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

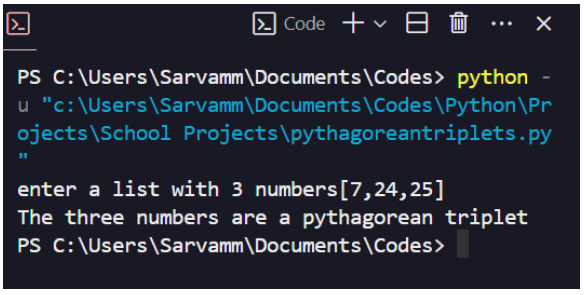
30	Binomial Expansion
31	Quadratic roots
32	Top 10 word count
33	Pointer
34	Number Game
35	Fibonacci
36	Calculator with pointer
37	Pattern printing
38	Finance calculator
39	Expense Splitter
40	Notepad

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")
```

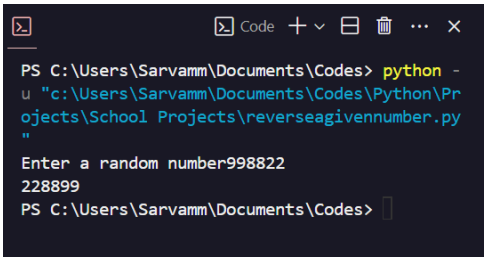


---

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)
```

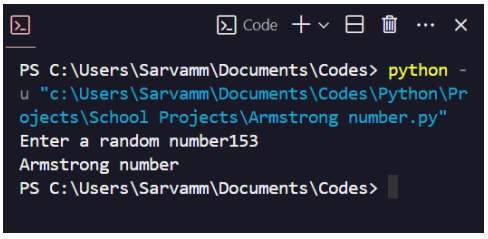


---

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")
```

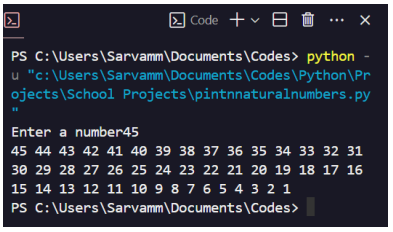


---

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> python -u "c:\Users\Sarvamm\Documents\Codes\Python\Projects\School Projects\countsubstring.py"
Enter text
In modern society, education is far more than a means to acquire knowledge—it is a fundamental pillar that supports individual growth, social equality, economic development, and global understanding. By investing in education, societies ensure a prosperous, stable, and harmonious future where individuals are empowered to contribute to the greater good. Whether through formal schooling or lifelong learning, education remains an essential component of modern life, shaping the trajectory 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)
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm\Do
Python\Projects\School Projects\tuplesorting.py"
Original Tuple: (('John', 25), ('Jane', 30), ('Mike', 28))
Sorted Tuple: [('John', 25), ('Mike', 28), ('Jane', 30)]
Sorted Tuple: [('Jane', 30), ('John', 25), ('Mike', 28)]
PS C:\Users\Sarvamm\Documents\Codes>
```

---

8. List Generation

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

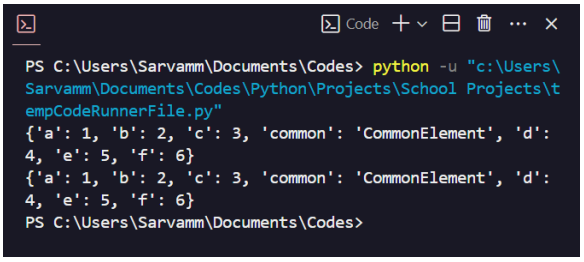
```
#list generation
l=[]
for i in range(1,11):
    l.append(["Multiples of: " + str(i)])
    for j in range(1,26):
        if j % i == 0:
            l[i-1].append(j)
print(l)
```

```
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)
```

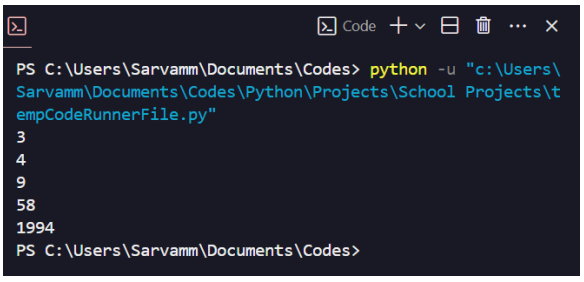


10. Convert a roman numeral to an integer

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
```



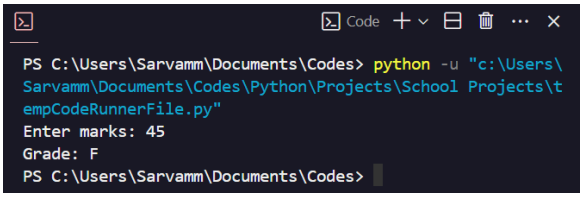
---

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}")
```



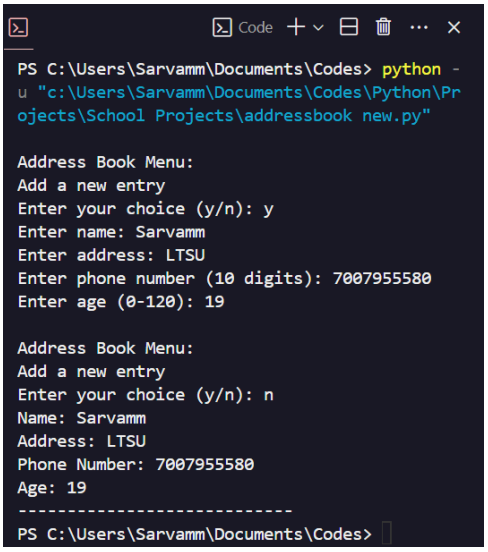
---

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()
```



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 = ""
```

```
Input: 99+11
 ----------------
| 99+11          |
 ----------------
             = 110
-----------------
Input: /10
 ----------------
| 110/10         |
 ----------------
            = 11.0
-----------------
```

---

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)
```

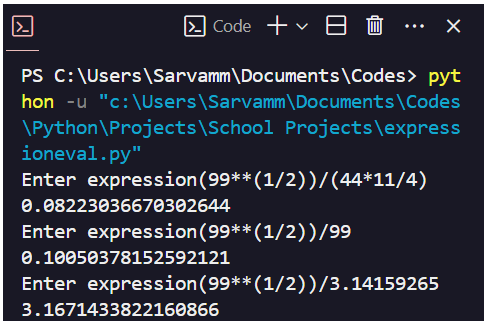
```
PS C:\Users\Sarvamm\Documents\Codes> python -
u "c:\Users\Sarvamm\Documents\Codes\Python\Pr
ojects\School Projects\greatestcommondivisor.
py"
36 99
greatest common divisor is  9
PS C:\Users\Sarvamm\Documents\Codes>
```

---

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: To group a list of items by their "type" and organize them into a 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 item in items:
    item_type = item["type"]  # Corrected this line to access
    if item_type not in grouped:
        grouped[item_type] = []
    grouped[item_type].append(item["name"])

print(grouped)
```

```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm
\Documents\Codes\Python\School Projects\dictgrouping.py"
{'concept': ['Data Science', 'Data Analysis'], 'language': ['Pyt
hon', 'R'], 'tool': ['Tableau', 'Power BI']}
PS C:\Users\Sarvamm\Documents\Codes>
```

17. Machine Value conversion

Aim: A program that can convert any decimal number to any base

```
def d2a(x, base):

    if 0 < int(base) <= 36:
        l = []
        divi = int(x)
        oppo = int(base)
        while divi != 0:
            l.append(divi % oppo)
            divi = divi // oppo
        r = ""
        for i in l[::-1]:
            if i >= 10:
                r += chr(i + 55)  # ASCII value of A=65, B=66
            else:
                r += str(i)
        return r
    else:
        print("invalid base")
        return
print(d2a(10,8))
```

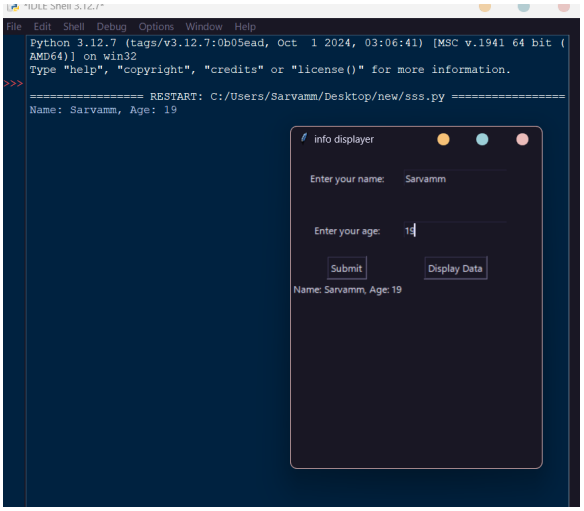
```
PS C:\Users\Sarvamm\Documents\Codes> python -u "c:\Users\Sarvamm
\Documents\Codes\Python\School Projects\machinevalue.py"
The number 16 is equivalent to 10000 in base 2.
PS C:\Users\Sarvamm\Documents\Codes>
```

18. 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()
```



---

19. Calculator GUI

Aim: Making a calculator using tkinter

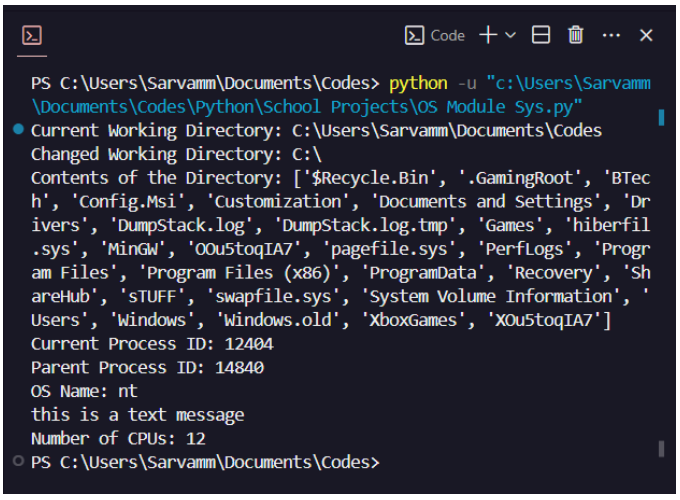
```
#making a calculator gui with number buttons and operator but

import tkinter as tk
from tkinter import ttk

def click(button_text):
    if button_text == "=":
        try:
            result = eval(entry.get())
            entry.delete(0, tk.END)
            entry.insert(tk.END, str(result))
        except Exception as e:
            entry.delete(0, tk.END)
            entry.insert(tk.END, "Error")
    elif button_text == "C":
        entry.delete(0, tk.END)
```

```
    else:
        entry.insert(tk.END, button_text)

root = tk.Tk()
root.title("Calculator")
root.geometry("300x400")
root.resizable(False, False)

entry = ttk.Entry(root, justify="right")
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10,)

buttons = [
    ("7", "8", "9", "/"),
    ("4", "5", "6", "*"),
    ("1", "2", "3", "-"),
    ("C", "0", "=", "+")
]
print(dict(enumerate(buttons, 1)))

for r, row in enumerate(buttons, 1):
    for c, btn_text in enumerate(row):
        btn = ttk.Button(root, text=btn_text, command=lambda
        btn.grid(row=r, column=c, padx=5, pady=5, sticky="nse
for i in range(5):
    root.grid_rowconfigure(i, weight=1)
    root.grid_columnconfigure(i, weight=1)

root.mainloop()
```

20. OS Module system services

Aim: Demonstrating use of system methods in os module

```
import os

def main() -> None:
    print("Current Working Directory:", os.getcwd())
    os.chdir("C:/")
    print("Changed Working Directory:", os.getcwd())
    print("Contents of the Directory:", os.listdir())
    print("Current Process ID:", os.getpid())
    print("Parent Process ID:", os.getppid())
    print("OS Name:", os.name)
    os.system("echo this is a text message")
    print("Number of CPUs:", os.cpu_count())

main()
```

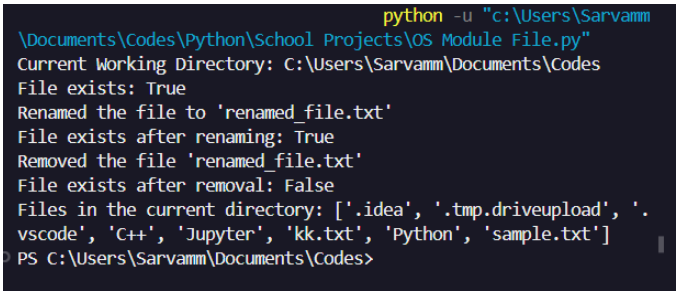


21. OS Module File Services

Aim: Demonstrating use of file methods in os module

```
#os module file services
import os
```

```
def main() -> None:
    print("Current Working Directory:", os.getcwd())
    with open('example_file.txt', 'w') as f:
        f.write("This is an example file.")
    print("File exists:", os.path.exists('example_file.txt'))
    os.rename('example_file.txt', 'renamed_file.txt')
    print("Renamed the file to 'renamed_file.txt'")
    print("File exists after renaming:", os.path.exists('rena
    os.remove('renamed_file.txt')
    print("Removed the file 'renamed_file.txt'")
    print("File exists after removal:", os.path.exists('renam
    print("Files in the current directory:", os.listdir())

main()
```



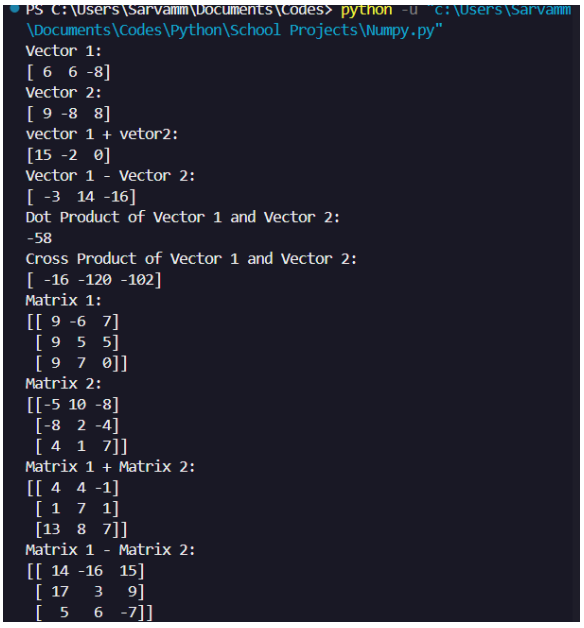
---

22. Numpy Operations

Aim: Performing operations using numpy

```
import numpy as np
import random as rd
def random2darray(r,c):
    array = np.array([[rd.randint(-10,10) for i in range(c)]
    return array
def random3dvector():
    vector = np.array([rd.randint(-10,10) for i in range(3)])
    return vector

v1 = random3dvector()
v2 = random3dvector()
```

```
print(f'Vector 1: \n{v1}')
print(f'Vector 2: \n{v2}')
print(f'vector 1 + vetor2: \n{v1 + v2}')
print(f'Vector 1 - Vector 2: \n{v1 - v2}')
print(f'Dot Product of Vector 1 and Vector 2: \n{np.dot(v1,v2
print(f'Cross Product of Vector 1 and Vector 2: \n{np.cross(v

m1 = random2darray(3,3)
m2 = random2darray(3,3)

print(f'Matrix 1: \n{m1}')
print(f'Matrix 2: \n{m2}')
print(f'Matrix 1 + Matrix 2: \n{m1 + m2}')
print(f'Matrix 1 - Matrix 2: \n{m1 - m2}')
print(f'Matrix 1 x Matrix 2: \n{np.dot(m1,m2)}')
print(f'Matrix 1 Transpose: \n{m1.T}')
print(f'8 * Matrix 1  Transpose: \n{8*m1}')
```

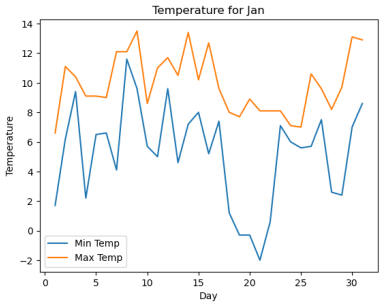


23. Charts matplotlib

Aim: Making charts using matplotlib library

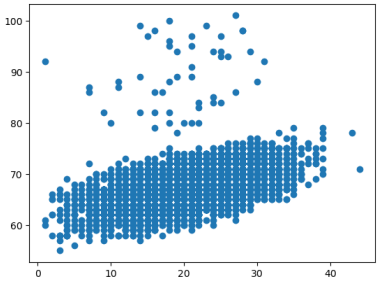
```
monthnames = df['month_name'].unique()
max_temps = {}
```

```
min_temps = {}
for i in monthnames:
    dk = df[df['month_name'] == i]
    max_temps[i] = np.max(dk['max_temp'])
    min_temps[i] = np.min(dk['min_temp'])
    plt.plot(dk['day'], dk['min_temp'], label='Min Temp')
    plt.plot(dk['day'], dk['max_temp'], label='Max Temp')
    plt.title(f"Temperature for {i}")
    plt.xlabel('Day')
    plt.ylabel('Temperature')
    plt.legend()
    plt.show()
```

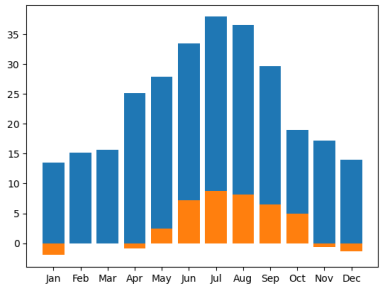


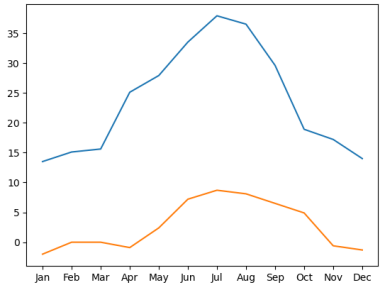

```
monthnames = df['month_na
max_temps = {}
min_temps = {}
for i in monthnames:
    dk = df[df['month_nai
    max_temps[i] = np.ma
    min_temps[i] = np.mi
months= list(max_temps.k
max = list(max_temps.val
min = list(min_temps.val
datadict= {}
datadict['Month_Name'] =
datadict['Max_Temp'] = m
datadict['Min_Temp'] = m
tempData = pd.DataFrame(
plt.bar('Month_Name', 'M
```

```
df2 = pd.read_csv("C:/Users/Sarv
plt.scatter('Hours_Studied','Exa
plt.title("relation of study hou
plt.xlabel("hrs studies in a wee
plt.ylabel("marks scored")
plt.show()
```

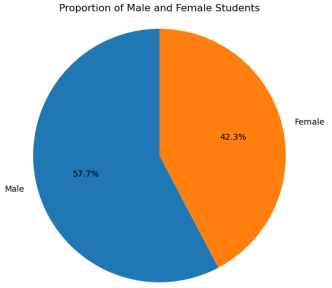


```
plt.bar('Month_Name', 'M
plt.show()
```






```
gender_counts = df2['Gender'].va

plt.figure(figsize=(8, 6))
plt.pie(gender_counts, labels=ge
plt.title('Proportion of Male an
plt.axis('equal')  # Equal aspec
plt.show()
```



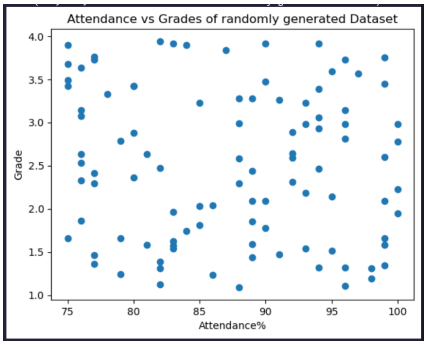
---

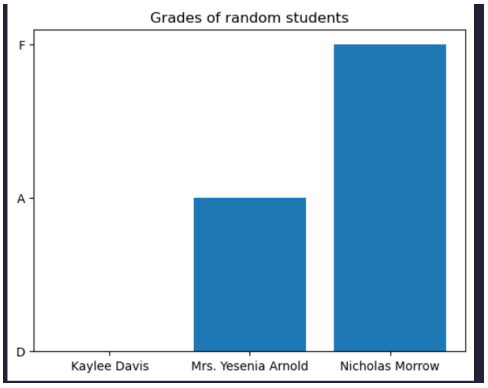
24. File Operations on Excel

Aim: Performing basic analysis on an excel database

```
path = "students_data.xlsx"
df = pd.read_excel(path)
print("successfuly imported")
df.describe()
df.shape
df.head()
df.columns
df.info()
df.isnull().sum()
name = df['Name']
name
name1 = df[['Name']]
name1
plt.scatter('Attendance', 'GPA', data = df)
plt.xlabel('Attendance%')
plt.ylabel('Grade')
```

```
plt.title('Attendance vs Grades of randomly generated Dataset

plt.bar('Name', 'Grade', data = df.iloc[rd.randint(0,2):rd.ra
plt.title('Grades of random students')
plt.figure(figsize=(50,5))
plt.show()

df.iloc[23:28, 4:12]
df.loc[23:28,'Roll Number':'Attendance']

grouped  =  df.groupby('Grade')['GPA'].count()
grouped
```





25. Approximation PI using Gregory Leibniz series

Aim: Using Gregory Leibniz series to approximate pi

```
#Approximating pi using Gregory-Leibniz series

n = int(input("Enter precision. (Higher is better but slower,
pi = 0


k, j = 1, 0
for i in range(1 ,n+1):
    pi += (4/k)*float((-1)**(j))
    k += 2
```

```
    if j == 0:
        j = 1
    else:
        j = 0

print(pi)
```

```
Enter precision. (Higher is better but slower, less than 100 is
inaccurate)
9999
3.1416926635905345
PS C:\Users\Sarvamm\Documents\Codes>
```

26. Approximating Pi using Nilkantha Series

Aim: Using Nilkantha series to approximate pi

```
#Approximating pi using Nilkantha Series
n = int(input("Enter precision (Higher is better, n = 100 wil

pi = 3
k,j = 2, 0
for i in range(1,n+1):
    pi += 4*( (-1)**j/(k*(k+1)*(k+2)) )
    k += 2
    if j == 0:
        j = 1
    else:
        j = 0
print(pi)
print(float(3))
```

```
Enter precision (Higher is better, n = 1
00 will be accurate to 6 decimal places
9999
3.1415926535900383
PS C:\Users\Sarvamm\Documents\Codes>
```

27. Approximating Pi using trigonometry

Aim: Using sin function to approximate pi

```
import math
n = int(input("enter a large number (5-7 digits)"))
# Convert degrees to radians
angle_degrees = 180
angle_radians = math.radians(angle_degrees)
from decimal import Decimal, getcontext

# Set precision to 3 decimal places
pi = Decimal((math.sin(angle_radians/99999))*99999)
print(pi)
```

```
enter a large number (5-7 digits)999999
3.14159265307301138747675395279657095670
7000732421875
```

28. Tic Tac Toe Game

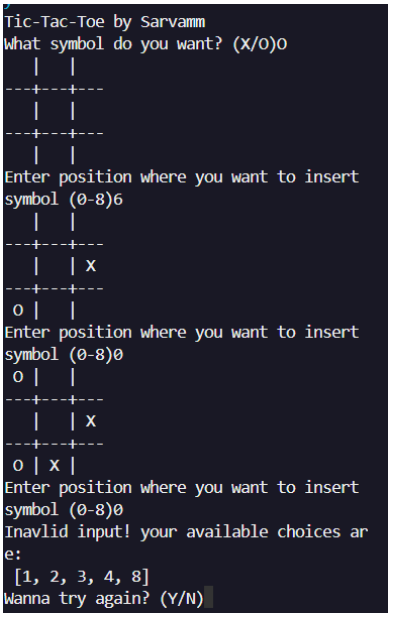
Aim: To build a tic tac toe game using python

```
#Created by Sarvamm
#Working Tic-Tac-Toe game using a one dimensional array
import random
cSym = ""
uSym = ""
matrix = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
avchoices = [0, 1, 2, 3, 4, 5, 6, 7, 8]  #Available position

#drawing the game
def draw():
    print(f" {matrix[0]} | {matrix[1]} | {matrix[2]} ")
    print("---+---+---")
    print(f" {matrix[3]} | {matrix[4]} | {matrix[5]} ")
    print("---+---+---")
    print(f" {matrix[6]} | {matrix[7]} | {matrix[8]} ")

#user selects his preferred symbol
def XorO():
```

```
    global uSym, cSym
    symbol = input("What symbol do you want? (X/O)")
    if symbol in "xX":
        uSym = "X"
        cSym = "O"
    elif symbol in "oO":
        uSym = "O"
        cSym = "X"
    else:
        print("Invalid input!, Let's try that again.")
        XorO()
#Checking win status
def check():
    win_conditions = [
        [0, 1, 2], [3, 4, 5], [6, 7, 8],  # Rows
        [0, 3, 6], [1, 4, 7], [2, 5, 8],  # Columns
        [0, 4, 8], [2, 4, 6]              # Diagonals
    ]

    for k in win_conditions:
        if matrix[k[0]] == matrix[k[1]] == matrix[k[2]]:
            if matrix[k[0]] == cSym:
                return "Computer Won"
            elif matrix[k[0]] == uSym:
                return "Player Won"

    if " " not in matrix:
        return "Tie"

#Try again wihout starting over
def TryAgain():
    ch = input(("Wanna try again? (Y/N)"))
    if ch in "Yy":
        game()

#Start a new game
def PlayAgain():
    global matrix, avchoices
```

```
    ch = input(("Wanna play again? (Y/N)"))
    if ch in "Yy":
        matrix = [" ", " ", " ", " ", " ", " ", " ", " ", " "
        avchoices = [0, 1, 2, 3, 4, 5, 6, 7, 8]
        game()
    else:
        print("Thanks for Playing")
        matrix = [" ", " ", " ", " ", " ", " ", " ", " ", " "
        print("Game Ended")


def checkResult():
    op = check()
    if op == "Player Won":
        draw()
        print("You Won! Congratulations!")
        return True
    elif op == "Computer Won":
        draw()
        print("You Lose!")
        return True
    elif op == "Tie":
        draw()
        print("Game Tied.")
        return True
    else:
        return False
#Main
print("Tic-Tac-Toe by Sarvamm")
XorO()
def game():
    global matrix, avchoices
    while len(avchoices) > 0:
        draw()

        #Player's Turn
        try:
            userCh = int(input("Enter position where you want
```

```
            if userCh in avchoices:
                matrix[userCh] = uSym
                avchoices.remove(userCh)
                if checkResult():
                    PlayAgain()
                    break

        #Computer's Turn
                comCh = random.choice(avchoices)
                matrix[comCh] = cSym
                avchoices.remove(comCh)
                if checkResult():
                    PlayAgain()
                    break
            else:
                print("Inavlid input! your available choices 
                TryAgain()
                break
        except ValueError:
            print("Inavlid input! your available choices are:
            TryAgain()
game() #Starting game
```



29. Conversion of number system

Aim: To convert a number from any base to any other base (2-36)

```
def checkbase(x , base):
    x = str(x)
    base = int(base)
    global l
    l=[]
    for i in x:
        if i.lower() in "abcdefghij":
            i = i.upper()
            l.append(int(chr(ord(i) - 17)) + 10)


        elif i.lower() in "klmnopqrst":
            i = i.upper()
            l.append(int(chr(ord(i) - 27)) + 20)


        elif i.lower() in "uvwxyz":
            i = i.upper()
            l.append(int(chr(ord(i) - 37)) + 30)

        else:
            l.append(i)
    for i in l:
        if type(i) != int:
            l[l.index(i)] = int(i)

    if max(l) < base:
        return True
    else:
        print("Invalid input, please check base and try again
        return False

 #Checking validity of given input
def isValid(x,base):
    for char in x:
        if not (char.isdigit() or char.lower() in 'abcdefghij
```

```
            print("Invalid entry, only alphanumeric allowed")
            return False
        elif base > 36 or base < 2:
            print("Invalid base (2-36 allowed)")
            return False
    return True

#Any number into Decimal --------------------------------
def a2d(x, base):
    x = str(x)
    base = int(base)

    if isValid(x,base):
        if checkbase(x, base):
            global l
            r=0
            hp = len(l) - 1
            for i in l:
                r += int(i)*(base**hp)
                hp -= 1
            return r
#Decimal number into given base -----------------------------
def d2a(x, base):

    if 0 < int(base) <= 36:
        l = []
        divi = int(x)
        oppo = int(base)
        while divi != 0:
            l.append(divi % oppo)
            divi = divi // oppo
        r = ""
        for i in l[::-1]:
            if i >= 10:
                r += chr(i + 55)  # ASCII value of A=65, B=66
            else:
                r += str(i)
        return r
```

```
    else:
        print("invalid base")
        return

#main function --------------------------------
def main(x, base1, base2):
    inDecimal = a2d(x, base1)
    inBase = d2a(inDecimal, base2)
    print(f"The number {x} is equivalent to {inBase} in base
    return inBase

#Testing the program
main("16aF2", 16, 36) #Number give, Base of number, Base to b
```

```
The number 16aF2 is equivalent to 1ZOY i
n base 36.
PS C:\Users\Sarvamm\Documents\Codes>
```

30. Binomial Expansion

Aim: To expand a binomial

```
def expand(x, y, n):
    x = str(x)
    y = str(y)
    print("(", x, "+", y, ")^", n, " =")
    for i in range(n+1):
        T = str(n) + "C" + str(i) + ". " + x + "^" + str(n -
        if i < n:
            print(T, end=" + \n")
        else:
            print(T)

def fct(n):
    r = 1
    while n > 0:
        r = r * n
        n -= 1
```

```
    return r

def C(n, r):
    if r <= n:
        return fct(n) / (fct(n-r) * fct(r))
    else:
        return "Error in r"

def bino(x, y, n):
    if isinstance(x, str) and isinstance(y, str):
        print("boing")
        for i in range(n+1):
            T = str(C(n, i)) + " . " + x + "^" + str(n-i) + "
            if i < n:
                print(T, end=" + \n")
            else:
                print(T)
    elif isinstance(x, int) and isinstance(y, str):
        for i in range(n+1):
            T = str(C(n, i) * (x**(n-i))) + " ." + y + "^" +
            if i < n:
                print(T, end=" + \n")
            else:
                print(T)
    elif isinstance(x, str) and isinstance(y, int):
        for i in range(n+1):
            T = str(C(n, i) * (y**i)) + " ." + x + "^" + str(
            if i < n:
                print(T, end=" + \n")
            else:
                print(T)
    elif isinstance(x, int) and isinstance(y, int):
        print((x + y) ** n)
    else:
        print("Error in input")

def main():
    print("For a binomial in the form of: \n (x+y)^n,")
```

```
    x_input = input("Enter x, (int or char)\n")
    y_input = input("Enter y, (int or char)\n")
    try:
        x = int(x_input)
    except ValueError:
        x = x_input

    try:
        y = int(y_input)
    except ValueError:
        y = y_input

    n = int(input("Enter n, a natural number\n"))

    if n < 0:
        return("Invalid value of n")

    elif (isinstance(x, (int, str)) and isinstance(y, (int, s
        expand(x, y, n)
        print("Which equates to ")
        bino(x, y, n)
        ch = input("Run again? (Y/N): ")
        if ch in "Yy":
            main()
        else:
            return
    else:
        print("Input error")

main()
```

```
For a binomial in the form of:
 (x+y)^n,
Enter x, (int or char)
x
Enter y, (int or char)
2
Enter n, a natural number
4
( x + 2 )^ 4  =
4C0. x^4. 2^0 +
4C1. x^3. 2^1 +
4C2. x^2. 2^2 +
4C3. x^1. 2^3 +
4C4. x^0. 2^4
Which equates to
1.0 .x^4 +
8.0 .x^3 +
24.0 .x^2 +
32.0 .x^1 +
16.0 .x^0
```

31. Solving Quadratic Roots

Aim: Solving Quadratic roots using python

```
import math

a = int(input("coefficient of x^2:\n"))
b = int(input("coefficient of x:\n"))
c = int(input("constant term:\n"))

def findRoots(a = 1, b = 1, c = 1):
    if a == 0:
        print("coeffecient of x^2 must be non zero")
        return
    discriminant = (b**2) - (4*a*c)

    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        print(root1, root2)

    elif discriminant == 0:
        root = -b / (2*a)
        print("Root is", root)
    else:
        print("No real roots")
```

```
        return
findRoots(a,b,c)
```

```
coefficient of x^2:
1
coefficient of x:
3
constant term:
2
-1.0 -2.0
PS C:\Users\Sarvamm\Documents\Codes>
```

32. Top 10 Word counts

Aim: Find the top 10 most occurring words in a paragraph

```
paragraph = input("Enter a paragraph: ")
words = paragraph.split()
word_count = {}

for word in words:
    word = word.lower().strip(".,!?;:")
    word_count[word] = word_count.get(word, 0) + 1

sorted_words = sorted(word_count.items(), key=lambda x: x[1],
top_10 = sorted_words[:10]

for word, count in top_10:
    print(word, count)
```

```
Enter a paragraph: this this this is is
a para para
this 3
is 2
para 2
a 1
```

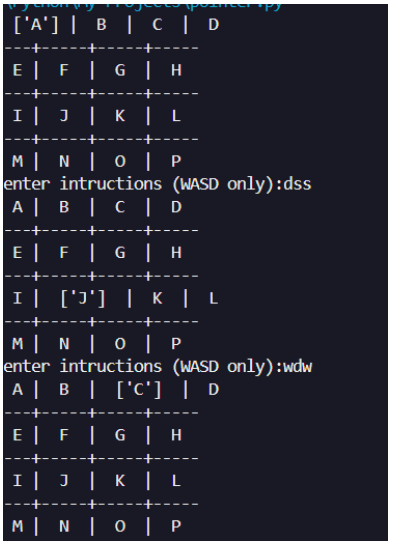
33. Pointer

Aim: Making a pointer that can move in a grid

```
matrix = [['A'], 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
```

```
# Draw the expanded 4x4 grid
def draw():
    print(f" {matrix[0]} |  {matrix[1]}  |  {matrix[2]}  |  {
    print("---+-----+-----+-----")
    print(f" {matrix[4]} |  {matrix[5]}  |  {matrix[6]}  |  {
    print("---+-----+-----+-----")
    print(f" {matrix[8]} |  {matrix[9]}  |  {matrix[10]}  |
    print("---+-----+-----+-----")
    print(f" {matrix[12]} |  {matrix[13]}  |  {matrix[14]}  |

draw()

def findPointer():
    for i in matrix:
        if type(i) == list:
            return matrix.index(i)
def remPointer():
            loc = findPointer()
            matrix[loc] = matrix[loc][0]
def movert():
    loc = findPointer() + 1
    if loc > len(matrix) - 1:
        loc = loc - len(matrix)
    remPointer()
    matrix[loc] = [matrix[loc]]
def movelt():
    loc = findPointer() - 1
    remPointer()
    matrix[loc] = [matrix[loc]]

def moveup():
    loc = findPointer() - 4
    remPointer()
    matrix[loc] = [matrix[loc]]

def movedwn():
    loc = findPointer() + 4
```

```
    if loc > len(matrix) - 4:
        loc = loc - len(matrix)
    remPointer()
    matrix[loc] = [matrix[loc]]

def main():
    while True:
        usin = input("enter intructions (WASD only):")
        if all(char in 'wasdWASD' for char in usin):
            for char in usin:
                if char in 'dD':
                    movert()
                elif char in 'aA':
                    movelt()
                elif char in 'wW':
                    moveup()
                elif char in 'sS':
                    movedwn()
            draw()
main()
```

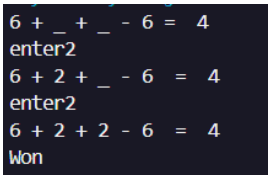


34. Number Game

Aim: Making a fill in the blank number game

```
import random
nums = [1,2,3,4,5,6,7,8,9]
ops = ["*", "/", "+", "-"]

def gencomb(k):
    return list(product(range(10), repeat=k))

# Example usage

def genq1():
    global nums, ops
    Q = ''
    expression = str(random.choice(nums))
    for _ in range(random.randint(2, 4)):
        expression += ' ' + random.choice(ops) + ' ' + str(ra
    result = eval(expression)
    if type(result) == int:

        for i in expression:
            if i.isdigit():
                Q += random.choice(['_',i])
            else:
                Q += i
        print(Q, '= ', result)
        return Q, result
    else:
        return genq1()
def uinput():
    t = genq1()
    Q, result = t[0], t[1]
    Qla = Q.split()
    for i in range(Qla.count('_')):
        a=str(int(input("enter")))
        Qla[Qla.index('_')] = a
        Q=''
        for i in Qla:
```

```
            Q+=i+' '
        print(Q, '= ', result)
    return (Q, result, Qla)

Q, result , Qla = uinput()
def won():

    global Q, result, Qla
    if eval(Q) == result:
        print('Won')
        return True
    else:
        print('Lost')
        return False
won()
```

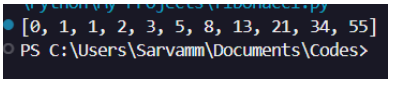


---

35. Fibonacci

Aim: To print n elements of the Fibonacci sequence

```
#code that generates fibonacci sequence

def fibonacci(n):
    fib_seq = [0, 1]
    while len(fib_seq) <= n:
        fib_seq.append(fib_seq[-1] + fib_seq[-2])
    return fib_seq

print(fibonacci(10)) #prints the first 10 numbers in the fibo
```



---

36. Calculator with Pointer

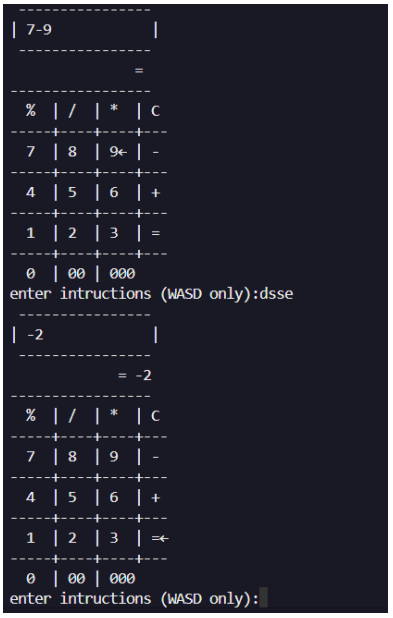
Aim: To make the user select instructions based on screen to use the calculator

```
import random
matrix = ['%', '/', '*', 'C', '7←', '8', '9', '-', '4', '5',
sym = "←"
result = ""
userinput = ""
def display_controls():
    print("=== Calculator Controls ===")
    print("Navigation:")
    print("W - Move Up")
    print("A - Move Left")
    print("S - Move Down")
    print("D - Move Right")
    print("E - Select Current Symbol")
    print("C - Clear Input")
    print("Q - Quit")
    print("H - to show controls")
    print("============================")
display_controls()
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("-----------------")
    print(" ",
        matrix[0] + ((2 - len(matrix[0])) * " "), "|",
        matrix[1] + ((2 - len(matrix[1])) * " "), "|",
        matrix[2] + ((2 - len(matrix[2])) * " "), "|",
        matrix[3] + ((2 - len(matrix[3])) * " ")
    )
```

```
    print("-----+----+----+---")
    print(" ",matrix[4]+ ((2 - len(matrix[4])) * " "), "|", m
           matrix[6]+ ((2 - len(matrix[6])) * " "), "|", matr
    print("-----+----+----+---")
    print(" ",matrix[8]+ ((2 - len(matrix[8])) * " "), "|", m
           matrix[10]+ ((2 - len(matrix[10])) * " "), "|", ma
    print("-----+----+----+---")
    print(" ",matrix[12]+ ((2 - len(matrix[12])) * " "), "|",
           "|", matrix[14]+ ((2 - len(matrix[14])) * " "), "|
    print("-----+----+----+---")
    print(" ",matrix[16]+ ((2 - len(matrix[16])) * " "), "|",
           matrix[18])


draw()

def calculate(expression):
    try:
        return eval(expression)
    except ZeroDivisionError:
        return "Error (division by zero)"
    except Exception:
        return "Error (invalid input)"

def findPointer():
    for i in matrix:
        if i[-1] == sym:
            return matrix.index(i)
def remPointer():
            loc = findPointer()
            matrix[loc] = matrix[loc].rstrip(sym)
def movert():
    loc = findPointer() + 1
    if loc > len(matrix) - 1:
        loc = loc - len(matrix)
    remPointer()
    matrix[loc] = matrix[loc] +sym
def movelt():
```

```
    loc = findPointer() - 1
    remPointer()
    matrix[loc] = matrix[loc] + sym

def moveup():
    loc = findPointer() - 4
    remPointer()
    matrix[loc] = matrix[loc]+ sym

def movedwn():
    loc = findPointer() + 4
    if loc > len(matrix) - 4:
        loc = loc - len(matrix)
    remPointer()
    matrix[loc] = matrix[loc] + sym

def main():
    global userinput, result

    while True:
        usin = input("enter intructions (WASD only):")
        if all(char in 'wasdWASDeEqQhH' for char in usin):
            for char in usin:
                if char in 'qQ':
                    return
                elif char in 'hH':
                    display_controls()
                elif char in 'dD':
                    movert()
                elif char in 'aA':
                    movelt()
                elif char in 'wW':
                    moveup()
                elif char in 'sS':
                    movedwn()
                elif char in 'eE':
                    if matrix[findPointer()][0] == "=":
                        result = calculate(userinput)
```

```
                        userinput = str(result)
                    elif matrix[findPointer()][0] == "C":
                        userinput = ""
                        result = ""
                    else:
                        userinput = str(userinput) + matrix[f
            draw()
            if result in ["Error (division by zero)", "Error
                userinput = ""
                result = ""
        else:
            print("Invalid input!")
            main()
main()
```



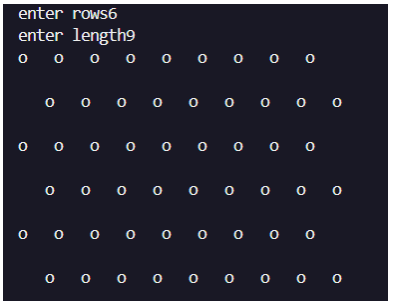
---

37. Pattern

Aim: To print a pattern based on user input

```
m = int(input("enter rows"))
n = int(input("enter length"))
```

```
for i in range(1,m+1):
    for j in range(1,n+1):
        if i % 2 == 1:
            print("o   ",end="")
        else:
            print("   o",end="")
    print("\n")
```



---

38. Finance Calculator

Aim: calculate finances based on user input

```
def calculate_finances(monthly_income: float, tax_rate: float
    yearly_salary: float = monthly_income*12
    monthly_tax: float = monthly_income*(tax_rate/100)
    yearly_tax: float = monthly_tax*12
    monthly_net_income: float = monthly_income - monthly_tax
    yearly_net_income: float = monthly_net_income*12

    print('-----------------------------------------------------
    print(f'Monthly Income: {monthly_income:,.2f} {currency}'
    print(f'Tax Rate: {tax_rate}%')
    print(f'Yearly Salary: {yearly_salary:,.2f} {currency}')
    print(f'Monthly Tax: {monthly_tax:,.2f} {currency}')
    print(f'Yearly Tax: {yearly_tax:,.2f} {currency}')
    print(f'Monthly Net Income: {monthly_net_income:,.2f} {cu
    print(f'Yearly Net Income: {yearly_net_income:,.2f} {curr
    print('-----------------------------------------------------
```

```
def main() -> None:
    while True:
        try:
            monthly_input: float = float(input('Monthly salar
            tax_rate_input: float = float(input('Tax rate (%)
            other_expenses_input: float = float(input('Other
            break
        except ValueError:
            print('Invalid input. Please enter a numeric valu
    currency_input: str = input('Currency (USD/EUR/GBP):')
    calculate_finances(monthly_input, tax_rate_input, other_e

main()
```

```
Monthly salary:40000
Tax rate (%):12
Other expenses:7000
Currency (USD/EUR/GBP):INR
-------------------------------------
---------------------
Monthly Income: 40,000.00 INR
Tax Rate: 12.0%
Yearly Salary: 480,000.00 INR
Monthly Tax: 4,800.00 INR
Yearly Tax: 57,600.00 INR
Monthly Net Income: 28,200.00 INR
Yearly Net Income: 338,400.00 INR
-------------------------------------
---------------------
```

---

39. Expense Splitter

Aim: To split expense between a number of users

```
def calc_split(total_amount: float, number_of_people: int, cu
    if number_of_people <= 0:
        raise ValueError("Number of people must be positive")
    share: float = total_amount / number_of_people
    print(f"Each person should pay {currency}{share:.2f}")

def main() -> None:
    while True:
        try:
```

```
            total_amount: float = float(input("Enter the tota
            number_of_people: int = int(input("Enter the numb
            currency: str = input("Enter the currency (e.g., 
            calc_split(total_amount, number_of_people, curren
            break
        except ValueError as e:
            print(f"Error: {e}")
main()
```

```
Enter the total amount: 6000
Enter the number of people: 4
Enter the currency (e.g., USD, EUR, GBP):
NR
Each person should pay INR1500.00
```

---

40. Notepad

Aim: Making a notepad using tkinter with save and load file functionality

```
import tkinter as tk
from tkinter import filedialog
from tkinter import Tk, Text, Frame, Button

class Notepad:
    def __init__(self, root: Tk ) -> None:
        self.root = root
        self.root.title("Notepad")
        self.root.geometry("600x500")

        self.text_area = Text(self.root, wrap='word')
        self.text_area.pack(fill='both', expand=True)

        #frame
        self.button_frame  = Frame(self.root)
        self.button_frame.pack()
        #save button
        self.save_button = Button(self.button_frame, text='sa
        self.save_button.pack(side='left')
```

```
        self.load_button = Button(self.button_frame, text='lo
        self.load_button.pack(side='left')

    def save_file(self):
        filePath = filedialog.asksaveasfilename(defaultextens
        with open(filePath, 'w') as file:
            file.write(self.text_area.get(1.0, tk.END))
        print(f'File saved to: {filePath}')

    def load_file(self):
        filePath = filedialog.askopenfilename(defaultextensio
        with open(filePath, 'r') as file:
            content = file.read()
            self.text_area.delete(1.0, tk.END)
            self.text_area.insert(tk.INSERT, content)

        print(f'File loaded')

    def run(self):
        self.root.mainloop()



def main() -> None:
    root= tk.Tk()
    app = Notepad(root=root)
    app.run()
main()
```