### **Coding Exercise-Python**

**ANSWERS**

QUESTION 1:

row\_num = int(input("Input number of rows: "))

col\_num = int(input("Input number of columns: "))

multi\_list = [[0 for col in range(col\_num)] for row in range(row\_num)]

for row in range(row\_num):

for col in range(col\_num):

multi\_list[row][col] = row \* col

print(multi\_list)

QUESTION 2:

def sort\_comma\_separated\_words(input\_string):

words = input\_string.split(',')

words.sort()

return ','.join(words)

input\_words = input("Enter comma-separated words: ")

sorted\_words = sort\_comma\_separated\_words(input\_words)

print(sorted\_words)

QUESTION 3:

def process\_words():

input\_string = input("Enter words separated by spaces: ")

words\_list = input\_string.split()

unique\_words = sorted(list(set(words\_list)))

print(" ".join(unique\_words))

process\_words()

QUESTION 4:

even\_digit\_numbers = []

for num in range(1000, 3001):

num\_str = str(num)

if all(int(digit) % 2 == 0 for digit in num\_str):

even\_digit\_numbers.append(num\_str)

print(",".join(even\_digit\_numbers))

QUESTION 5:

s = input()

letters = sum(c.isalpha() for c in s)

digits = sum(c.isdigit() for c in s)

print("LETTERS", letters)

print("DIGITS", digits)

QUESTION 6:

sentence = input("Enter a sentence: ")

upper\_case = 0

lower\_case = 0

for char in sentence:

if char.isupper():

upper\_case += 1

elif char.islower():

lower\_case += 1

print("UPPER CASE", upper\_case)

print("LOWER CASE", lower\_case)

QUESTION 7:

input\_data = input("Enter transactions: ")

transactions = input\_data.split(',')

net\_amount = 0

for txn in transactions:

txn = txn.strip()

if not txn:

continue

type\_, amount = txn.split()

amount = int(amount)

if type\_.upper() == 'D':

net\_amount += amount

elif type\_.upper() == 'W':

net\_amount -= amount

print(net\_amount)

QUESTION 8:

import re

input\_data = input("Enter comma-separated passwords to validate: ")

passwords = input\_data.split(',')

valid\_passwords = []

for password in passwords:

password = password.strip()

if 6 <= len(password) <= 12:

if (re.search("[a-z]", password) and

re.search("[A-Z]", password) and

re.search("[0-9]", password) and

re.search("[$#@]", password) and

not re.search("\s", password)):

valid\_passwords.append(password)

# Print the valid passwords as comma-separated string

print(",".join(valid\_passwords))

QUESTION 9:

records = []

print("Enter name, age, height tuples (type 'done' to finish):")

while True:

line = input()

if line.lower() == 'done':

break

parts = tuple(line.split(","))

if len(parts) == 3:

records.append(parts)

records.sort(key=lambda x: (x[0], int(x[1]), int(x[2])))

QUESTION 10:

import math

input\_data = input("Enter movement commands (e.g., UP 5,DOWN 3,LEFT 3,RIGHT 2): ")

commands = input\_data.split(',')

x = 0

y = 0

for command in commands:

command = command.strip()

if not command:

continue

direction, steps = command.split()

steps = int(steps)

if direction.upper() == "UP":

y += steps

elif direction.upper() == "DOWN":

y -= steps

elif direction.upper() == "LEFT":

x -= steps

elif direction.upper() == "RIGHT":

x += steps

distance = math.sqrt(x\*\*2 + y\*\*2)

print(round(distance))

QUESTION 11:

input\_str = input("Enter a string: ")

input\_str = input\_str.lower()

result = ""

i = 0

while i < len(input\_str):

count = 1

while i + 1 < len(input\_str) and input\_str[i] == input\_str[i + 1]:

count += 1

i += 1

result += input\_str[i] + str(count)

i += 1

print(result)

QUESTION 12:

import re

def find\_pairs\_with\_sum\_9(s):

result = []

i = 0

while i < len(s):

if s[i].isalpha():

char1 = s[i]

j = i + 1

num\_str = ""

while j < len(s) and not s[j].isalpha():

num\_str += s[j]

j += 1

if j < len(s) and s[j].isalpha():

char2 = s[j]

if num\_str and sum(int(d) for d in num\_str) == 9:

result.append(f"{char1},{char2}")

i = j

else:

i += 1

return result

input\_str = input("Enter an alphanumeric string: ")

pairs = find\_pairs\_with\_sum\_9(input\_str)

for pair in pairs:

print(pair)

QUESTION 13:

binary\_str = input("Enter a binary number: ")

count\_ones = binary\_str.count('1')

pairs = count\_ones \* (count\_ones - 1) // 2

print(pairs)

QUESTION 14:

def find\_minimum\_denominations(valid\_currency, money):

valid\_currency.sort(reverse=True)

result = {}

for denom in valid\_currency:

if money >= denom:

count = money // denom

result[denom] = count

money -= denom \* count

for denom in result:

print(f"{denom}-{result[denom]}")

valid\_currency = list(map(int, input("Enter valid currency (comma-separated): ").split(',')))

money = int(input("Enter the money amount: "))

find\_minimum\_denominations(valid\_currency, money)

QUESTION 15:

import math

def non\_consecutive\_stop\_ways(n, m):

if m > n:

return 0

return math.comb(n - m + 1, m)

n = int(input("Enter total number of stops (n): "))

m = int(input("Enter number of stops to make (m): "))

print("Output:", non\_consecutive\_stop\_ways(n, m))

QUESTION 16:  
  
def determine\_winner(a, b):

a = a.lower()

b = b.lower()

if a == b:

return "DRAW"

elif (a == "stone" and b == "scissor") or \

(a == "paper" and b == "stone") or \

(a == "scissor" and b == "paper"):

return "Player A wins"

else:

return "Player B wins"

score\_a = 0

score\_b = 0

round\_num = 1

print("Game: Stone Paper Scissor")

print("Instructions: First to reach 5 points wins.\n")

while score\_a < 5 and score\_b < 5:

print(f"Round {round\_num}:")

move\_a = input("Player A: ").strip()

move\_b = input("Player B: ").strip()

result = determine\_winner(move\_a, move\_b)

if result == "Player A wins":

score\_a += 1

elif result == "Player B wins":

score\_b += 1

print(f"Result: {result}")

print(f"Score -> Player A: {score\_a} | Player B: {score\_b}")

print("-" \* 30)

round\_num += 1

if score\_a == 5:

print("Player A is the WINNER!")

else:

print("Player B is the WINNER!")

QUESTION 17:

import re

def validate\_email(email):

if email.count('@') != 1:

return "Invalid: Email must contain exactly one '@' symbol"

pattern = r'^[a-z0-9.\_]+@[a-z0-9.\_]+$'

if not re.match(pattern, email):

return "Invalid: Email contains invalid characters or uppercase letters"

return "Valid Email"

email\_input = input("Enter an email to validate: ").strip()

print(validate\_email(email\_input))

QUESTION 18:

a)

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

num = 1

for i in range(1, n + 1):

row = []

for j in range(i):

row.append(str(num))

num += 1

print(" \* ".join(row))

b)

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

for i in range(1, n + 1):

print(" " \* (n - i) + "\* " \* i)

for i in range(n - 1, 0, -1):

print(" " \* (n - i) + "\* " \* i)

c)

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

arr = []

num = 1

for i in range(1, n + 1):

row = []

for j in range(i):

row.append(str(num))

num += 1

arr.append(" \* ".join(row))

for row in arr:

print(row)

for row in reversed(arr[:-1]):

print(row)

d)

n = int(input("Enter rows (7 recommended): "))

for i in range(n):

if i == 0:

print(" \*\*\* ")

elif i == 3:

print(" \* \*\*\* ")

elif i == n - 1:

print(" \*\*\* ")

elif i > 3:

print(" \* \* ")

else:

print(" \* ")

e)

n = int(input("Enter odd row count: "))

for i in range(n):

row = []

for j in range(n):

if i == 0 or i == n - 1 or j == n // 2:

row.append("1")

else:

row.append("0")

print(" ".join(row))

QUESTION 19:

def cyclic\_rotate(case\_type, s, times):

s = list(s)

for \_ in range(times):

if case\_type == 1:

first = s.pop(0)

s.append(first)

elif case\_type == 2:

last = s.pop()

s.insert(0, last)

print("".join(s)) # Print after each rotation

case = int(input("Enter case (1 for left, 2 for right): "))

string\_input = input("Enter string (e.g., happy): ").strip().lower()

rotations = int(input("Enter number of rotations: "))

cyclic\_rotate(case, string\_input, rotations)

QUESTION 20:

healthy\_data = {

"Sugar level": 15,

"Blood pressure": 32,

"Heartbeat rate": 71,

"weight": 65,

"fat percentage": 10

}

patient\_data = {}

print("Enter your pathology test values:")

for key in healthy\_data:

value = int(input(f"{key}: "))

patient\_data[key] = value

print("\n--- Patient Input ---")

for k, v in patient\_data.items():

print(f"{k}: {v}")

difference\_report = {}

print("\n--- Differences and Warnings ---")

for key in healthy\_data:

diff = patient\_data[key] - healthy\_data[key]

difference\_report[key] = diff

if diff != 0:

print(f" WARNING: {key} differs from ideal value.")

print("\nDifference Report:")

print(difference\_report)

print("\n--- Detailed Explanation ---")

for key, diff in difference\_report.items():

if diff < 0:

print(f"{key} {diff}\nThe {key.lower()} is {-diff} less than the ideal value\n")

elif diff > 0:

print(f"{key} {diff}\nThe {key.lower()} is {diff} more than the ideal value\n")

else:

print(f"{key} is ideal.\n")

QUESTION 21:

def is\_armstrong(number):

num\_str = str(number)

power = len(num\_str)

total = sum(int(digit) \*\* power for digit in num\_str)

return total == number

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

if is\_armstrong(num):

print("Armstrong number")

else:

print("Not an Armstrong number")

QUESTION 22:

def decimal\_to\_binary(n):

if n == 0:

return "0"

binary = ""

while n > 0:

remainder = n % 2

binary = str(remainder) + binary

n = n // 2

return binary

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

binary\_result = decimal\_to\_binary(num)

print(binary\_result)

QUESTION 23:

def is\_perfect\_number(n):

if n <= 1:

return False

sum\_divisors = 0

for i in range(1, n):

if n % i == 0:

sum\_divisors += i

return sum\_divisors == n

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

if is\_perfect\_number(num):

print("Perfect number")

else:

print("Not a perfect number")