PART 1 (1–10) Python Coding Questions

□ 1. Reverse a String

```
s = "python"
print(s[::-1])
```

Line-by-line explanation:

- `s = "python"` → Stores string in variable `s`
- `s[::-1]` → Slicing:
  - start to end
  - step = -1 means reverse direction
- `print()` → Displays reversed string

Output:

```
nohtyp
```

□ 2. Even or Odd

```
num = 10

if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Explanation:

- `num = 10` → stores number
- `num % 2` → gives remainder when divided by 2
- `== 0` → checks if divisible
- `if` → condition checking
- `print("Even")` → runs if condition true
- `else` → runs otherwise

□ 3. Factorial

```
n = 5
fact = 1

for i in range(1, n+1):
    fact *= i

print(fact)
```

Explanation:

- `n = 5` → number input
- `fact = 1` → initial multiplication value
- `range(1, n+1)` → loop from 1 to 5
- `fact *= i` → multiply step-by-step
- `print()` → final result

Logic:

1 × 2 × 3 × 4 × 5 = 120

□ 4. Prime Number Check

```
n = 7
flag = True

for i in range(2, n):
    if n % i == 0:
        flag = False
        break

print("Prime" if flag else "Not Prime")
```

Explanation:

- `n = 7` → input number
- `flag = True` → assume number is prime
- `range(2, n)` → check divisibility from 2 to n-1
- `n % i == 0` → checks factor
- `flag = False` → not prime if divisible
- `break` → stop loop early
- final print → uses ternary condition

□ 5. Fibonacci Series

```
n = 10
a, b = 0, 1

for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

Explanation:

- `n = 10` → number of terms
- `a, b = 0, 1` → starting values
- `range(n)` → loop 10 times
- `print(a)` → print current number
- `a, b = b, a + b` → update values:
    - next = sum of previous two

□ 6. Largest in List

```
arr = [10, 25, 5, 99]
print(max(arr))
```

Explanation:

- `arr = [...]` → list of numbers
- `max(arr)` → finds highest value
- `print()` → displays result

□ 7. Remove Duplicates

```
arr = [1, 2, 2, 3, 3]
print(list(set(arr)))
```

Explanation:

- `set(arr)` → removes duplicates automatically
- `list()` → converts set back to list

- `print()` → displays unique elements

□ 8. Count Vowels

```
s = "hello world"
vowels = "aeiou"
count = 0

for ch in s:
    if ch in vowels:
        count += 1

print(count)
```

Explanation:

- `s` → input string
- `vowels` → reference string
- `count = 0` → counter
- `for ch in s` → loop each character
- `if ch in vowels` → checks vowel
- `count += 1` → increases count

□ 9. Palindrome Check

```
s = "madam"

if s == s[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")
```

Explanation:

- `s` → input string
- `s[::-1]` → reversed string
- `==` → compares original and reverse
- if equal → palindrome

□ 10. Swap Numbers

```
a = 10
b = 20

a, b = b, a
print(a, b)
```

Explanation:

- `a = 10, b = 20` → initial values
- `a, b = b, a` → Python tuple swapping
- `print()` → displays swapped values

□ END OF PART 1

□ PART 2 (11–20) Python Coding Questions

□ 11. Sum of Digits of a Number

```
n = 1234
sum_digits = 0

while n > 0:
    sum_digits += n % 10
    n //= 10

print(sum_digits)
```

Line-by-line explanation:

- `n = 1234` → input number

- `sum_digits = 0` → stores final sum
- `while n > 0` → loop until number becomes 0
- `n % 10` → extracts last digit
- `sum_digits +=` → adds digit to total sum
- `n //= 10` → removes last digit
- `print()` → displays final result

Logic:

1234 → 1 + 2 + 3 + 4 = 10

□ 12. Armstrong Number

```
n = 153
temp = n
sum = 0

while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if n == sum:
    print("Armstrong")
else:
    print("Not Armstrong")
```

Line-by-line explanation:

- `n = 153` → input number
- `temp = n` → copy for manipulation
- `sum = 0` → stores sum of cubes
- `while temp > 0` → loop through digits
- `temp % 10` → extract last digit
- `digit ** 3` → cube of digit
- `sum +=` → add cube to sum
- `temp //= 10` → remove last digit
- compare original number with sum

Logic:

153 → $1^3 + 5^3 + 3^3 = 153$

□ 13. Second Largest Element in List

```
arr = [10, 20, 5, 40, 30]

arr = list(set(arr))
arr.sort()

print(arr[-2])
```

Line-by-line explanation:

- `arr = [...]` → input list
- `set(arr)` → removes duplicates
- `list()` → converts back to list
- `sort()` → sorts in ascending order
- `arr[-2]` → second last element (second largest)

□ 14. Frequency Count of Elements

```
arr = [1, 2, 2, 3, 3, 3]
freq = {}

for i in arr:
    freq[i] = freq.get(i, 0) + 1

print(freq)
```

Line-by-line explanation:

- `arr` → input list
- `freq = {}` → empty dictionary
- `for i in arr` → loop each element
- `freq.get(i, 0)` → returns current count or 0
- `+1` → increases count
- store back in dictionary
- print final frequency map

Output:

```
{1:1, 2:2, 3:3}
```

□ 15. Check Palindrome Number (Numeric)

```
n = 121
temp = n
rev = 0

while n > 0:
    digit = n % 10
    rev = rev * 10 + digit
    n //= 10

if temp == rev:
    print("Palindrome")
else:
    print("Not Palindrome")
```

Line-by-line explanation:

- `n = 121` → input number
- `temp = n` → store original number
- `rev = 0` → reversed number
- `n % 10` → extract last digit
- `rev * 10 + digit` → build reversed number
- `n //= 10` → remove last digit
- compare original and reversed

□ 16. Find Maximum in List (Manual)

```
arr = [10, 50, 30, 90, 20]
max_val = arr[0]

for i in arr:
    if i > max_val:
        max_val = i

print(max_val)
```

Line-by-line explanation:

- `arr` → list of numbers
- `max_val = arr[0]` → assume first element is largest
- loop through each element
- compare with current max
- update if larger found
- print final maximum

□ 17. Check If String Contains Only Digits

```
s = "12345"

if s.isdigit():
    print("Only digits")
else:
    print("Not only digits")
```

Line-by-line explanation:

- `s` → input string
- `.isdigit()` → built-in function checks digits only
- if true → only numbers
- else → contains characters

□ 18. Count Words in a String

```
s = "Python is easy to learn"

words = s.split()

print(len(words))
```

Line-by-line explanation:

- `s` → input sentence
- `split()` → splits string into words
- default split is space
- `len()` → counts number of words
- print result

□ 19. Swap Without Third Variable

```
a = 5
b = 10

a = a + b
b = a - b
a = a - b
```

```
print(a, b)
```

Line-by-line explanation:

- `a + b` → stores total in a
- `b = a - b` → gets original value of a
- `a = a - b` → gets original value of b
- no extra variable used

□ 20. Check Positive, Negative or Zero

```
num = -5

if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

Line-by-line explanation:

- `num` → input number
- `if num > 0` → checks positive
- `elif num < 0` → checks negative
- `else` → number is zero

□ END OF PART 2

□ PART 3 (21–30) Python Coding Questions

□ 21. Print Prime Numbers in a Range

```
start = 10
end = 30

for num in range(start, end + 1):
    if num > 1:
        for i in range(2, num):
            if num % i == 0:
                break
        else:
            print(num)
```

Line-by-line explanation:

- `start, end` → define range
- `range(start, end + 1)` → loop through numbers
- `if num > 1` → ignore 0 and 1 (not prime)
- inner loop checks divisibility
- `num % i == 0` → if divisible, not prime
- `break` → stop checking
- `else` (of for-loop) → runs if no break → prime number
- `print(num)` → displays prime numbers

□ 22. Star Pattern (Right Triangle)

```
n = 5

for i in range(1, n + 1):
    print("*" * i)
```

Line-by-line explanation:

- `n = 5` → number of rows
- outer loop controls rows
- `"*"` multiplied by `i` → prints increasing stars
- first row = 1 star, second = 2 stars, etc.

□ 23. Sort List Without Built-in sort()

```
arr = [5, 2, 9, 1, 6]

for i in range(len(arr)):
    for j in range(len(arr)):
        if arr[i] < arr[j]:
            arr[i], arr[j] = arr[j], arr[i]

print(arr)
```

Line-by-line explanation:

- two loops → compare each element with others
- `if arr[i] < arr[j]` → compare values
- swap if condition is true
- repeats until list is sorted

Logic:

This is a simple **bubble-sort-like approach**

□ 24. Check Anagram

```
s1 = "listen"
s2 = "silent"

if sorted(s1) == sorted(s2):
    print("Anagram")
else:
    print("Not Anagram")
```

Line-by-line explanation:

- two strings input
- `sorted()` → sorts characters alphabetically
- if both sorted versions match → anagram
- print result

Logic:

Same letters, different order

□ 25. Reverse Words in Sentence

```
s = "Python is easy"

words = s.split()
reversed_sentence = words[::-1]

print(" ".join(reversed_sentence))
```

Line-by-line explanation:

- `split()` → converts sentence into list
- `[::-1]` → reverses word list
- `" ".join()` → joins words back into sentence
- prints reversed word order

□ 26. Matrix Addition (2x2)

```
a = [[1, 2],
     [3, 4]]

b = [[5, 6],
     [7, 8]]

result = [[0, 0],
          [0, 0]]

for i in range(2):
    for j in range(2):
        result[i][j] = a[i][j] + b[i][j]

print(result)
```

Line-by-line explanation:

- `a`, `b` → 2 matrices
- `result` → empty matrix
- outer loop → rows
- inner loop → columns
- add corresponding elements
- store in result matrix

□ 27. Remove Spaces from String

```
s = "P y t h o n"

result = ""

for ch in s:
    if ch != " ":
        result += ch

print(result)
```

Line-by-line explanation:

- `result = ""` → empty string
- loop through each character
- if not space → add to result
- skips spaces completely

□ 28. Find Missing Number in List

```
arr = [1, 2, 4, 5]
n = 5

total = n * (n + 1) // 2
sum_arr = sum(arr)

print(total - sum_arr)
```

Line-by-line explanation:

- `n = 5` → expected range 1 to 5
- formula `n(n+1)/2` → sum of first n numbers
- `sum(arr)` → actual sum
- difference = missing number

Logic:

1 to 5 → 1+2+3+4+5 = 15
list sum = 12
missing = 3

□ 29. Merge Two Lists

```
a = [1, 2, 3]
b = [4, 5, 6]

result = a + b

print(result)
```

Line-by-line explanation:

- two lists
- `+` operator merges lists
- result contains all elements

□ 30. Common Elements in Two Lists

```
a = [1, 2, 3, 4]
b = [3, 4, 5, 6]

result = []

for i in a:
    if i in b:
        result.append(i)

print(result)
```

Line-by-line explanation:

- `result = []` → empty list
- loop through list `a`
- check if element exists in `b`
- if yes → add to result
- prints common elements

□ END OF PART 3

⭐ PART 4 (31–40) Python Coding Questions

🔥 31. Word Frequency Count (Advanced)

```
s = "python is easy python is powerful"

words = s.split()
freq = {}

for w in words:
    if w in freq:
        freq[w] += 1
    else:
        freq[w] = 1

print(freq)
```

Line-by-line explanation:

- `s` → input sentence
- `split()` → converts sentence into list of words
- `freq = {}` → empty dictionary
- loop through each word
- if word already exists → increase count
- else → set count = 1
- print final frequency map

Logic:

Counts how many times each word appears.

🔥 32. Find Duplicate Elements in List

```
arr = [1, 2, 3, 2, 4, 3]
seen = []
duplicates = []

for i in arr:
    if i in seen and i not in duplicates:
        duplicates.append(i)
```

```
    else:
        seen.append(i)

print(duplicates)
```

Line-by-line explanation:

- `seen` → stores visited elements
- `duplicates` → stores repeated elements
- loop through list
- if element already in seen → duplicate
- avoid adding same duplicate twice
- else → add to seen list

□ 33. Replace Vowels with "*"

```
s = "python programming"
result = ""

for ch in s:
    if ch in "aeiou":
        result += "*"
    else:
        result += ch

print(result)
```

Line-by-line explanation:

- `result = ""` → empty string
- loop through characters
- check if vowel
- replace vowel with `*`
- else keep character unchanged

□ 34. Factorial Using Recursion

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

Line-by-line explanation:

- `def factorial(n)` → function definition
- base condition: if n = 0 → return 1
- recursive call: n * factorial(n-1)
- function calls itself repeatedly

Logic:

5 × 4 × 3 × 2 × 1 = 120

□ 35. Fibonacci Using Recursion

```
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)

for i in range(10):
    print(fib(i), end=" ")
```

Line-by-line explanation:

- function `fib(n)` defined
- base case: 0 and 1 return same value
- recursive formula:
    - fib(n-1) + fib(n-2)
- loop prints first 10 terms

□ 36. Armstrong Numbers in Range

```
for num in range(1, 500):
    temp = num
    sum = 0

    while temp > 0:
        digit = temp % 10
        sum += digit ** 3
        temp //= 10

    if num == sum:
        print(num)
```

Line-by-line explanation:

- loop from 1 to 500
- for each number:
  - copy into temp
  - extract digits using `% 10`
  - cube each digit
  - sum digits
- if original equals sum → Armstrong number

□ 37. Binary Search

```
arr = [1, 3, 5, 7, 9]
target = 5

low = 0
high = len(arr) - 1

while low <= high:
    mid = (low + high) // 2

    if arr[mid] == target:
        print("Found")
        break
    elif arr[mid] < target:
        low = mid + 1
    else:
        high = mid - 1
```

Line-by-line explanation:

- array must be sorted
- `low`, `high` → search range
- `mid` → middle element
- if match → found
- if target is greater → search right half
- else → search left half

□ 38. Linear Search

```
arr = [10, 20, 30, 40]
target = 30

for i in arr:
```

```
    if i == target:
        print("Found")
        break
```

Line-by-line explanation:

- loop through each element
- compare with target
- if match → print found
- stop search using break

□ 39. Remove Duplicates Without Set

```
arr = [1, 2, 2, 3, 3, 4]
result = []

for i in arr:
    if i not in result:
        result.append(i)

print(result)
```

Line-by-line explanation:

- `result` → empty list
- loop through original list
- check if element already in result
- if not → add it
- keeps only unique elements

□ 40. Pyramid Pattern

```
n = 5

for i in range(1, n + 1):
    print(" " * (n - i) + "*" * (2*i - 1))
```

Line-by-line explanation:

- `n = 5` → number of rows
- `" " * (n - i)` → spaces for alignment
- `"*" * (2*i - 1)` → stars increase in odd numbers

- forms pyramid shape

Output:

```
    *
   ***
  *****
 *******
*********
```

□ PART 5 (41–50) Python Coding Questions

□ 41. Reverse a Number

```
n = 1234
rev = 0

while n > 0:
    digit = n % 10
    rev = rev * 10 + digit
    n //= 10

print(rev)
```

Line-by-line explanation:

- `n = 1234` → input number
- `rev = 0` → stores reversed number
- `n % 10` → extracts last digit
- `rev * 10 + digit` → builds reversed number step by step
- `n //= 10` → removes last digit
- loop continues until number becomes 0

Logic:

1234 → 4321

□ 42. Armstrong Number (Function Version)

```
def is_armstrong(n):
    temp = n
    sum = 0

    while temp > 0:
        digit = temp % 10
        sum += digit ** 3
        temp //= 10

    return n == sum

print(is_armstrong(153))
```

Line-by-line explanation:

- `def` → defines function
- `temp = n` → copy of number
- loop extracts digits
- cube each digit
- sum all cubes
- return True if equal

□ 43. Sum of Even Numbers in List

```
arr = [1, 2, 3, 4, 5, 6]
total = 0

for i in arr:
    if i % 2 == 0:
        total += i

print(total)
```

Line-by-line explanation:

- `total = 0` → stores sum
- loop through list
- check even using `% 2 == 0`
- add only even numbers
- print final sum

□ 44. Count Uppercase and Lowercase Letters

```
s = "PyThOn"
upper = 0
lower = 0

for ch in s:
    if ch.isupper():
        upper += 1
    elif ch.islower():
        lower += 1

print(upper, lower)
```

Line-by-line explanation:

- `upper`, `lower` → counters
- loop through each character
- `.isupper()` → checks uppercase
- `.islower()` → checks lowercase
- increase counters accordingly

□ 45. Find GCD (Greatest Common Divisor)

```
a = 12
b = 18

while b != 0:
    a, b = b, a % b

print(a)
```

Line-by-line explanation:

- Euclidean algorithm used
- swap values repeatedly
- `a % b` → remainder
- loop continues until b becomes 0
- final `a` is GCD

□ 46. Find LCM

```
a = 12
b = 18

gcd = a

while b != 0:
    a, b = b, a % b
    gcd = a

lcm = (12 * 18) // gcd

print(lcm)
```

Line-by-line explanation:

- first find GCD
- formula: LCM = (a × b) / GCD
- integer division `//` used
- print result

□ 47. Check Perfect Number

```
n = 6
sum = 0

for i in range(1, n):
    if n % i == 0:
        sum += i

if sum == n:
    print("Perfect Number")
else:
    print("Not Perfect")
```

Line-by-line explanation:

- divisors of number are found
- sum all divisors except number itself
- compare sum with original number
- if equal → perfect number

Example:

6 → 1 + 2 + 3 = 6

□ 48. Merge Two Dictionaries

```
a = {"x": 1, "y": 2}
b = {"z": 3}

a.update(b)

print(a)
```

Line-by-line explanation:

- `a` and `b` are dictionaries
- `.update()` merges b into a
- keys and values combined
- prints merged dictionary

□ 49. Remove Punctuation from String

```
import string

s = "Hello, World!"

result = ""

for ch in s:
    if ch not in string.punctuation:
        result += ch

print(result)
```

Line-by-line explanation:

- `import string` → built-in module
- `string.punctuation` → all symbols (!, , . etc.)
- loop through string
- skip punctuation characters
- build clean string

□ 50. Mini Project: Simple Calculator

```
def add(a, b):
    return a + b

def sub(a, b):
    return a - b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

print(add(10, 5))
print(sub(10, 5))
print(mul(10, 5))
print(div(10, 5))
```

Line-by-line explanation:

- each operation is a function
- `def` defines function
- return gives result back
- functions are reusable
- final print calls each function