**Python program day-2** Date:22/02/2023

*1.Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid using Stack.*

**Program:**

def isValid(s):

stack = []

mapping = {')': '(', '}': '{', ']': '['} Python program day-2

for char in s:

if char in mapping.values():

stack.append(char)

elif char in mapping.keys():

if stack == [] or mapping[char] != stack.pop():

return False

else:

return False

return stack == []

# Get the input string from the user

input\_string = input("Enter a string containing only the characters '(', ')', '{', '}', '[' and ']': ")

# Check if the input string is valid

result = isValid(input\_string)

# Print the result

if result:

print("true.")

else:

print("false.")

**Output:**

Input: s = "()"

Output: true

Input: s = "(]"

Output: false

*2.Please fill Blank1 and Blank2 in the following program to get the output shown at the end:*class Account:

def \_init\_(self, balance):

self.balance = balance

self.min\_balance = 0

def withdraw(self, amount):

if ((self.balance - amount) >= self.min\_balance): # Blank1

return amount

else:

return 0

class SavingsAccount(Account):

def \_init\_(self, balance):

Account.\_init\_(self, balance)

self.min\_balance = 500

class CurrentAccount(Account):

def \_init\_(self, balance):

Account.\_init\_(self, balance)

self.min\_balance = 0

a1 = SavingsAccount(1000)

a2 = CurrentAccount(1000)

accounts = [a1, a2]

for account in accounts:

print(account.withdraw(1000))

**Output:**

0

1000

*3.Write a program to print numbers from P to Q but except the digit R?*

**Program:**

def contains\_digit(number, digit\_to\_exclude):

return str(digit\_to\_exclude) not in str(number)

def print\_numbers\_except\_digit(P, Q, R):

for number in range(P, Q + 1):

if contains\_digit(number, R):

print(number)

P = int(input("Enter the starting number (P): "))

Q = int(input("Enter the ending number (Q): "))

R = int(input("Enter the digit to exclude (R): "))

print(f"Numbers from {P} to {Q} except those containing the digit {R}:")

print\_numbers\_except\_digit(P, Q, R)

**Sample Input**:

P = 60

Q = 70

R = 3

**Sample Output:**

Numbers are = 60, 61, 62, 64, 65, 66, 67, 68, 69, 70.

*4. Write a program to read a character until a \* is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.*

**Program:**

def count\_characters():

uppercase\_count = 0

lowercase\_count = 0

number\_count = 0

while True:

char = input("Enter a character (\* to stop): ")

if char == '\*':

break

if char.isupper():

uppercase\_count += 1

elifchar.islower():

lowercase\_count += 1

elifchar.isdigit():

number\_count += 1

print("Uppercase count:", uppercase\_count)

print("Lowercase count:", lowercase\_count)

print("Number count:", number\_count)

count\_characters()

**Sample Input:**

Enter \* to exit…

Enter any character: W Enter any character: d

Enter any character: A

Enter any character: G

Enter any character: g

Enter any character: H

Enter any character: \*

**Sample Output:**

Total count of lower

case:2 Total count of upper

case:4 Total count of numbers =0

*5. Python Program to Create a List of Tuples with the First Element as the Number and Second Element as the Square of the Number*.

**Program:**

def create\_list\_of\_tuples(n):

# Using list comprehension to create the list of tuples

return [(i, i\*i) for i in range(1, n+1)]

# Taking input from the user for the range

n = int(input("Enter the range for the list of tuples: "))

# Creating the list of tuples

result\_list = create\_list\_of\_tuples(n)

# Printing the result

print("List of tuples with number and its square:")

print(result\_list)

**Sample Input:**

Enter the lower range:45 Enter the upper range:49

**Sample Output:**

[(45, 2025), (46, 2116), (47, 2209), (48, 2304), (49, 2401)]

*6.Python Program to create a list of all numbers in a range which are perfect squares and the sum of the digits of the number is less than 10*

**Program:**

def is\_perfect\_square(n):

return n == int(n\*\*0.5) \*\*2

def sum\_of\_digits(n):

return sum(int(digit) for digit in str(n))

def perfect\_squares\_with\_sum\_less\_than\_10(lower, upper):

result = []

for number in range(lower, upper+1):

if is\_perfect\_square(number) and sum\_of\_digits(number) < 10:

result.append(number)

return result

# Example usage:

lower = int(input("Enter lower range: "))

upper = int(input("Enter upper range: "))

result\_list = perfect\_squares\_with\_sum\_less\_than\_10(lower, upper)

print("List of perfect squares with sum of digits less than 10:")

print(result\_list)

**Sample Input & Output:**

Enter lower range: 1 Enter upper range: 40 [1, 4, 9, 16, 25, 36]

*7.Write a program to calculate tax given the following conditions:*

*a. If income is less than or equal to 1,50,000 then no tax*

*b. If taxable income is 1,50,001 – 3,00,000 the charge 10% tax*

*c. If taxable income is 3,00,001 – 5,00,000 the charge 20% tax*

*d. If taxable income is above 5,00,001 then charge 30% tax*

**Program:**

def calculate\_tax(income):

if income <= 150000:

tax = 0

elif income <= 300000:

tax = (income - 150000) \* 0.1

elif income <= 500000:

tax = 15000 + (income - 300000) \* 0.2

else:

tax = 45000 + (income - 500000) \* 0.3

return tax

# Taking input for income

income = float(input("Enter the income: "))

# Calculating tax

tax = calculate\_tax(income)

# Printing the result

print("Tax payable:", tax)

**Sample Input:**

Enter the income:200000

**Sample Output:**

Tax= 20000

*8.Write a program to print the total amount available in the ATM machine with the conditions applied.*

*Total denominations are 2000, 500, 200, 100, get the denomination priority from the user and the total number of notes from the user to display the total available balance to the user*

**Program:**

def main():

denominations = [2000, 500, 200, 100]

num\_notes = []

# Get the number of notes for each denomination from the user

print("Enter the number of notes for each denomination:")

for denomination in denominations:

num = int(input(f"Number of {denomination} notes: "))

num\_notes.append(num)

# Calculate the total amount available

total = sum(denomination \* num for denomination, num in zip(denominations, num\_notes))

# Display the total amount available

print(f"Total amount available in the ATM machine: {total}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Sample Input:**

Enter the 1st Denomination: 500

Enter the 1st Denomination number of notes: 4 Enter the 2nd Denomination: 100

Enter the 2nd Denomination number of notes: 20 Enter the 3rd Denomination: 200

Enter the 3rd Denomination number of notes: 32 Enter the 4th Denomination: 2000

Enter the 4th Denomination number of notes: 1

**Sample Output:**

Total Available Balance in ATM: 12400

*9. Find the Mth maximum number and Nth minimum number in an array and then find the sum of it and difference of it.*

**Program:**

def find\_mth\_maximum\_and\_nth\_minimum(arr, m, n):

# Sort the array in ascending order

sorted\_arr = sorted(arr)

# Find the Mth maximum number

mth\_max = sorted\_arr[-m]

# Find the Nth minimum number

nth\_min = sorted\_arr[n - 1]

return mth\_max, nth\_min

def main():

# Example array

arr = [10, 5, 20, 15, 30]

# Example values of M and N

m = 2

n = 3

# Find the Mth maximum number and Nth minimum number

mth\_max, nth\_min = find\_mth\_maximum\_and\_nth\_minimum(arr, m, n)

# Calculate the sum and difference

total = mth\_max + nth\_min

difference = mth\_max - nth\_min

**Sample Input**:

Array of elements = {14, 16, 87, 36, 25, 89, 34}

M = 1

N = 3

**Sample Output:**

1st Maximum Number = 89 3rd Minimum Number = 25 Sum = 114

Difference = 64

*10. Write a program to count all the prime and composite numbers entered by the user.*

**Program:**

def is\_prime(n):

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i<= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def count\_prime\_and\_composite():

prime\_count = 0

composite\_count = 0

while True:

try:

num = int(input("Enter a number (enter 0 to quit): "))

if num == 0:

break

if is\_prime(num):

prime\_count += 1

else:

composite\_count += 1

except ValueError:

print("Invalid input. Please enter a valid number.")

print("Total prime numbers entered:", prime\_count)

print("Total composite numbers entered:", composite\_count)

# Function call

count\_prime\_and\_composite()

**Sample Input:**

Enter the numbers 4

54

29

71

7

59

98

23

**Sample Output:**

Composite number:3

Prime number:5

1*1.Write a program to calculate Pow(x,n), Add(x,n), Sub(x,n), Mul(x,n), Div(x,n)? Get the input and choice from the user.*

**Program:**

def power(x, n):

return x \*\* n

def add(x, n):

return x + n

def subtract(x, n):

return x - n

def multiply(x, n):

return x \* n

def divide(x, n):

if n == 0:

return "Error: Division by zero!"

return x / n

def main():

x = float(input("Enter the value of x: "))

n = float(input("Enter the value of n: "))

choice = input("Enter the operation (power/add/subtract/multiply/divide): ").lower()

if choice == "power":

result = power(x, n)

elif choice == "add":

result = add(x, n)

elif choice == "subtract":

result = subtract(x, n)

elif choice == "multiply":

result = multiply(x, n)

elif choice == "divide":

result = divide(x, n)

else:

result = "Invalid operation choice!"

print(f"Result of {choice}({x}, {n}): {result}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Sample Input:**

X = 2

N = 4

Choice : 2

**Sample Output**:

Add(X,N) = 6

*12.Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid using Stack.*

**Program:**

def isValid(s):

stack = []

mapping = {")": "(", "}": "{", "]": "["}

for char in s:

if char in mapping:

top\_element = stack.pop() if stack else '#'

if mapping[char] != top\_element:

return False

else:

stack.append(char)

return not stack

# Test the function with examples

if \_\_name\_\_ == "\_\_main\_\_":

test\_cases = ["()", "()[]{}", "(]", "([)]", "{[]}", "((()))", "((())"]

for s in test\_cases:

result = isValid(s)

print(f"Input: {s}, Valid: {result}")

**Input**: s = "()"

**Output:** true

*13. Write a program to print numbers from P to Q but except the digit R?*

**Program:**

def print\_numbers\_without\_digit(P, Q, R):

for num in range(P, Q + 1):

if str(R) not in str(num):

print(num)

# Test the function with examples

if \_\_name\_\_ == "\_\_main\_\_":

P = int(input("Enter the starting number (P): "))

Q = int(input("Enter the ending number (Q): "))

R = int(input("Enter the digit to exclude (R): "))

print(f"Numbers from {P} to {Q} excluding the digit {R}:")

print\_numbers\_without\_digit(P, Q, R)

**Sample Input:**

P = 60

Q = 70

R = 3

**Sample Output:**

Numbers are = 60, 61, 62, 64, 65, 66, 67, 68, 69, 70

*14. Write a program to read a character until a \* is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.*

**Program:**

def count\_characters():

uppercase\_count = 0

lowercase\_count = 0

number\_count = 0

print("Enter characters (type \* to end):")

while True:

char = input("Enter a character: ")

if char == '\*':

break

elifchar.isupper():

uppercase\_count += 1

elifchar.islower():

lowercase\_count += 1

elifchar.isdigit():

number\_count += 1

return uppercase\_count, lowercase\_count, number\_count

if \_\_name\_\_ == "\_\_main\_\_":

uppercase, lowercase, numbers = count\_characters()

print(f"Number of uppercase letters: {uppercase}")

print(f"Number of lowercase letters: {lowercase}")

print(f"Number of numbers: {numbers}")

**Sample Input:**

Enter \* to exit…

Enter any character: W Enter any character: d Enter any character: A Enter any character: G Enter any character: g Enter any character: H Enter any character: \*

**Sample Output:**

Total count of lower case:2 Total count of upper case:4 Total count of numbers =0

15.In an organization they decide to give bonus to all the employees on New Year. A 5%

bonus on salary is given to the grade A workers and 10% bonus on salary to the grade B

workers. Write a python program to enter the salary and grade of the employee. If the salary

of the employee is less than $10,000 then the employee gets an extra 2% bonus on salary

Calculate the bonus that has to be given to the employee and print the salary that the

employee will get.

**Program:**

def calculate\_bonus(salary, grade):

bonus = 0

if grade == 'A':

bonus = 0.05 \* salary

elif grade == 'B':

bonus = 0.10 \* salary

if salary < 10000:

bonus += 0.02 \* salary

return bonus

def calculate\_total\_salary(salary, bonus):

return salary + bonus

def main():

salary = float(input("Enter the salary of the employee: "))

grade = input("Enter the grade of the employee (A or B): ").upper()

bonus = calculate\_bonus(salary, grade)

total\_salary = calculate\_total\_salary(salary, bonus)

print("Bonus:", bonus)

print("Total salary with bonus:", total\_salary)

if \_name\_ == "\_main\_":

main()

**Sample Input &amp; Output**:

Enter the grade of the employee: B

Enter the employee salary: 50000

Salary=50000

Bonus=5000.0

Total to be paid:55000.0

*16. Little Robert likes mathematics. Today his teacher has given him two integers and asked*

*him to find out how many integers can divide both the numbers. Would you like to help him*

*in completing his school assignment?*

**Program:**

def common\_divisors(a, b):

def gcd(x, y):

while y:

x, y = y, x % y

return x

gcd\_value = gcd(a, b)

count = 0

for i in range(1, int(gcd\_value \*\* 0.5) + 1):

if gcd\_value % i == 0:

count += 1

if i != gcd\_value // i:

count += 1

return count

a = int(input("Enter the first integer: "))

b = int(input("Enter the second integer: "))

result = common\_divisors(a, b)

print(f"The number of integers that can divide both {a} and {b} is: {result}")

**Sample input:**

Enter the numbers: 4, 12

**Output:** 4

*17. Given two strings “s” and “t”, determine if they are isomorphic. Two strings “s” and “t” are*

*isomorphic if the characters in “s” can be replaced to get “t”. All occurrences of a character*

*must be replaced with another character while preserving the order of characters. No two*

*characters may map to the same character, but a character may map to itself.*

*Constraints*

*s and t consist of any valid ascii character.*

**Program:**

def are\_isomorphic(s, t):

if len(s) != len(t):

return False

# Mapping of characters from s to t

s\_to\_t = {}

# Mapping of characters from t to s

t\_to\_s = {}

for i in range(len(s)):

char\_s = s[i]

char\_t = t[i]

# If s\_to\_t does not contain char\_s and t\_to\_s does not contain char\_t

if char\_s not in s\_to\_t and char\_t not in t\_to\_s:

s\_to\_t[char\_s] = char\_t

t\_to\_s[char\_t] = char\_s

# If s\_to\_t does not contain char\_s or t\_to\_s does not contain char\_t

elifchar\_s not in s\_to\_t or char\_t not in t\_to\_s:

return False

# If the mappings are not consistent

elifs\_to\_t[char\_s] != char\_t or t\_to\_s[char\_t] != char\_s:

return False

return True

# Input strings

s = input("Enter the first string: ")

t = input("Enter the second string: ")

# Check if the strings are isomorphic

if are\_isomorphic(s, t):

print(f"The strings '{s}' and '{t}' are isomorphic.")

else:

print(f"The strings '{s}' and '{t}' are not isomorphic.")

**Input:** s = “egg”, t = “add”

**Output:** true

**Input:** s = “foo”, t = “bar”

**Output:** false

*18. Write a python program to print the first n perfect numbers and its first m factors*

***Program:***

def is\_perfect\_number(num):

factors\_sum = sum([i for i in range(1, num) if num % i == 0])

return factors\_sum == num

def find\_factors(num, m):

factors = [i for i in range(1, num+1) if num % i == 0]

return factors[:m]

def print\_perfect\_numbers\_with\_factors(n, m):

count = 0

num = 1

while count < n:

if is\_perfect\_number(num):

factors = find\_factors(num, m)

print(f"Perfect number {count+1}: {num}, Factors: {factors}")

count += 1

num += 1

# Example usage:

n = int(input("Enter the value of n (number of perfect numbers): "))

m = int(input("Enter the value of m (number of factors to display): "))

print(f"First {n} perfect numbers and their first {m} factors:")

print\_perfect\_numbers\_with\_factors(n, m)

**Sample Input:**

N=3, M=4

**Sample Output:**

First 4 factors of 6 are: 1,2,3,6

First 4 Factors of 28 are: 1,2,4,7

First 4 Factors of 496 are: 1,2,4,8

19.

**Program:**

def main():

# Read the number of loaves of day old bread from the user

num\_loaves = int(input("Enter the number of loaves of day old bread being purchased: "))

# Regular price for each loaf

regular\_price = 185

# Calculate discount

discount = 0.60 \* regular\_price

# Calculate total price

total\_price = (1 - 0.60) \* regular\_price \* num\_loaves

# Display the results

print(f"Regular price for each loaf: {regular\_price:.2f}")

print(f"Discount because it is day old: {discount:.2f}")

print(f"Total price: {total\_price:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Sample Input:**

Enter the number of fresh loves purchased: 5

Enter the number of day old loaves purchased: 3

**Sample Output:**

Regular price: Rs.185.00

Amount of  new loaves: 925.00

Amount of day old loaves: 333.00

Total amount: Rs. 1258.00

*20.(a). Write a Python program to display the current date and time*

***Program:***

import datetime

current\_datetime = datetime.datetime.now()

print("Current date and time:")

print(current\_datetime)

**Sample Output :**

Current date and time :

2014-07-05 14:34:14

*20(b). Write a Python program that calculates the area of a circle based on the radius entered by the user*

**Program:**

def calculate\_circle\_area(radius):

import math

area = math.pi \* (radius \*\* 2)

return area

radius = float(input("Enter the radius of the circle: "))

area = calculate\_circle\_area(radius)

print(f"Area = {area}")

**Sample Output:**

r = 1.1

Area = 3.8013271108436504