LAMRIN TECH SKILLS UNIVERSITY

PROGRAM FILE
Using python

NAME- SARVAMM PRATAP SINGH RATHORE

CLASS- DOSSIER

ROLL NO. 24100020013

SUBJECT -PYTHON

SEMESTER-1ST

FACULTY INCHARGE SIGNATURE-

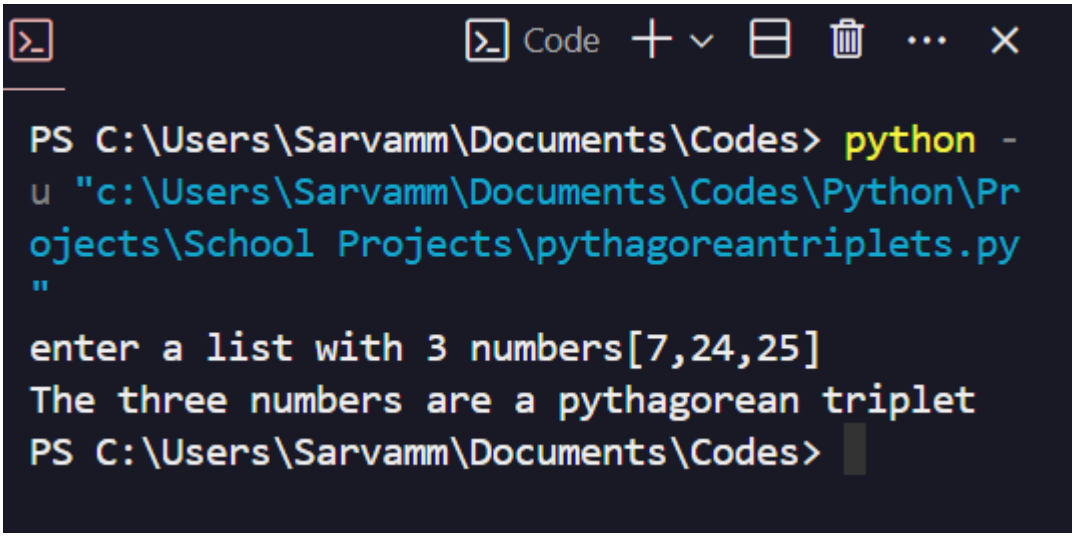
Index

1. Pythagorean Triplet

Aim: To figure out if the three numbers entered by user form a pythagorean triplet or not

```
x = eval(input("enter a list with 3 numbers"))
if len(x) != 3:
    print("Error: List should contain exactly 3 numbers")
elif len(x) == 3:
    for i in x:
        if type(i) != int:
            print("Error: List should contain only integer
s")
```

```
            break

    c = x.pop(x.index(max(x)))
    a, b = x[0], x[-1]
    if c**2 == a**2 + b**2:
        print("The three numbers are a pythagorean triple
t")
    else:
        print("The three numbers are not a pythagorean trip
let")
```



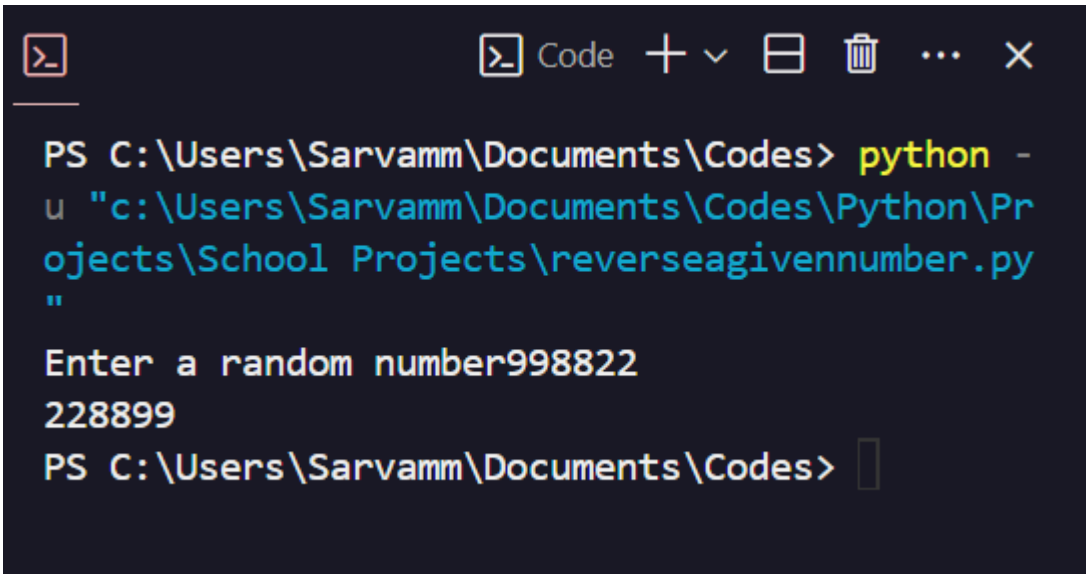
---

2. Reverse a Given number

Aim: To reverse a given string of numbers using different methods

```
import math
x = int(input("Enter a random number"))
#method 1
# x = str(x)
# print(int(x[::-1]))
#method2
# r=""
# while x%10 > 0:
#     r = r + str(x%10)
```

```
#     x = x//10
# print(r)
#method 3
#without using strings:
r, num  = 0 , x
k = int(math.log10(x))     # number of digits in the given nu
while num/10 > 0:
    r = r + (num%10)*(10**k)
    k -= 1
    num = num//10
print(r)
```



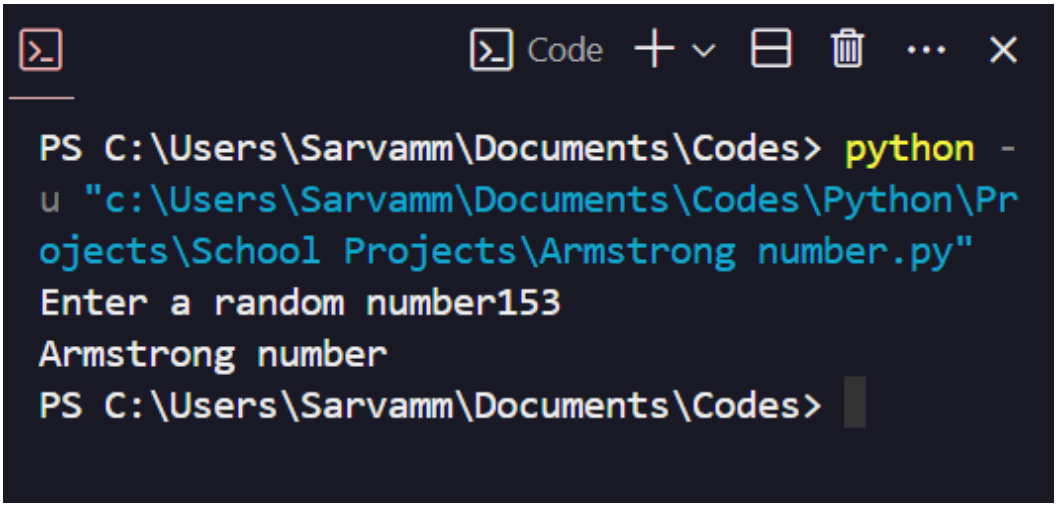
---

3. Check if a number is Armstrong number or not

Aim: To check whether integer input by user is an Armstrong number or not

```
#Check whether armstrong number or not
import math
x  = int(input("Enter a random number"))
digits =  int(math.log10(x)) + 1
r, num = 0,x

while num/10 > 0:
```

```
    r = r + (num%10)**digits
    num = num//10
if r == x:
    print("Armstrong number")
else:
    print("Non-armstrong number")
```



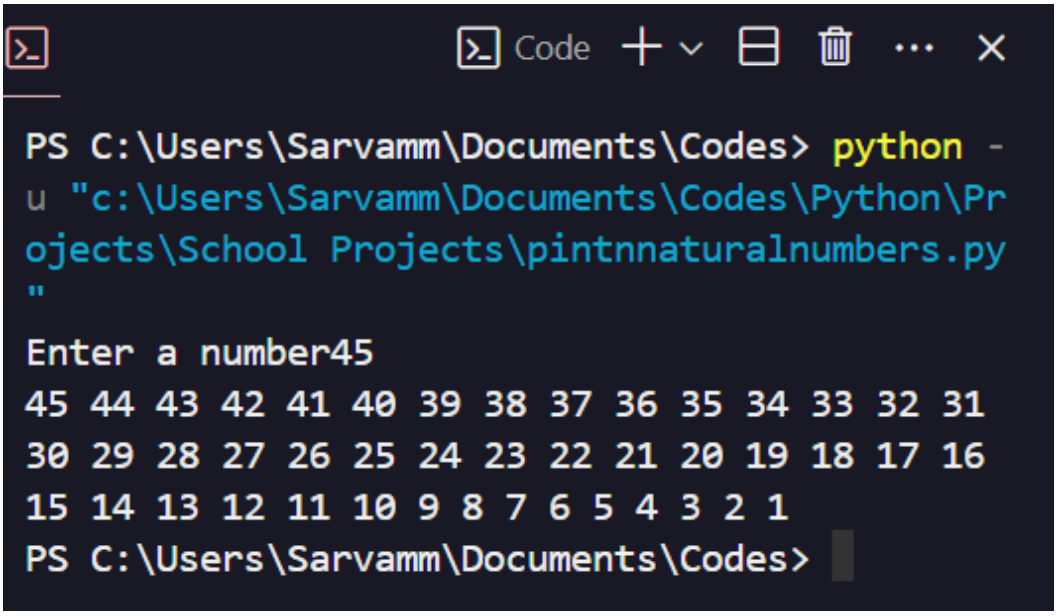
---

4. Print n natural numbers

Aim: To print n natural numbers, n is input given by user

```
#Print n natural numbers

#Using for loop
x = int(input("Enter a number"))
# for i in range(1,x+1):
#     print(i)

#Using Recursion
def pnn(x):
    if x == 0:
        return
    print(x, end=" ")
    x -= 1
```

```
    pnn(x)
pnn(x)
```



---

5. Remove vowels and punctuations

Aim: To remove vowels and punctuations from a given string

```
x = input("Enter text")
for i in x:
    if i in "AEIOUaeiou!@#$%^&*()":
        x = x.replace(i, "")
print(x)
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\removevowelsandpunctuations.py"
Enter texthello! hi! how are u & your mate ###??
hll h hw r   yr mt ??
PS C:\Users\Sarvamm\Documents\Codes>
```

---

6. Count the number of strings

Aim: To count the occurrence of a given substring in another given string

```
#Count no of strings
x = input("Enter text\n")
y = input("Enter word you want the count of:\n")

l = x.split()
print("count = ", l.count(y))
```

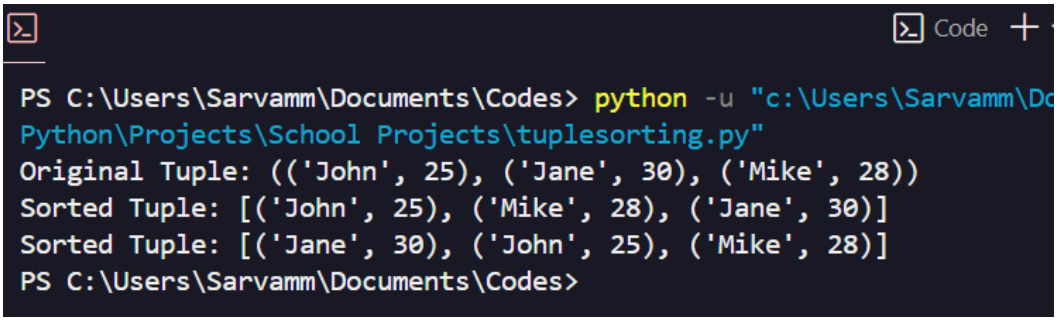
```
PS C:\Users\Sarvamm\Documents\Code> p
ython -u "c:\Users\Sarvamm\Documents\
Codes\Python\Projects\School Projects
\countsubstring.py"
Enter text
In modern society, education is far m
ore than a means to acquire knowledge
—it is a fundamental pillar that supp
orts individual growth, social equali
ty, economic development, and global
understanding. By investing in educat
ion, societies ensure a prosperous, s
table, and harmonious future where in
dividuals are empowered to contribute
 to the greater good. Whether through
 formal schooling or lifelong learnin
g, education remains an essential com
ponent of modern life, shaping the tr
ajectory of both individuals and the
world at large.
Enter word you want the count of:
is
count =  2
PS C:\Users\Sarvamm\Documents\Codes>
```

7. Tuple Sorting

Aim: To sort tuple using different methods

```
#tuple sorting
my_tuple = (('John', 25), ('Jane', 30), ('Mike', 28))
print('Original Tuple:', my_tuple)
```

```
#sorting tuple in different ways
sorted_tuple = sorted(my_tuple, key=lambda x: x[1])
print('Sorted Tuple:', sorted_tuple)

sorted_tuple2 = sorted(my_tuple, key=lambda x: x[0])
print('Sorted Tuple:', sorted_tuple2)
```



---

8. List Generation

Aim: To generate a list containing lists of multiples of numbers

```
#list generation
l=[]
for i in range(1,11):
    l.append(["Multiples of: " + str(i)])
    for j in range(1,26):
        if j % i == 0:
            l[i-1].append(j)
print(l)
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\
Sarvamm\Documents\Codes\Python\Projects\School Projects\l
istgeneration.py"
[['Multiples of: 1', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25], [
'Multiples of: 2', 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22
, 24], ['Multiples of: 3', 3, 6, 9, 12, 15, 18, 21, 24],
['Multiples of: 4', 4, 8, 12, 16, 20, 24], ['Multiples of
: 5', 5, 10, 15, 20, 25], ['Multiples of: 6', 6, 12, 18,
24], ['Multiples of: 7', 7, 14, 21], ['Multiples of: 8',
8, 16, 24], ['Multiples of: 9', 9, 18], ['Multiples of: 1
0', 10, 20]]
PS C:\Users\Sarvamm\Documents\Codes>
```

---

9. Merge Dictionaries

Aim: To merge dictionaries using different methods

```
#merge dictionary
dict1 = {'a': 1, 'b': 2, 'c': 3 , 'common': 'CommonElement' }
dict2 = {'d': 4, 'e': 5, 'f': 6, 'common': 'CommonElement'}

merged_dict = {**dict1, **dict2}
merged_dict2 = dict1 | dict2
print(merged_dict)
print(merged_dict2)
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\tempCodeRunnerFile.py"
{'a': 1, 'b': 2, 'c': 3, 'common': 'CommonElement', 'd': 4, 'e': 5, 'f': 6}
{'a': 1, 'b': 2, 'c': 3, 'common': 'CommonElement', 'd': 4, 'e': 5, 'f': 6}
PS C:\Users\Sarvamm\Documents\Codes>
```

---

10. Convert a roman numeral to an integer

Aim: To convert roman numerals to integers

```
#Convert roman numerals to numbers

def roman_to_int(s):
    roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100,
    int_val = 0
    for i in range(len(s)):
        if i > 0 and roman_dict[s[i]] > roman_dict[s[i-1]]:
            int_val += roman_dict[s[i]] - 2*roman_dict[s[i-1]
            continue
        int_val += roman_dict[s[i]]
    return int_val

#Test the function

print(roman_to_int('III'))  # Expected output: 3
print(roman_to_int('IV'))  # Expected output: 4
print(roman_to_int('IX'))  # Expected output: 9
print(roman_to_int('LVIII'))  # Expected output: 58
print(roman_to_int('MCMXCIV'))  # Expected output: 1994
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\tempCodeRunnerFile.py"
3
4
9
58
1994
PS C:\Users\Sarvamm\Documents\Codes>
```

---

11. Calculate student Grades

Aim: To calculate student grades based on their marks

```
#Calculate student grades
def calculate_grade(marks):
    if not isinstance(marks, int) or marks < 0 or marks > 100
        return "Invalid marks"
    elif marks >= 95:
        return "A+"
    elif marks >= 90:
        return "A"
    elif marks >= 80:
        return "B"
    elif marks >= 70:
        return "C"
    elif marks >= 60:
        return "D"
    elif marks < 60:
        return "F"

#Example usage
marks = int(input("Enter marks: "))
grade = calculate_grade(marks)
print(f"Grade: {grade}")
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\
Sarvamm\Documents\Codes\Python\Projects\School Projects\t
empCodeRunnerFile.py"
Enter marks: 45
Grade: F
PS C:\Users\Sarvamm\Documents\Codes>
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\
Sarvamm\Documents\Codes\Python\Projects\School Projects\t
empCodeRunnerFile.py"
Enter marks: 89
Grade: B
PS C:\Users\Sarvamm\Documents\Codes>
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\
Sarvamm\Documents\Codes\Python\Projects\School Projects\t
empCodeRunnerFile.py"
Enter marks: -99
Grade: Invalid marks
PS C:\Users\Sarvamm\Documents\Codes>
```

---

12. Create Address Book

Aim: To create an address book and fill it with random or manual entries

```
address_book = {}
def add_entry():
    name = input("Enter name: ").strip()
    address = input("Enter address: ").strip()
    try:
        phone_number = int(input("Enter phone number (10 digi
        age = int(input("Enter age (0-120): ").strip())
```

```
        if len(str(phone_number)) == 10 and 0 < age < 120:

            address_book[name] = [address, phone_number, age]


        else:
            print("Invalid phone number or age. Please try ag
    except ValueError:
        print("Invalid input. Please enter numeric values for
def showAddressBook():
    for name in address_book:
        address, phone_number, age = address_book[name]
        print(f"Name: {name}")
        print(f"Address: {address}")
        print(f"Phone Number: {phone_number}")
        print(f"Age: {age}")
        print("----------------------------")
def main():
    while True:
        print("\nAddress Book Menu:")
        print("Add a new entry")

        choice = input("Enter your choice (y/n): ").strip()
        if choice in "yY":
            add_entry()
        else:
            break
main()
showAddressBook()
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\addressbook new.py"

Address Book Menu:
Add a new entry
Enter your choice (y/n): y
Enter name: Sarvamm
Enter address: LTSU
Enter phone number (10 digits): 7007955580
Enter age (0-120): 19

Address Book Menu:
Add a new entry
Enter your choice (y/n): n
Name: Sarvamm
Address: LTSU
Phone Number: 7007955580
Age: 19
----------------------------
PS C:\Users\Sarvamm\Documents\Codes>
```

13. Implement Calculator

Aim: To create a basic calculator

```
#Calculator by Sarvamm
import math
def draw():
    global userinput, result
    l = len(userinput)
    r = len(str(result))
    print(" " + "-"*l + "-"*(16-l))
    print("|", userinput + " "*(15-l) + "|")
    print(" " + "-"*l + "-"*(16-l))
```

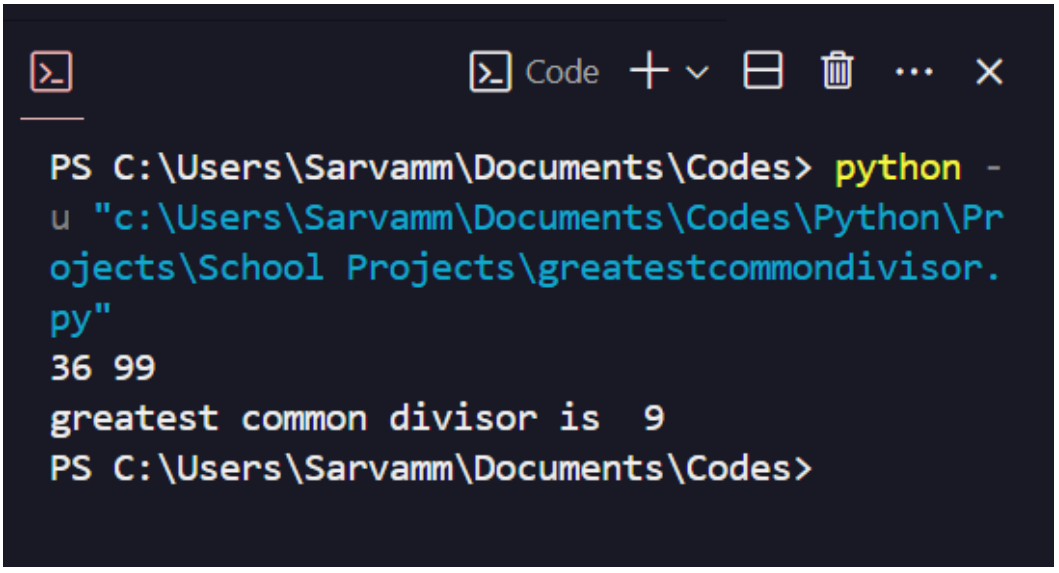
```
    print(" "*(15-r) + "= " + str(result))
    print("-----------------")
def calculate(expression):
    try:
        return eval(expression)
    except ZeroDivisionError:
        return "Error (division by zero)"
    except Exception:
        return "Error (invalid input)"

result = ""
userinput = ""
while True:
    x = input("Input: ")

    if x.lower() in ['c', 'clr', 'clear']:
        userinput = ""
        result = ""
        print("cleared.")
    elif x == "=":
        result = calculate(userinput)
    else:
        userinput = str(result) + x
        result = calculate(userinput)

    draw()
    if result in ["Error (division by zero)", "Error (invalid
        userinput = ""
        result = ""
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\calculator.py"
Input: 99+11
 ----------------
| 99+11          |
 ----------------
            = 110
-----------------
Input: /10
 ----------------
| 110/10         |
 ----------------
           = 11.0
-----------------
Input: **2
 ----------------
| 11.0**2        |
 ----------------
          = 121.0
-----------------
Input: c
cleared.
 ----------------
|                |
 ----------------
               =
-----------------
Input: 69
 ----------------
| 69             |
 ----------------
              = 69
```

14. Greatest Common Divisor

Aim: To find the greatest common divisor of any two random numbers

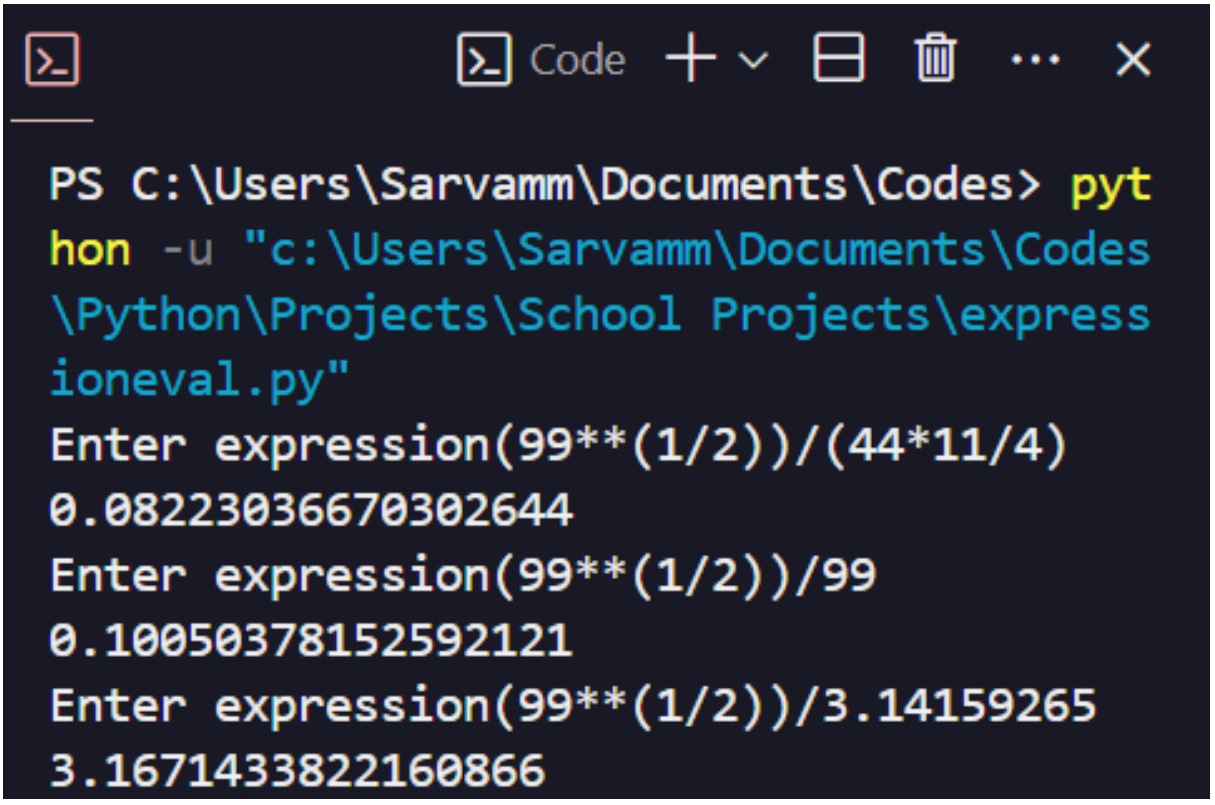
```
#greatest common divisor
import random
num1 = random.randint(0,50)
num2 = random.randint(50,100)
print(num1, num2)

for i in range(num1, 0, -1):
    if (num1%i == 0 and num2%i==0):
        c=i
        break
print("greatest common divisor is ",c)
```



15. Expression Evaluation

Aim: Basic expression evaluation using eval() function

```
#Expression evaluation:
while True:
    x = input("Enter expression")
    print(eval(x))
```

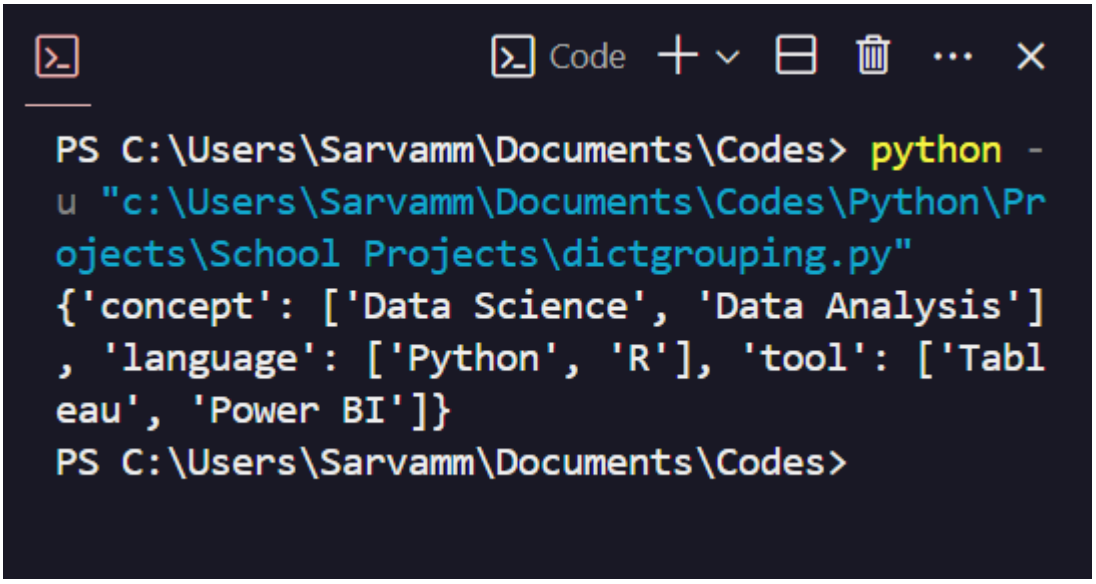


16. Dictionary Grouping

Aim: Grouping dictionaries with similar types into a new dictionary

```
items = [
    {"name": "Data Science", "type": "concept"},
    {"name": "Data Analysis", "type": "concept"},
    {"name": "Python", "type": "language"},
    {"name": "R", "type": "language"},
    {"name": "Tableau", "type": "tool"},
    {"name": "Power BI", "type": "tool"},
]
```

```
# Grouping items by type using a for loop
grouped = {}
for i in items:
    item_type = items[i]["type"]
    if item_type not in grouped:
        grouped[item_type] = []
    grouped[item_type].append(items[i]["name"])

print(grouped)
```



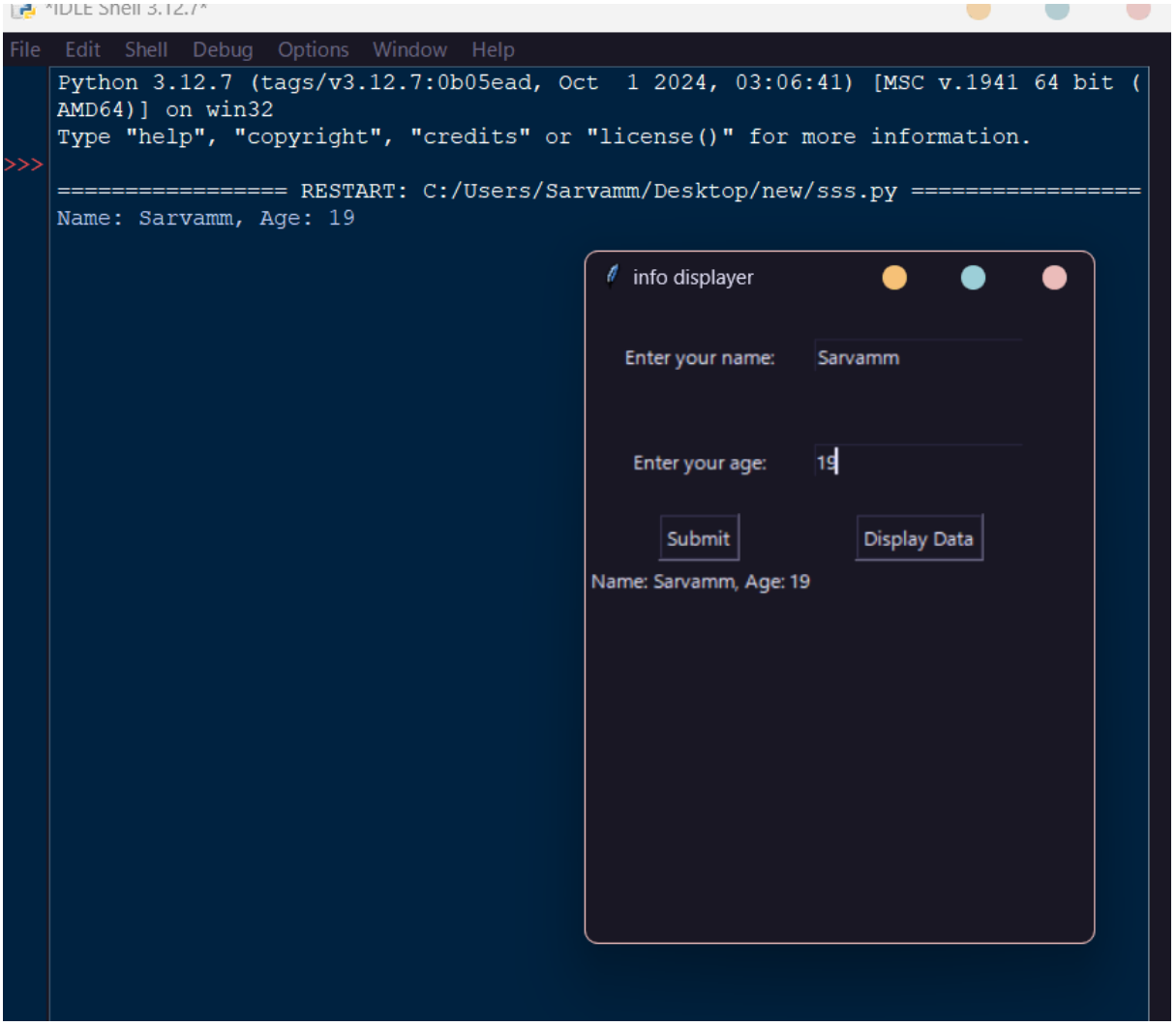
---

17. GUI with TKinter

Aim: To create a window that prompts user to input name and age and then displays it

```
import tkinter as tk
window = tk.Tk()
window.title("info displayer")
window.geometry("300x400")


label1 = tk.Label(window, text="Enter your name:")
label1.grid(row=0, column=0, pady = 20)
```

```
name_entry = tk.Entry(window)
name_entry.grid(row=0, column=1)

label2 = tk.Label(window, text="Enter your age:")
label2.grid(row=1, column=0, pady = 20)

age_entry = tk.Entry(window)
age_entry.grid(row=1, column=1)


submit_button = tk.Button(window, text="Submit", command=lamb
submit_button.grid(row=2, columnspan = 1)

def submit_data():
    name = name_entry.get()
    age = age_entry.get()
    print(f"Name: {name}, Age: {age}")
    name_entry.delete(0, tk.END)
    age_entry.delete(0, tk.END)
def displayData():
    name = name_entry.get()
    age = age_entry.get()
    display_label = tk.Label(window, text=f"Name: {name}, Age
    display_label.grid(row=3, column=0)

display_button = tk.Button(window, text="Display Data", comma
display_button.grid(row=2, column=1)

window.mainloop()
```


IDLE Shell 3.12.7
File Edit Shell Debug Options Window Help
Python 3.12.7 (tags/v3.12.7:0b05ead, Oct 1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/Sarvamm/Desktop/new/sss.py ==========
Name: Sarvamm, Age: 19
info displayer
Enter your name:
Sarvamm
Enter your age:
19
Submit
Display Data
Name: Sarvamm, Age: 19