"Farm_ID": 7, "Crop": "Apples", "Acres_Planted": 30, ("Yield_Per_Acre", 100), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 15000), ("Profit", 21000)},
    {"Farm_ID": 15, "Crop": "Cucumbers", "Acres_Planted": 45, "Yield_Per_Acre": 80, ("Total_Production", 3600), ("Market_Price", 12), ("Revenue", 43200), ("Expense", 16000), ("Profit", 27200)},
    {"Farm_ID": 11, "Crop": "Tomatoes", "Acres_Planted": 25, "Yield_Per_Acre", 120, ("Total_Production", 3000), ("Market_Price", 20), ("Revenue", 60000), ("Expense", 18000), ("Profit", 42000)},
    {"Farm_ID": 10, "Crop": "Carrots", "Acres_Planted": 70, "Yield_Per_Acre", 60, ("Total_Production", 4200), ("Market_Price", 9), ("Revenue", 37800), ("Expense", 15000), ("Profit", 22800)},
    {"Farm_ID": 16, "Crop": "Oats", "Acres_Planted": 95, "Yield_Per_Acre", 30, ("Total_Production", 2850), ("Market_Price", 7), ("Revenue", 19950), ("Expense", 9500), ("Profit", 10450)},
    {"Farm_ID": 28, "Crop": "Hops", "Acres_Planted": 35, "Yield_Per_Acre", 100, ("Total_Production", 3500), ("Market_Price", 40), ("Revenue", 140000), ("Expense", 35000), ("Profit", 105000)},
    {"Farm_ID": 6, "Crop": "Potatoes", "Acres_Planted": 50, "Yield_Per_Acre", 50, ("Total_Production", 2500), ("Market_Price", 10), ("Revenue", 25000), ("Expense", 12000), ("Profit", 13000)},
    {"Farm_ID": 22, "Crop": "Broccoli", "Acres_Planted": 40, ("Yield_Per_Acre", 60), ("Total_Production", 2400), ("Market_Price", 14), ("Revenue", 33600), ("Expense", 15000), ("Profit", 18600)},
    {"Farm_ID": 14, "Crop": "Sunflowers", "Acres_Planted": 75, "Yield_Per_Acre", 25, ("Total_Production", 1875), ("Market_Price", 22), ("Revenue", 41250), ("Expense", 14000), ("Profit", 27250)},
    {"Farm_ID": 13, "Crop": "Peppers", "Acres_Planted": 35, "Yield_Per_Acre", 90, ("Total_Production", 3150), ("Market_Price", 14), ("Revenue", 44100), ("Expense", 17000), ("Profit", 27100)},
    {"Farm_ID": 3, "Crop": "Rice", "Acres_Planted": 80, "Yield_Per_Acre", 40, ("Total_Production", 3200), ("Market_Price", 8), ("Revenue", 25600), ("Expense", 10000), ("Profit", 15600)},
    {"Farm_ID": 29, "Crop": "Blueberries", "Acres_Planted": 50, ("Yield_Per_Acre", 120), ("Total_Production", 6000), ("Market_Price", 18), ("Revenue", 108000), ("Expense", 28000), ("Profit", 80000)},
    {"Farm_ID": 30, "Crop": "Avocado", "Acres_Planted": 20, ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 40), ("Revenue", 120000), ("Expense", 40000), ("Profit", 80000)},
    {"Farm_ID": 19, "Crop": "Peanuts", "Acres_Planted": 65, ("Yield_Per_Acre", 35), ("Total_Production", 2275), ("Market_Price", 18), ("Revenue", 40950), ("Expense", 12000), ("Profit", 28950)},
    {"Farm_ID": 9, "Crop": "Grapes", "Acres_Planted": 60, ("Yield_Per_Acre", 70), ("Total_Production", 4200), ("Market_Price", 18), ("Revenue", 75600), ("Expense", 25000), ("Profit", 50600)},
    {"Farm_ID": 2, "Crop": "Corn", "Acres_Planted": 150, ("Yield_Per_Acre", 25), ("Total_Production", 3750), ("Market_Price", 4), ("Revenue", 15000), ("Expense", 9000), ("Profit", 6000)},
    {"Farm_ID": 27, "Crop": "Tobacco", "Acres_Planted": 90, ("Yield_Per_Acre", 30), ("Total_Production", 2700), ("Market_Price", 25), ("Revenue", 67500), ("Expense", 20000), ("Profit", 47500)},
    {"Farm_ID": 5, "Crop": "Barley", "Acres_Planted": 90, ("Yield_Per_Acre", 35), ("Total_Production", 3150), ("Market_Price", 6), ("Revenue", 18900), ("Expense", 8500), ("Profit", 10400)},
    {"Farm_ID": 17, "Crop": "Strawberries", "Acres_Planted": 20, ("Yield_Per_Acre", 150), ("Total_Production", 3000), ("Market_Price", 30), ("Revenue", 90000), ("Expense", 25000), ("Profit", 65000)},
    {"Farm_ID": 21, "Crop": "Sorghum", "Acres_Planted": 85, ("Yield_Per_Acre", 25), ("Total_Production", 2125), ("Market_Price", 9), ("Revenue", 19125), ("Expense", 8000), ("Profit", 11125)},
    {"Farm_ID": 4, "Crop": "Soybeans", "Acres_Planted": 120, ("Yield_Per_Acre", 20), ("Total_Production", 2400), ("Market_Price", 7), ("Revenue", 16800), ("Expense", 7500), ("Profit", 9300)},
    {"Farm_ID": 1, "Crop": "Wheat", "Acres_Planted": 100, ("Yield_Per_Acre", 30), ("Total_Production", 2000), ("Market_Price", 5), ("Revenue", 15000), ("Expense", 8000), ("Profit", 7000)},
    {"Farm_ID": 20, "Crop": "Lettuce", "Acres_Planted": 30, ("Yield_Per_Acre", 70), ("Total_Production", 2100), ("Market_Price", 15), ("Revenue", 31500), ("Expense", 12000), ("Profit", 19500)},
    {"Farm_ID": 12, "Crop": "Cotton", "Acres_Planted": 110, ("Yield_Per_Acre", 15), ("Total_Production", 1650), ("Market_Price", 25), ("Revenue", 41250), ("Expense", 12000), ("Profit", 29250)},
    {"Farm_ID": 18, "Crop": "Beans", "Acres_Planted": 55, ("Yield_Per_Acre", 40), ("Total_Production", 2200), ("Market_Price", 10), ("Revenue", 22000), ("Expense", 10000), ("Profit", 12000)}

agricultural_dataset2 = [
    {"Farm_ID": 24, "Crop": "Sugar Beets", "Acres_Planted": 75, "Yield_Per_Acre": 40, ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 14000), ("Profit", 22000)},
    {"Farm_ID": 31, "Crop": "Spinach", "Acres_Planted": 15, ("Yield_Per_Acre", 90), ("Total_Production", 1350), ("Market_Price", 18), ("Revenue", 24300), ("Expense", 10000), ("Profit", 14300)},
    {"Farm_ID": 33, "Crop": "Kale", "Acres_Planted": 35, ("Yield_Per_Acre", 80), ("Total_Production", 2800), ("Market_Price", 15), ("Revenue", 42000), ("Expense", 15000), ("Profit", 27000)},
    {"Farm_ID": 7, "Crop": "Apples", "Acres_Planted": 30, ("Yield_Per_Acre", 100), ("Total_Production", 3000), ("Market_Price", 12), ("Revenue", 36000), ("Expense", 15000), ("Profit", 21000)}]

print(agricultural_data.symmetric_difference_update(agricultural_dataset2))
print(en[agricultural_data])
```

Output:

```
>>> = RESTART: D:\SMITA\MSc IT CC P1\DSPPY\Programs\Agriculture_dataset_sets.py
None
30
>>> |
```

Practical No. 7

Aim: Practical on functions

❖ Functions Operation on Netflix dataset:

a. mean() Code:

```

function new.py - D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py (3.11.7)
File Edit Format Run Options Window Help
import pandas as pd
import random
from datetime import datetime, timedelta
# Set a seed for reproducibility:
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = range(1, num_users + 1)
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)] # Watch date in minutes

# Create a DataFrame
data = [
    {'User_ID': user_ids,
     'Age': ages,
     'Preferred_Genre': preferred_genres,
     'Watch_Time_Minutes': watch_times,
     'Watch_Date': watch_dates
}
]

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Mean, Min, Max, Sum
mean_watch_time = netflix_df['Watch_Time_Minutes'].mean()
print(f"Mean Watch Time: {mean_watch_time} minutes")

```

Output:

```

idle Shell 3.11.7
File Edit Shell Debug Options Window Help
Python 3.11.7 (tags/v3.11.7f7a662, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py
Original Dataset:
   User_ID  Age  Watch_Time_Minutes  Watch_Date
0        1  25            237  2023-12-09 04:19:11.982578
1        2  25            104  2023-11-30 04:19:11.982578
2        3  19            50  2023-12-02 04:19:11.982578
3        4  35            89  2023-11-27 04:19:11.982578
4        5  25            55  2023-12-01 04:19:11.982578
5        6  32            127  2023-12-01 04:19:11.982578
6        7  26            101  2023-11-20 04:19:11.982578
7        8  24            146  2023-12-06 04:19:11.982578
8        9  25            100  2023-11-28 04:19:11.982578
9       10  23            123  2023-12-04 04:19:11.982578
10      11  55            71  2023-11-11 04:19:11.982578
11      12  45            124  2023-11-17 04:19:11.982578
12      13  20            129  2023-11-24 04:19:11.982578
13      14  19            83  2023-11-11 04:19:11.982578
14      15  23            201  2023-12-05 04:19:11.982578
15      16  31            98  2023-11-20 04:19:11.982578
16      17  22            208  2023-11-15 04:19:11.982578
17      18  50            204  2023-11-18 04:19:11.982578
18      19  56            195  2023-11-12 04:19:11.982578
19      20  19            48  2023-11-16 04:19:11.982578
20      21  53            165  2023-11-21 04:19:11.982578
21      22  30            192  2023-12-10 04:19:11.982578
22      23  59            73  2023-11-28 04:19:11.982578
23      24  52            166  2023-12-08 04:19:11.982578
24      25  44            210  2023-11-29 04:19:11.982578
25      26  32            92  2023-11-20 04:19:11.982578
26      27  46            71  2023-11-16 04:19:11.982578
27      28  55            148  2023-11-30 04:19:11.982578
28      29  35            127  2023-11-25 04:19:11.982578
29      30  18            99  2023-11-22 04:19:11.982578

[30 rows x 5 columns]
Mean Watch Time: 131.2 minutes
>>>

```

b. min(): Code:

```

idle Shell 3.11.7
File Edit Format Run Options Window Help
import pandas as pd
import random
from datetime import datetime, timedelta
# Set a seed for reproducibility
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = range(1, num_users + 1)
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)] # Watch date in minutes

# Create a DataFrame
data = [
    {'User_ID': user_ids,
     'Age': ages,
     'Preferred_Genre': preferred_genres,
     'Watch_Time_Minutes': watch_times,
     'Watch_Date': watch_dates
}
]

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Mean, Min, Max, Sum
min_watch_time = netflix_df['Watch_Time_Minutes'].min()
print(f"Minimum Watch Time: {min_watch_time} minutes")

```

Output:

```
= RESTART: D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py
Original Dataset:
   User_ID  Age ... Watch_Time_Minutes      Watch_Date
0       1  58 ...          227 2023-12-09 04:21:01.795066
1       2  25 ...          104 2023-11-30 04:21:01.795066
2       3  19 ...          50 2023-12-02 04:21:01.795066
3       4  35 ...          89 2023-11-27 04:21:01.795066
4       5  33 ...          55 2023-11-17 04:21:01.795066
5       6  32 ...          172 2023-12-02 04:21:01.795066
6       7  26 ...          101 2023-11-20 04:21:01.795066
7       8  24 ...          146 2023-12-06 04:21:01.795066
8       9  52 ...          192 2023-12-04 04:21:01.795066
9      10  23 ...          123 2023-12-04 04:21:01.795066
10     11  55 ...          71 2023-11-11 04:21:01.795066
11     12  45 ...          124 2023-11-17 04:21:01.795066
12     13  20 ...          120 2023-12-06 04:21:01.795066
13     14  19 ...          83 2023-11-11 04:21:01.795066
14     15  23 ...          201 2023-12-05 04:21:01.795066
15     16  31 ...          98 2023-11-20 04:21:01.795066
16     17  32 ...          204 2023-11-23 04:21:01.795066
17     18  50 ...          101 2023-11-20 04:21:01.795066
18     19  56 ...          146 2023-11-18 04:21:01.795066
19     20  19 ...          195 2023-11-12 04:21:01.795066
20     21  53 ...          48 2023-11-16 04:21:01.795066
21     22  30 ...          185 2023-12-09 04:21:01.795066
22     23  59 ...          192 2023-12-10 04:21:01.795066
23     24  52 ...          73 2023-11-28 04:21:01.795066
24     25  44 ...          166 2023-12-08 04:21:01.795066
25     26  32 ...          216 2023-12-02 04:21:01.795066
26     27  46 ...          92 2023-11-20 04:21:01.795066
27     28  55 ...          71 2023-11-16 04:21:01.795066
28     29  35 ...          146 2023-11-20 04:21:01.795066
29     30  18 ...          127 2023-11-25 04:21:01.795066
30  ...  ...          99 2023-11-22 04:21:01.795066
```

[30 rows x 5 columns]

Minimum Watch Time: 48 minutes

c. max() Code:

function new.py - D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py (3:11)

```
File Edit Format Run Options Window Help
import pandas as pd
import random
from datetime import datetime, timedelta

# Set a seed for reproducibility
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = range(1, num_users + 1)
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)]

# Create a DataFrame
data = {
    'User_ID': user_ids,
    'Age': ages,
    'Preferred_Genre': preferred_genres,
    'Watch_Time_Minutes': watch_times,
    'Watch_Date': watch_dates
}

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Mean, Min, Max, Sum
max_watch_time = netflix_df['Watch_Time_Minutes'].max()
print(f"Maximum Watch Time: {max_watch_time} minutes")
```

Output:

```
>>> = RESTART: D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py
Original Dataset:
   User_ID  Age ... Watch_Time_Minutes      Watch_Date
0       1  58 ...          227 2023-12-09 04:23:41.565251
1       2  25 ...          104 2023-11-30 04:23:41.565251
2       3  19 ...          50 2023-12-02 04:23:41.565251
3       4  35 ...          89 2023-11-27 04:23:41.565251
4       5  33 ...          55 2023-11-17 04:23:41.565251
5       6  32 ...          127 2023-12-01 04:23:41.565251
6       7  26 ...          101 2023-11-20 04:23:41.565251
7       8  24 ...          146 2023-12-06 04:23:41.565251
8       9  52 ...          192 2023-12-04 04:23:41.565251
9      10  23 ...          123 2023-12-04 04:23:41.565251
10     11  55 ...          71 2023-11-11 04:23:41.565251
11     12  45 ...          124 2023-11-17 04:23:41.565251
12     13  20 ...          120 2023-12-06 04:23:41.565251
13     14  19 ...          83 2023-11-11 04:23:41.565251
14     15  23 ...          201 2023-12-05 04:23:41.565251
15     16  31 ...          98 2023-11-20 04:23:41.565251
16     17  32 ...          209 2023-11-22 04:23:41.565251
17     18  50 ...          101 2023-11-18 04:23:41.565251
18     19  56 ...          146 2023-11-12 04:23:41.565251
19     20  19 ...          195 2023-11-16 04:23:41.565251
20     21  53 ...          48 2023-11-16 04:23:41.565251
21     22  30 ...          185 2023-12-09 04:23:41.565251
22     23  59 ...          192 2023-12-10 04:23:41.565251
23     24  52 ...          73 2023-11-28 04:23:41.565251
24     25  44 ...          166 2023-12-08 04:23:41.565251
25     26  32 ...          216 2023-12-02 04:23:41.565251
26     27  46 ...          92 2023-11-20 04:23:41.565251
27     28  55 ...          148 2023-11-30 04:23:41.565251
28     29  35 ...          127 2023-11-25 04:23:41.565251
29     30  18 ...          99 2023-11-22 04:23:41.565251
```

[30 rows x 5 columns]

Maximum Watch Time: 227 minutes

d. sum() Code:

```

function new.py - D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py (3.11.7)
File Edit Format Run Options Window Help
import pandas as pd
import random
from datetime import datetime, timedelta

# Set a seed for reproducibility
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = range(1, num_users + 1)
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)]

# Create a DataFrame
data = [
    {'User_ID': user_ids,
     'Age': ages,
     'Preferred_Genre': preferred_genres,
     'Watch_Time_Minutes': watch_times,
     'Watch_Date': watch_dates,
    }
]

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Mean, Min, Max, Sum
total_watch_time = netflix_df['Watch_Time_Minutes'].sum()
print(f"Total Watch Time: {total_watch_time} minutes")

```

Output:

```

= RESTART: D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py
Original Dataset:
   User_ID  Age  Watch_Time_Minutes  Watch_Date
0        1  58             227 2023-12-09 04:25:30.024008
1        2  25             104 2023-11-30 04:25:30.024008
2        3  19              50 2023-12-02 04:25:30.024008
3        4  35              89 2023-11-27 04:25:30.024008
4        5  33              55 2023-11-17 04:25:30.024008
5        6  32             127 2023-12-01 04:25:30.024008
6        7  26              101 2023-11-20 04:25:30.024008
7        8  24              146 2023-12-06 04:25:30.024008
8        9  52              192 2023-12-04 04:25:30.024008
9       10  23              123 2023-12-04 04:25:30.024008
10      11  55              71 2023-11-11 04:25:30.024008
11      12  45              124 2023-11-17 04:25:30.024008
12      13  20              120 2023-12-06 04:25:30.024008
13      14  19              83 2023-11-11 04:25:30.024008
14      15  23              201 2023-12-05 04:25:30.024008
15      16  31              98 2023-11-20 04:25:30.024008
16      17  32              209 2023-11-22 04:25:30.024008
17      18  50              204 2023-11-18 04:25:30.024008
18      19  56              193 2023-11-12 04:25:30.024008
19      20  19              48 2023-11-16 04:25:30.024008
20      21  53              185 2023-12-09 04:25:30.024008
21      22  30              195 2023-12-10 04:25:30.024008
22      23  59              73 2023-11-28 04:25:30.024008
23      24  52              166 2023-12-08 04:25:30.024008
24      25  44              216 2023-12-02 04:25:30.024008
25      26  32              92 2023-11-20 04:25:30.024008
26      27  46              71 2023-11-16 04:25:30.024008
27      28  55              148 2023-11-30 04:25:30.024008
28      29  35              127 2023-11-25 04:25:30.024008
29      30  18              99 2023-11-22 04:25:30.024008

```

[30 rows x 5 columns]
Total Watch Time: 3936 minutes

e. agg() – aggregate Code:

```

function new.py - D:/SMITA/MSc IT CC P1/DSPY/Programs/function new.py (3.11.7)
File Edit Format Run Options Window Help
import pandas as pd
import random
from datetime import datetime, timedelta

# Set a seed for reproducibility
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = range(1, num_users + 1)
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)]

# Create a DataFrame
data = [
    {'User_ID': user_ids,
     'Age': ages,
     'Preferred_Genre': preferred_genres,
     'Watch_Time_Minutes': watch_times,
     'Watch_Date': watch_dates,
    }
]

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Aggregate function
aggregated_data = netflix_df.groupby('Preferred_Genre').agg({'Watch_Time_Minutes': ['mean', 'min', 'max', 'sum']})
print("\nAggregate by Preferred Genre:")
print(aggregated_data)

```

Output:

```

File Edit Shell Debug Options Window Help
>>> = RESTART: D:/SMITA/MSc IT CC P1/DSPV/Programs/function new.py
Original Dataset:
User_ID Age ... Watch_Time_Minutes Watch_Date
0 1 58 ... 227 2023-12-09 04:26:38.963541
1 2 25 ... 104 2023-11-30 04:26:38.963541
2 3 19 ... 50 2023-12-02 04:26:38.963541
3 4 35 ... 89 2023-11-27 04:26:38.963541
4 5 33 ... 55 2023-11-17 04:26:38.963541
5 6 22 ... 101 2023-11-20 04:26:38.963541
6 7 26 ... 101 2023-11-20 04:26:38.963541
7 8 24 ... 146 2023-12-06 04:26:38.963541
8 9 52 ... 123 2023-12-04 04:26:38.963541
9 10 28 ... 71 2023-11-11 04:26:38.963541
11 11 55 ... 104 2023-11-18 04:26:38.963541
12 13 20 ... 120 2023-12-06 04:26:38.963541
14 14 29 ... 120 2023-12-06 04:26:38.963541
15 15 23 ... 201 2023-12-06 04:26:38.963541
16 16 31 ... 201 2023-11-20 04:26:38.963541
17 17 22 ... 201 2023-11-20 04:26:38.963541
18 18 50 ... 195 2023-11-12 04:26:38.963541
19 19 56 ... 201 2023-11-20 04:26:38.963541
20 21 53 ... 185 2023-12-09 04:26:38.963541
21 22 30 ... 192 2023-12-10 04:26:38.963541
22 23 29 ... 192 2023-12-10 04:26:38.963541
23 24 52 ... 166 2023-12-08 04:26:38.963541
24 25 44 ... 216 2023-12-08 04:26:38.963541
25 26 22 ... 71 2023-11-16 04:26:38.963541
26 27 46 ... 148 2023-11-30 04:26:38.963541
27 28 55 ... 157 2023-11-25 04:26:38.963541
29 29 57 ... 99 2023-11-22 04:26:38.963541
[130 rows x 5 columns]

Aggregate by Preferred_Genre:
   Watch_Time_Minutes
   mean min max sum
Preferred_Genre
Action      145.25 71 195 1162
Comedy     115.80 55 227 579
Documentary 147.00 83 206 737
Drama      123.00 50 204 974
Science Fiction 121.00 49 209 404

```

f. Average Code:

```

File Edit Format Run Options Window Help
File function new.py - D:/SMITA/MSc IT CC P1/DSPV/Programs/function new.py (3.11.7)
import pandas as pd
import random
from datetime import datetime, timedelta

# Set a seed for reproducibility
random.seed(42)

# Generate synthetic data for a smaller dataset
num_users = 30
user_ids = [f'U_{i}' for i in range(num_users)]
ages = [random.randint(18, 60) for _ in range(num_users)]
genres = ['Action', 'Comedy', 'Drama', 'Science Fiction', 'Documentary']
preferred_genres = [random.choice(genres) for _ in range(num_users)]
watch_times = [random.randint(30, 240) for _ in range(num_users)] # Watch time in minutes

# Generate random dates for the past month
start_date = datetime.now() - timedelta(days=30)
end_date = datetime.now()
watch_dates = [start_date + timedelta(days=random.randint(0, 30)) for _ in range(num_users)]

# Create a DataFrame
data = [
    {'User_ID': user_ids,
     'Age': ages,
     'Preferred_Genre': preferred_genres,
     'Watch_Time_Minutes': watch_times,
     'Watch_Date': watch_dates
}
]

netflix_df = pd.DataFrame(data)

# Display the smaller dataset
print("Original Dataset:")
print(netflix_df)

# Average calculation
average_age = netflix_df['Age'].mean()
print(f"\nAverage Age: {average_age} years")

```

Output:

```

File Edit Shell Debug Options Window Help
Python 3.11.7 (tags/v3.11.7:faf7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> = RESTART: D:/SMITA/MSc IT CC P1/DSPV/Programs/function new.py
Original Dataset:
User_ID Age ... Watch_Time_Minutes Watch_Date
0 1 58 ... 227 2023-12-09 04:28:53.492791
1 2 25 ... 104 2023-11-30 04:28:53.492791
2 3 19 ... 50 2023-12-02 04:28:53.492791
3 4 35 ... 89 2023-11-27 04:28:53.492791
4 5 33 ... 55 2023-11-17 04:28:53.492791
5 6 32 ... 127 2023-12-01 04:28:53.492791
6 7 26 ... 101 2023-11-20 04:28:53.492791
7 8 24 ... 146 2023-12-06 04:28:53.492791
8 9 52 ... 192 2023-12-04 04:28:53.492791
9 10 23 ... 123 2023-12-04 04:28:53.492791
10 11 55 ... 71 2023-11-11 04:28:53.492791
11 12 45 ... 124 2023-11-17 04:28:53.492791
12 13 20 ... 120 2023-12-06 04:28:53.492791
13 14 19 ... 83 2023-11-11 04:28:53.492791
14 15 23 ... 201 2023-12-05 04:28:53.492791
15 16 31 ... 98 2023-11-20 04:28:53.492791
16 17 32 ... 209 2023-11-22 04:28:53.492791
17 18 50 ... 203 2023-11-18 04:28:53.492791
18 19 56 ... 195 2023-11-12 04:28:53.492791
19 20 19 ... 48 2023-11-16 04:28:53.492791
20 21 53 ... 181 2023-12-09 04:28:53.492791
21 22 30 ... 192 2023-12-10 04:28:53.492791
22 23 59 ... 73 2023-11-28 04:28:53.492791
23 24 52 ... 166 2023-12-08 04:28:53.492791
24 25 44 ... 216 2023-12-02 04:28:53.492791
25 26 32 ... 92 2023-11-20 04:28:53.492791
26 27 46 ... 71 2023-11-16 04:28:53.492791
27 28 55 ... 148 2023-11-30 04:28:53.492791
28 29 35 ... 127 2023-11-25 04:28:53.492791
29 30 18 ... 99 2023-11-22 04:28:53.492791
[30 rows x 5 columns]

Average Age: 36.7 years

```

Practical 8 Aim:

Practical on exception handling and file handling

❖ File Handling Operation in Python:

1) Open file:

Code:

```
f = open("my_file.txt")
f.close()
```

2) Read: Code:

```
# open file in current directory
file1 = open("example.txt", "r")

# read the file
read_content = file1.read()
print(read_content)
```

Output:

```
>>> = RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\file handling.py
Hello !
This is an example file.
```

3) Write: Code:

```
# open file in current directory
file1 = open("example.txt", "w")
file1.write("Hello!....")
file1.close()

file1=open("example.txt", "r")
print(file1.read())
```

Output:

```
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\file handling.py
Hello!....
```



4) Append: Code:

```
# open file in current directory
file1 = open("example.txt", "a")
file1.write("Now the file has more content!")
file1.close()

file1=open("example.txt", "r")
print(file1.read())
```

Output:

```
>>> = RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\file handling.py
Hello!....Now the file has more content!Now the file has more content!
```

5) Exclusive creation: Code:

```
f = open("my_file.txt", "x")
```

Output:

A screenshot of a Windows File Explorer window. The path is 'This PC > DATA (D) > SMITA > MSc IT CC P1 > DSPY > Programs > prgs'. Inside the 'prgs' folder, there is a file named 'my_file'. The file details are: Name: my_file, Date modified: 11-12-2023 05:18, Type: Text Document, Size: 0 KB.

6) Close file:

Code:

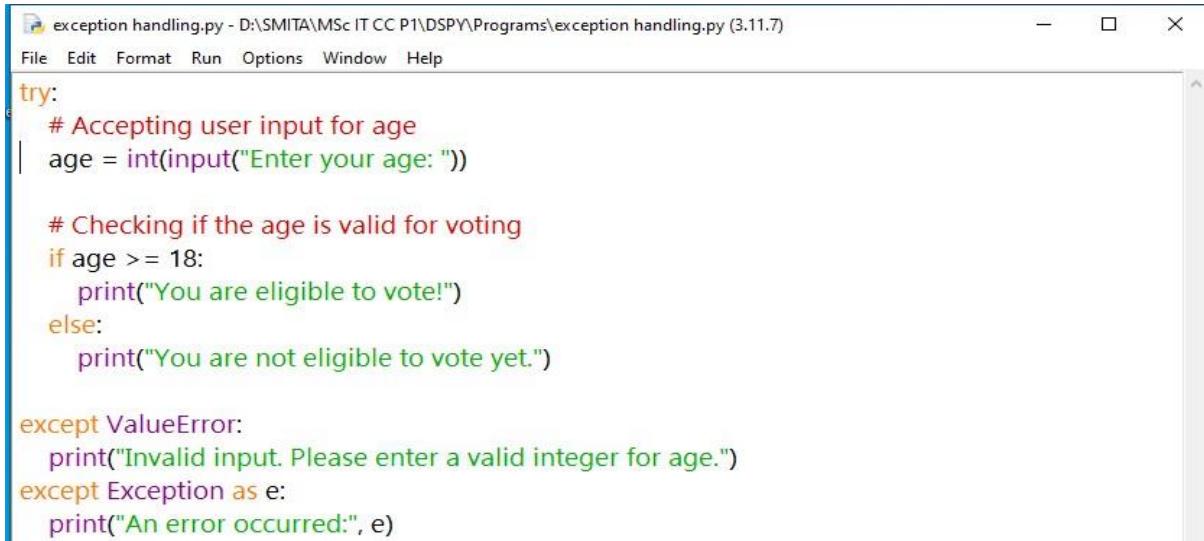


```
f = open("my_file.txt")
```

❖ Exception Handling in Python

Q.1. Accept the value from user of age to check it is valid for voting or not.

Code:

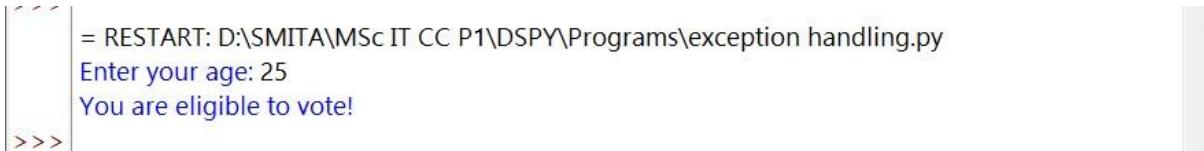


```
try:
    # Accepting user input for age
    age = int(input("Enter your age: "))

    # Checking if the age is valid for voting
    if age >= 18:
        print("You are eligible to vote!")
    else:
        print("You are not eligible to vote yet.")

except ValueError:
    print("Invalid input. Please enter a valid integer for age.")
except Exception as e:
    print("An error occurred:", e)
```

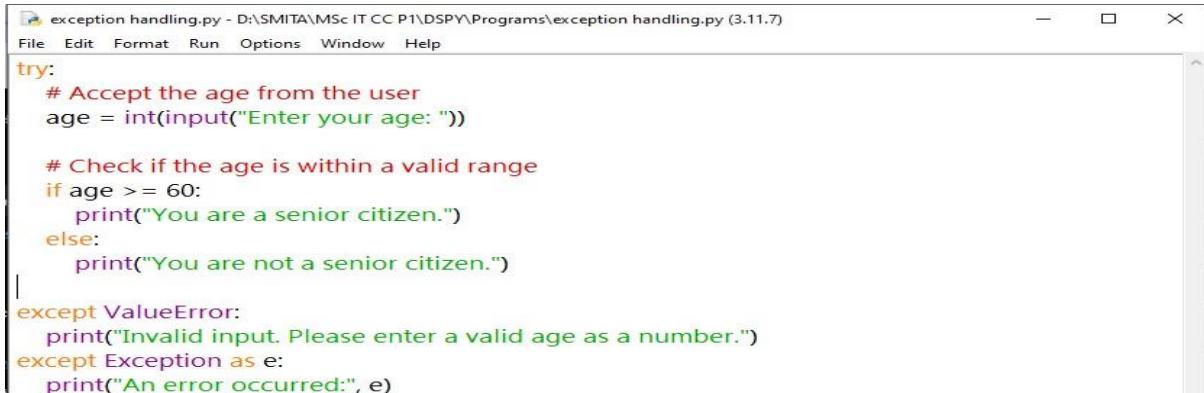
Output:



```
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\exception handling.py
Enter your age: 25
You are eligible to vote!
```

Q.2. Accept the value from user of age to check it is senior citizen or not.

Code:



```
try:
    # Accept the age from the user
    age = int(input("Enter your age: "))

    # Check if the age is within a valid range
    if age >= 60:
        print("You are a senior citizen.")
    else:
        print("You are not a senior citizen.")

except ValueError:
    print("Invalid input. Please enter a valid age as a number.")
except Exception as e:
    print("An error occurred:", e)
```

Output:

```
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\exception handling.py
Enter your age: 65
You are a senior citizen.
```

Q.3. Accept experience and salary from employee if experience is 5+ then 15% increment in salary.

Code:

```
exception handling.py - D:\SMITA\MSc IT CC P1\DSPY\Programs\exception handling.py (3.11.7)
File Edit Format Run Options Window Help
class InvalidInputError(Exception):
    pass

def calculate_salary_increment(experience, current_salary):
    try:
        # Validate input for experience
        if not isinstance(experience, int) or experience < 0:
            raise InvalidInputError("Experience should be a non-negative integer.")

        # Validate input for current salary
        if not isinstance(current_salary, (int, float)) or current_salary < 0:
            raise InvalidInputError("Current salary should be a non-negative number.")

        # Check if the employee has 5 or more years of experience
        if experience >= 5:
            # Apply a 15% increment in salary
            new_salary = current_salary * 1.15
            return new_salary
        else:
            return current_salary
    except InvalidInputError as e:
        print(f"Invalid input: {e}")
        return None

try:
    experience_input = int(input("Enter years of experience: "))
    salary_input = float(input("Enter current salary: "))

    incremented_salary = calculate_salary_increment(experience_input, salary_input)

    if incremented_salary is not None:
        print(f"New salary: ${incremented_salary:.2f}")
except ValueError:
    print("Invalid input. Please enter valid numeric values.")
```

Output:

```
>>>
= RESTART: D:\SMITA\MSc IT CC P1\DSPY\Programs\exception handling.py
Enter years of experience: 8
Enter current salary: 75000
New salary: $86250.00
```

Practical 9

Aim: Practical on regular expression

1. '\': use to draw special meaning of character following it.

Code:

```
main.py > ...
1 import re
2 a = 'python is Programming Language'
3 b ='Apthon'
4 x=re.match(b,a)
5 if x:
6     print('yes, Matches')
7 else:
8     print('No Match')
```

Output:

```
Run
yes, Matches
```

2. '['] Returns a match for any lower-case character,

alphabetically between a and n

Code:

```
main.py > ...
1 import re
2 a = 'python is Programming Language'
3 b ='[a-n]'
4 x=re.findall(b,a)
5 if x:
6     print('yes, Matches')
7     print(x)
8 else:
9     print('No Match')
```

Output:

```
Run
109ms on 20:39:09, 12/15 ✓
yes, Matches
['h', 'n', 'i', 'g', 'a', 'm', 'i', 'n', 'g', 'a', 'n', 'g', 'a', 'g', 'e']
```

3. '^' Starts with.

Code:

```
main.py > ...
1 import re
2 txt='python is case sensitive'
3 p='^p'
4 x=re.match(p,txt)
5 if x:
6     print('yes, Matches')
7 else:
8     print('No Match')
```

Output:

```
yes, Matches
```

4. '\$' Ends with.

Code:

```
main.py > ...
1 import re
2 txt='Rahul'
3 p='....l$'
4 result=re.match(p,txt)
5 if result:
6     print('yes, Matches')
7 else:
8     print('No Match')
```

Output:

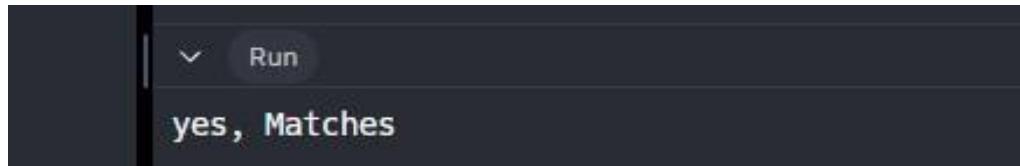
```
yes, Matches
```

5. '.' Any character (except newline character)

Code:

```
main.py > ...
1 import re
2 txt = "hello world"
3 x = re.match('he..o', txt)
4 if x:
5     print('yes, Matches')
6 else:
7     print('No Match')
```

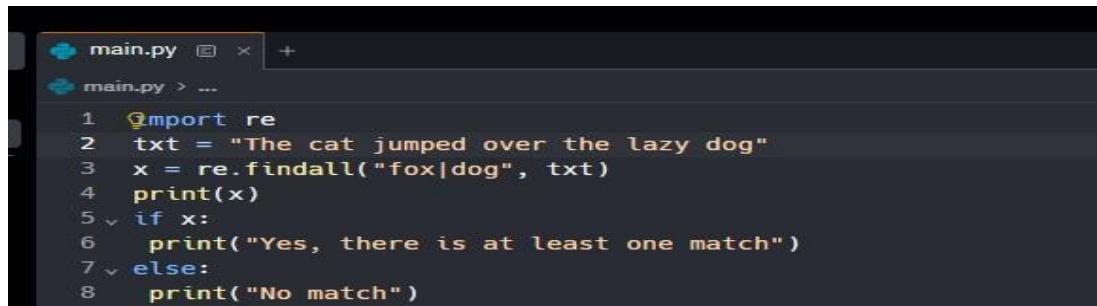
Output:



```
Run
yes, Matches
```

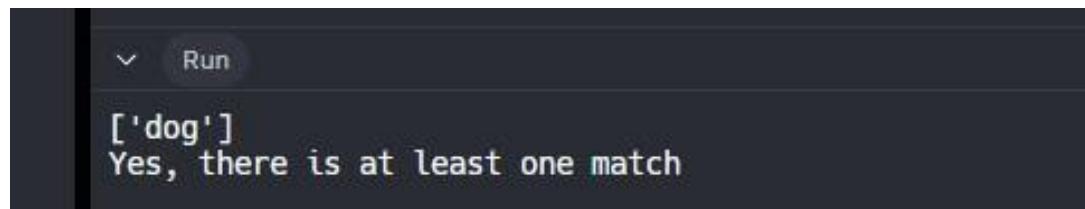
6. '|' Check if the string contains either "fox" or "dog":

Code:



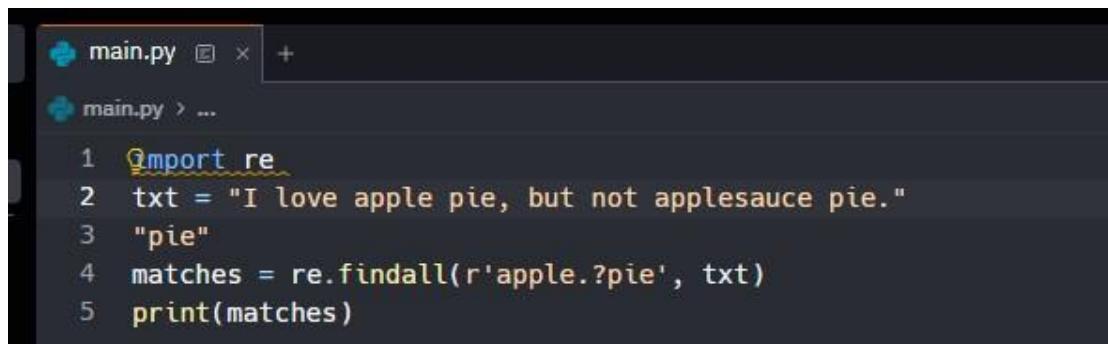
```
main.py
main.py > ...
1 import re
2 txt = "The cat jumped over the lazy dog"
3 x = re.findall("fox|dog", txt)
4 print(x)
5 if x:
6     print("Yes, there is at least one match")
7 else:
8     print("No match")
```

Output:



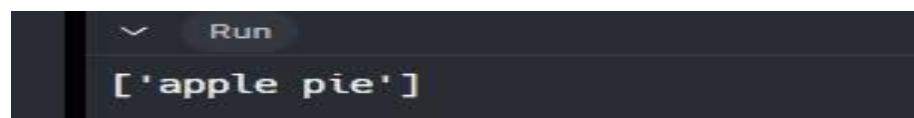
```
Run
['dog']
Yes, there is at least one match
```

7. '?' Zero or one occurrences Code:



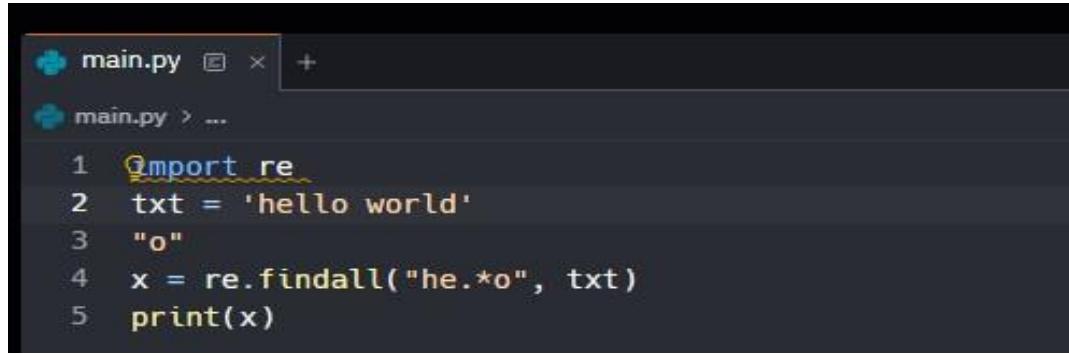
```
main.py
main.py > ...
1 import re
2 txt = "I love apple pie, but not applesauce pie."
3 "pie"
4 matches = re.findall(r'apple.?pie', txt)
5 print(matches)
```

Output:



```
Run
['apple pie']
```

8. '*' Zero or more occurrences Code:



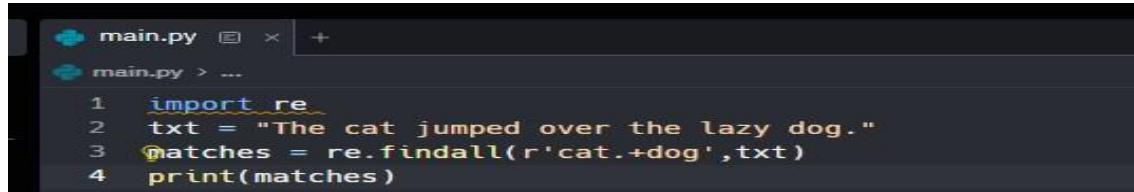
```
main.py +  
main.py > ...  
1 import re  
2 txt = 'hello world'  
3 "o"  
4 x = re.findall("he.*o", txt)  
5 print(x)
```

Output:



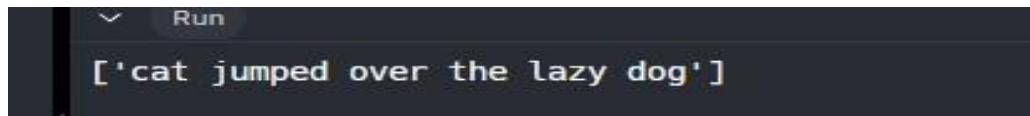
```
Run  
['hello wo']
```

9. '+' One or more occurrences Code:



```
main.py +  
main.py > ...  
1 import re  
2 txt = "The cat jumped over the lazy dog."  
3 matches = re.findall(r'cat.+dog',txt)  
4 print(matches)
```

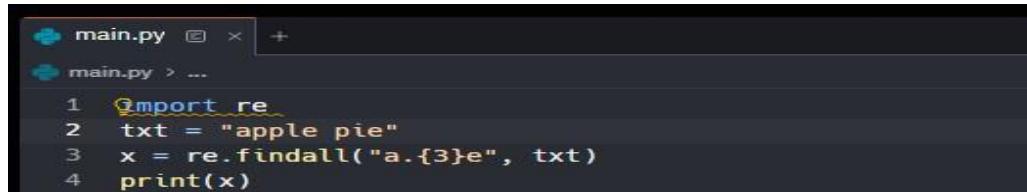
Output:



```
Run  
['cat jumped over the lazy dog']
```

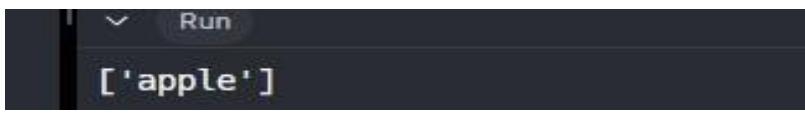
10. '{}' Exactly the specified number of occurrences

Code:



```
main.py +  
main.py > ...  
1 import re  
2 txt = "apple pie"  
3 x = re.findall("a.{3}e", txt)  
4 print(x)
```

Output:



```
Run  
['apple']
```

11. Enclose a group of regex Code:

```
main.py +  
main.py > ...  
1 import re  
2 txt = "The price of apples is $1.25 and the price of oranges is $0.75."  
3 matches = '(oranges|apples) is \$(\d+\.\d+)'  
4 for match in re.finditer(matches, txt):  
5     print(f"The price of '{match.group()}'")
```

Output:

```
Run  
The price of 'apples' is '$1.25'  
The price of 'oranges' is '$0.75'
```

❖ Application of Regular Expression 1.

Web scraping and Data Collection (url)

Code:

```
main.py +  
main.py > ...  
1 import re  
2 import requests  
3 # Example: Extracting URLs from the YouTube Home page  
4 url = 'https://accounts.google.com/v3/signin/identifier?  
authuser=0&continue=https%3A%2F%2Fmail%2Fdata&ec=GAlAFw&hl=en-  
GB&service=mail&flowName=GlifWebSignIn&flowEntry=AddSession&5775%3A1702654890765744&the-  
me=glif'  
5 response = requests.get(url)  
6 # Check if the request was successful (status code 200)  
7 if response.status_code == 200:  
8     # Extracting URLs using regular expression  
9     html_content = response.text  
10    url_pattern = re.compile(r'href="(https?:\/\/.*?)"')  
11    urls = url_pattern.findall(html_content)  
12    # Displaying the extracted URLs  
13    for extracted_url in urls:  
14        print(extracted_url)  
15    else:  
16        print(f"Failed to retrieve the webpage. Status Code: {response.status_code}")
```

Output:

```
Run 1s on 21:11:51, 12/15 ✓  
https://accounts.google.com/v3/signin/  
https://www.gstatic.com/_/mss/boq-identity/_/ss/k=boq-identity.AccountsSignInUi.EcigJeQdnIY.L.  
W.O/am=P4GFJI4NgBgzyzl_zzg5DAAAAAAQAAAsAawgw/d=1/ed=1/rs=A0aEmlGf6vdU1wnYZish42M5Ym3ccNcRbQ/m  
=identifierview,_b,_tp  
https://support.google.com/accounts?p=signin_privatebrowsing&hl=en-GB  
https://support.google.com/accounts?hl=en-GB&p=account_iph  
https://accounts.google.com/TOS?loc=IN&hl=en-GB&privacy=true  
https://accounts.google.com/TOS?loc=IN&hl=en-GB
```

2. Natural Language Processing Code:

```
main.py +  
main.py > ...  
1 import re  
2 text="Natural language processing is machine learning technology"  
3 tokens = re.findall(r'\b\w+\b', text)  
4 print(tokens)
```

Output:

```
Run 141ms on 21:18:33, 12/15 ✓  
['Natural', 'language', 'processing', 'is', 'machine', 'learning', 'technology']
```

3. Entity Recognition:

Code:

```
main.py +  
main.py > ...  
1 import re  
2 text = "Appointment on 2022-12-15 at 2:30 PM. Call 123-456-7891 for details."  
3 dates = re.findall(r'\b\d{4}-\d{2}-\d{2}\b', text)  
4 phone_numbers = re.findall(r'\d{3}-\d{3}-\d{4}', text)  
5 print(dates, '\n', phone_numbers)
```

Output:

```
Run  
['2022-12-15']  
['123-456-7891']
```

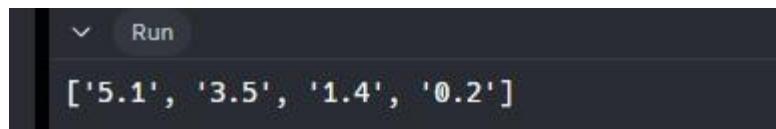
3. pattern detection for iris, emails, text, date time manipulation

a) Pattern Detection in Iris Data:

Code:

```
main.py +  
main.py > ...  
1 import re  
2 iris_data = "SepalLength:5.1 SepalWidth:3.5 PetalLength:1.4 PetalWidth:0.2"  
3 matches = re.findall(r'\d+\.\d+',iris_data)  
4 print(matches)
```

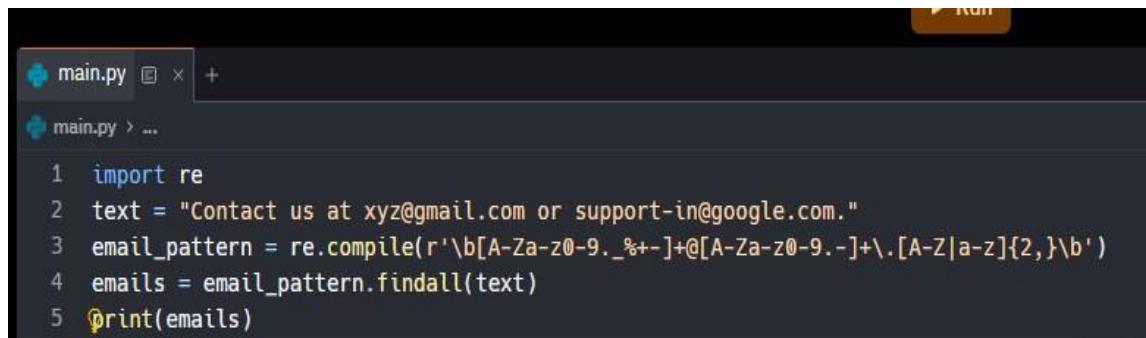
Output:



```
Run
['5.1', '3.5', '1.4', '0.2']
```

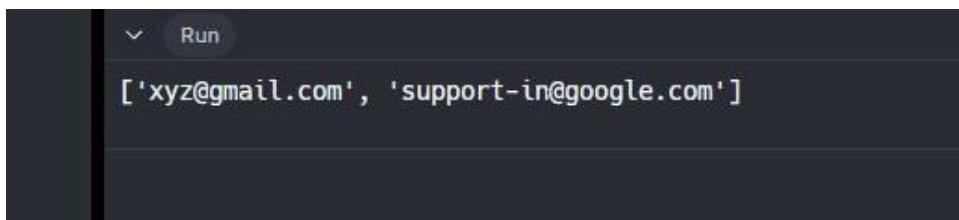
b) Email Pattern Detection:

Code:



```
main.py > ...
main.py > ...
Run
import re
text = "Contact us at xyz@gmail.com or support-in@google.com."
email_pattern = re.compile(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b')
emails = email_pattern.findall(text)
print(emails)
```

Output:



```
Run
['xyz@gmail.com', 'support-in@google.com']
```

c) Text Detection:

Code:



```
main.py > ...
main.py > ...
Run
import re
text = "The quick brown fox jumps over the lazy dog."
match = re.search(r'brown fox',text)
if match:
    print("Pattern found:", match.group())
else:
    print("Pattern not found.")
```

Output:

```
Pattern found: brown fox
```

d) Date and Time Manipulation:

Code:

```
main.py
import re
date_time_string = "2023-12-14 14:24:00"
datetime_pattern = re.compile(r'(\d{4})-(\d{2})-(\d{2}) (\d{2}):(\d{2}):(\d{2})')
match = datetime_pattern.match(date_time_string)
if match:
    year, month, day, hour, minute, second = match.groups()
    print(f"Year: {year}, Month: {month}, Day: {day}, Hour: {hour}, Minute: {minute}, Second {second}")
else:
    print("Invalid date-time format.")
```

Output:

```
Year: 2023, Month: 12, Day: 14, Hour: 14, Minute: 24, Second 00
```