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")
```