Python Programs

1. WAP to swap two numbers

num1 **=** int(input("Enter first number: "))

num2 **=** int(input("Enter second number: "))

print()

print("Before swap")

print("Number1 = " **,** num1**,** " & Number2 = " **,** num2)

temp **=** num1

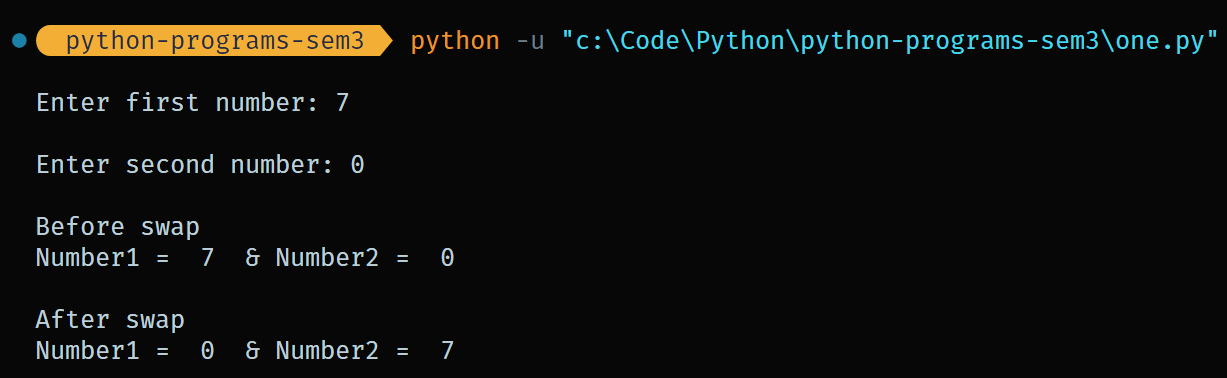
num1 **=** num2

num2 **=** temp

print()

print("After swap")

print("Number1 = " **,** num1**,** " & Number2 = " **,** num2)



1. WAP to check whether the given number is even or odd using conditional statement.

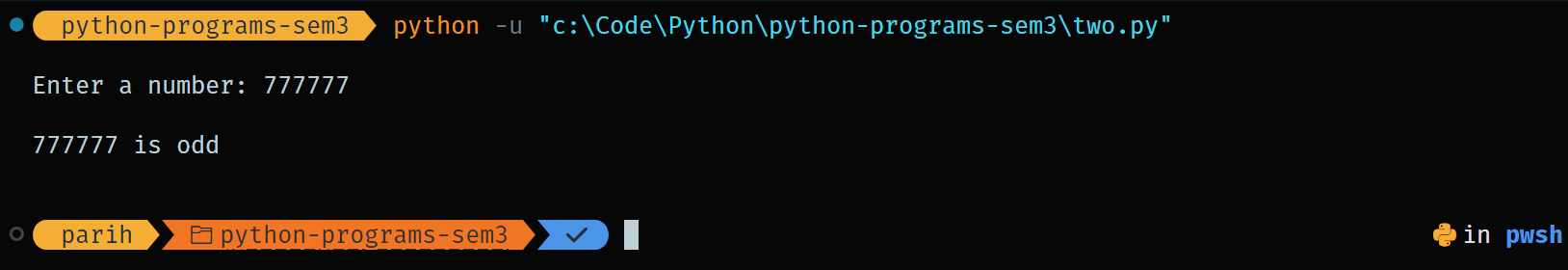
num **=** int(input("Enter a number: "))

**if** num **%** 2 **==** 0 **:**

    print(num**,** "is even")

**else:**

    print(num**,** "is odd")



1. WAP to check whether the given year is leap year or not using conditional statement.

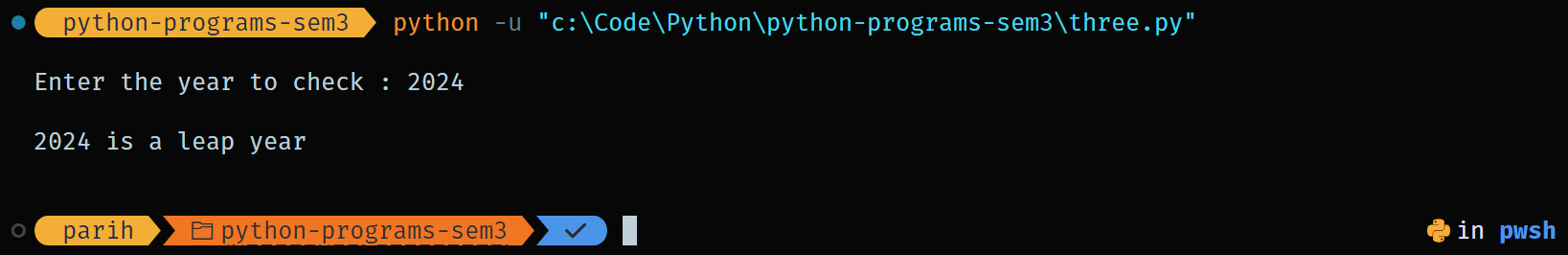
year **=** int(input("Enter the year to check : "))

**if** (year **%** 400 **==** 0) **and** (year **%** 4 **==** 0) **or** (year **%** 100 **!=** 0)**:**

    print(year**,** "is a leap year")

**else:**

    print(year**,** "is not a leap year")



1. WAP to find max. number from list of no's using loops

numbers\_list **=** [33**,**22**,**45**,**67**,**89**,**9**,**1**,**23**,**100]

maxNum **=** numbers\_list[0]

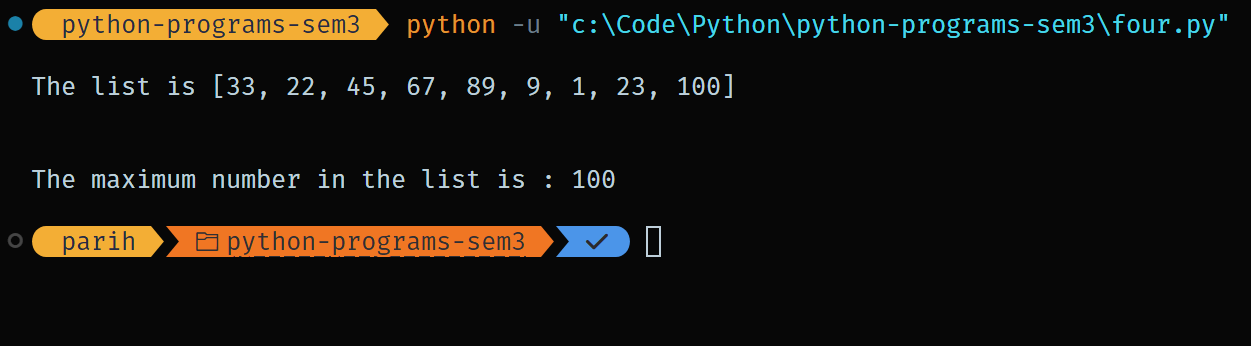
print("The list is"**,** numbers\_list**,**)

**for** num **in** numbers\_list**:**

**if** num **>** maxNum**:**

        maxNum **=** num

print("The maximum number in the list is :"**,** maxNum)



1. WAP to calculate sum of even no's from 1 to 100 including 1 & 100 using range function.

start **=** 1

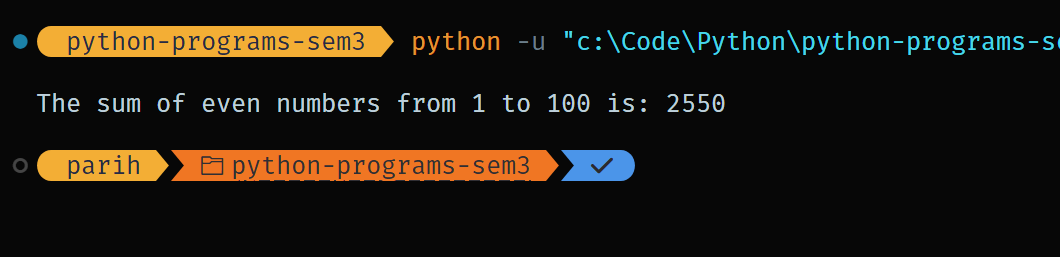
end **=** 100

sum **=** 0

**for** num **in** range(start**,** end **+** 1**,** 2)**:**

    sum **+=** num

print("The sum of even numbers from 1 to 100 is:"**,** sum)



1. WAP to add sum of two-digit numbers.

def checkTwoDigit(num):

if num < 10 or num > 99:

print("Please enter a valid two-digit number.")

return False

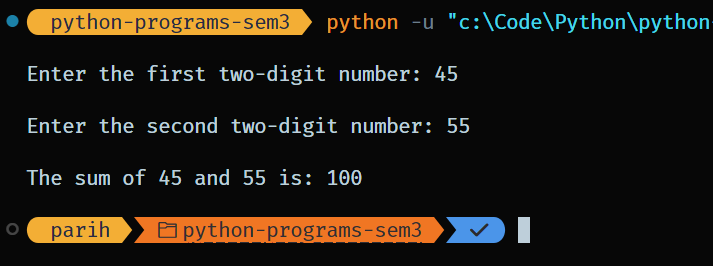
return True

first\_number = int(input("Enter the first two-digit number: "))

second\_number = int(input("Enter the second two-digit number: "))

if (checkTwoDigit(first\_number) and checkTwoDigit(second\_number)):

print(f"The sum of {first\_number} and {second\_number} is: {first\_number + second\_number}")

else: print("Both numbers should be two-digit numbers.") 

1. WAP to convert variables from one datatype to another.

def type\_conversion\_demo():

int\_var = 10

float\_var = 20.5

str\_var = "30"

converted\_float = float(int\_var)

print(f"Converted int to float: {converted\_float} (Type: {type(converted\_float)})")

converted\_int = int(float\_var)

print(f"Converted float to int: {converted\_int} (Type: {type(converted\_int)})")

converted\_str\_to\_int = int(str\_var)

print(f"Converted string to int: {converted\_str\_to\_int} (Type:{type(converted\_str\_to\_int)})")

converted\_int\_to\_str = str(int\_var)

print(f"Converted int to string: '{converted\_int\_to\_str}' (Type: {type(converted\_int\_to\_str)})")

str\_float = "45.67"

converted\_str\_to\_float = float(str\_float)

print(f"Converted string to float: {converted\_str\_to\_float} (Type: {type(converted\_str\_to\_float)})")

list\_var = [1, 2, 3]

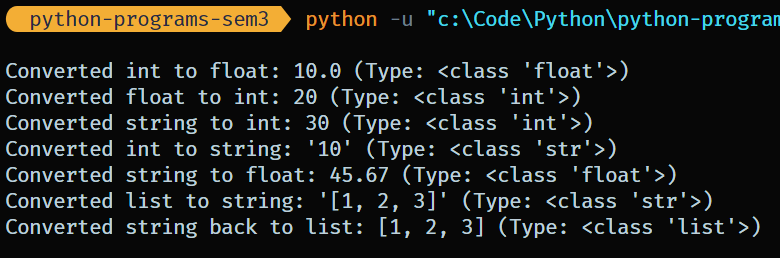
converted\_list\_to\_str = str(list\_var)

print(f"Converted list to string: '{converted\_list\_to\_str}' (Type: {type(converted\_list\_to\_str)})")

converted\_str\_to\_list = eval(converted\_list\_to\_str)

print(f"Converted string back to list: {converted\_str\_to\_list} (Type: {type(converted\_str\_to\_list)})")

type\_conversion\_demo()



1. WAP to generate random numbers and then calculate its sum of digits.

import random

def sum\_of\_digits(number):

total = 0

while number > 0:

digit = number % 10

total += digit

number //= 10

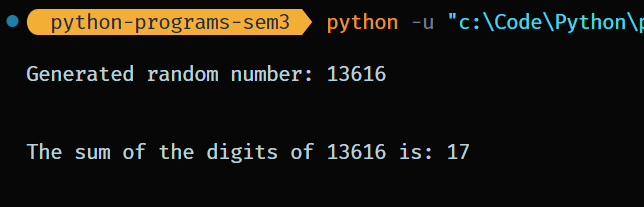
return total

random\_number = random.randint(1, 99999)

print(f"Generated random number: {random\_number}")

digit\_sum = sum\_of\_digits(random\_number)

print(f"The sum of the digits of {random\_number} is: {digit\_sum}")



1. WAP to generate first 50 prime number series. Also check whether given any two no's are coprime or not.

import math

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(math.sqrt(n)) + 1):

if n % i == 0:

return False

return True

def generate\_primes(count):

primes = []

num = 2

while len(primes) < count:

if is\_prime(num):

primes.append(num)

num += 1

return primes

def gcd(a, b):

while b:

a, b = b, a % b

return a

def are\_coprime(a, b):

return gcd(a, b) == 1

first\_50\_primes = generate\_primes(50)

print("First 50 prime numbers:")

print(first\_50\_primes)

num1 = int(input("Enter the first number: "))

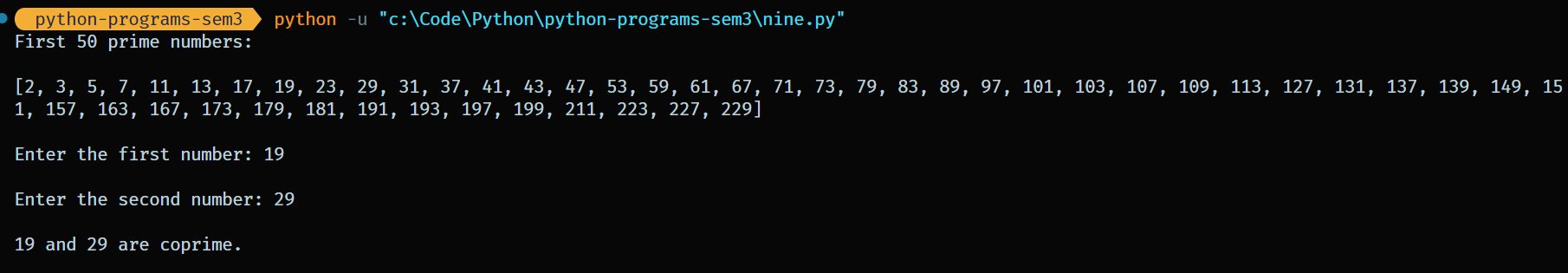
num2 = int(input("Enter the second number: "))

if are\_coprime(num1, num2):

print(f"{num1} and {num2} are coprime.")

else:

print(f"{num1} and {num2} are not coprime.")



1. WAP to display Fibonacci series.

num = int(input("Enter a number upto which you want to display the fibonacci series: "))

n1, n2 = 0, 1

print("Fibonacci Series:", n1, n2, end=" ")

for i in range(2, num):

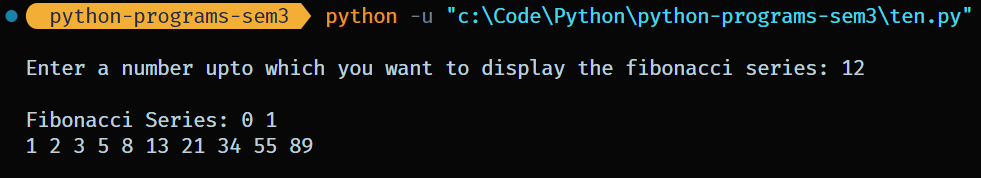
n3 = n1 + n2

n1 = n2

n2 = n3

print(n3, end=" ")

print()



1. WAP to check whether a number is palindrome or not.

n=int(input("Enter number:"))

temp=n

rev=0

while(n>0):

dig=n%10

rev=rev\*10+dig

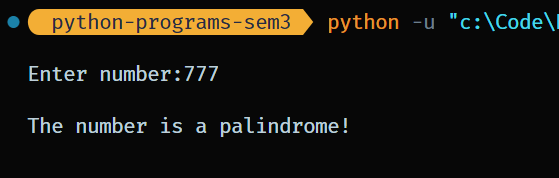
n=n//10

if(temp==rev):

print("The number is a palindrome!")

else:

print("The number isn't a palindrome!")



1. WAP to perform the following operations on string:
2. find index of particular character.

text = input("Enter the string: ")

char\_to\_find = input("Enter the character to find: ")

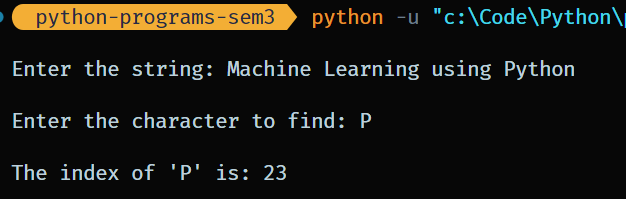
if char\_to\_find in text:

index = text.index(char\_to\_find)

print(f"The index of '{char\_to\_find}' is: {index}")

else:

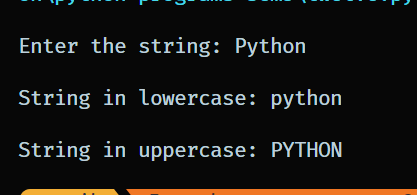
print(f"'{char\_to\_find}' not found in the string.")



1. perform string uppercase and lowercase.

print("String in lowercase:", text.lower())

print("String in uppercase:", text.upper())



1. count the occurrence of particular character in string.

text = input("Enter the string: ")

char = input("Enter the character to find and count: ")

if char in text:

index = text.index(char)

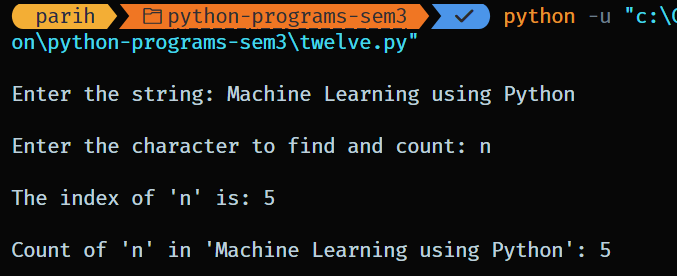
print(f"The index of '{char}' is: {index}")

else:

print(f"'{char}' not found in the string.")

count = text.count(char)

print(f"Count of '{char}' in '{text}': {count}")



1. WAP to perform operations on sets.

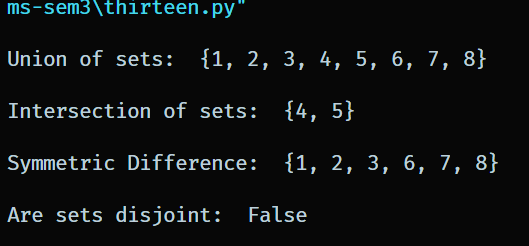
s1 = {1, 2, 3, 4, 5}

s2 = {4, 5, 6, 7, 8}

print("Union of sets: ", s1.union(s2))

print("Intersection of sets: ", s1.intersection(s2))

print("Symmetric Difference: ", s1.symmetric\_difference(s2))

print("Are sets disjoint: ", s1.isdisjoint(s2)) 

1. WAP to perform Operations on Tuples.

x = ('a', 'b', 'c', 'b')

y = ('d', 'e', 'f')

print("Count of b: ", x.count('b'))

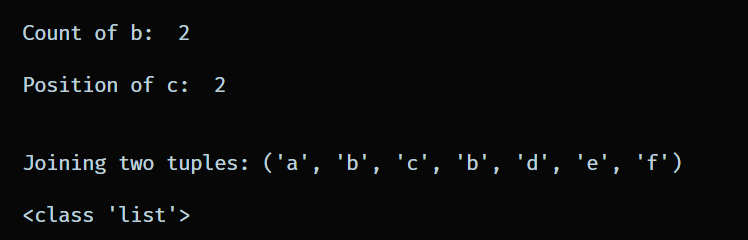
print("Position of c: ", x.index('c'))

z = x+y

print("Joining two tuples:", z)

l = list(x)

print(type(l))



1. WAP to perform operations on List.

list1 = [1, 2, 3, 4, 5, 6]

list2 = [7, 8, 9, 10, 11]

print("Joining two lists: ", list1+list2)

list2.append(12)

print("After appending: ", list2)

list1.reverse()

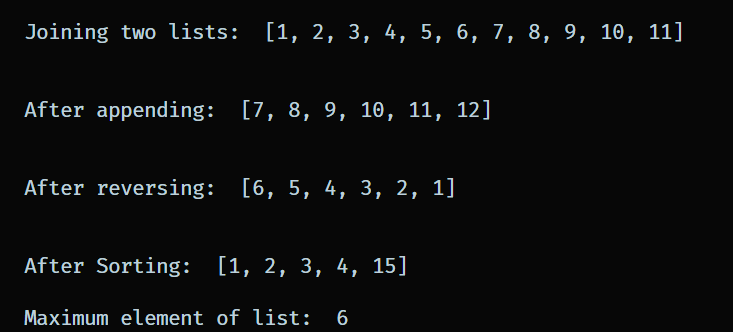
print("After reversing: ", list1)

x = [3, 15, 4, 1, 2]

x.sort()

print("After Sorting: ", x)

print("Maximum element of list: ", max(list1))



1. WAP for Exception Handling.

try:

num1 = int(input("Enter the numerator: "))

num2 = int(input("Enter the denominator: "))

result = num1 / num2

print(f"The result is: {result}")

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

except ValueError:

print("Error: Invalid input. Please enter numeric values.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

finally:

print("Program execution completed.")

