IIT Bhilai Data Analytics & Visualization Workshop Practice Question Set

Instructions

- For each question below, create a Python file named <YourName>_<questionID>.py, e.g., Ramesh_q1.py.
- Each file must contain the specified function signature.
- You may write additional helper functions if needed.
- Your code will be tested against hidden test cases, so follow the I/O precisely.
- Save and submit all .py files together in the given Google form link following all submission criteria.

Questions

Solved Example

Q1. Disarium Number (Solved Example)
Filename: <YourName>_q1.py
Function:

```
def is_disarium(n):
    """
    A Disarium number is one where the sum of its digits
    raised to their positions equals the number itself.
    e.g. 135: 1^1 + 3^2 + 5^3 = 135
    """
    s = str(n)
    total = 0
    for i in range(len(s)):
        digit = int(s[i])
        power = i + 1
        total += digit ** power
    return total == n
```

Explanation:

- Split the number into its digits from left to right.
- Raise the first digit to the 1st power, the second to the 2nd, etc.
- Sum those powered values.
- If the sum equals the original number, it is Disarium.

Sample Test Cases:

- is_disarium(75) # False, because $7^1 + 5^2 = 32$
- is_disarium(135) # True, $1^1 + 3^2 + 5^3 = 135$
- is_disarium(88) # False, $8^1 + 8^2 = 72$

Practice Questions

Q2. Armstrong Number
Filename: <YourName>_q2.py
Function:

```
def is_armstrong(n):
    """
    An Armstrong number is an n-digit number that equals
    the sum of each digit raised to the power n.
    e.g. 153 = 1^3 + 5^3 + 3^3
    """
    pass
```

Explanation:

- Count how many digits the number has (call it d).
- Raise each digit to the d-th power and sum them.
- If the sum equals the original number, it is an Armstrong number.

Test Cases:

- is_armstrong(153) # True
- is_armstrong(370) # True
- is_armstrong(123) # False

Q3. Trailing Zeros in Factorial
Filename: <YourName>_q3.py
Function:

```
def trailing_zeros(n):
    """
    Return the number of trailing zeros in n! by counting
    how many times 5 divides factors up to n.
    """
    pass
```

Explanation:

- Each pair of 2×5 in the factorial contributes one zero.
- Count multiples of 5, 25, 125, ... up to n and sum those counts.

Test Cases:

- `trailing_zeros(5)  # 1`
- `trailing_zeros(100)  # 24`
- `trailing_zeros(3)  # 0`

Q4. String Palindrome

Filename: <YourName>_q4.py

Function:

```
def is_palindrome(s):
    """
    Return True if string s reads the same forwards and backwards.
    Ignore case and non-alphanumeric characters.
    """
    pass
```

Explanation:

- Clean the string: remove punctuation, convert to lowercase.
- Compare the cleaned string to its reverse.

Test Cases:

- `is_palindrome("Madam")  # True`
- `is_palindrome("Step on no pets")  # True`
- `is_palindrome("Hello, World!")  # False`

Q5. Count Vowels and Consonants

Filename: <YourName>_q5.py

Function:

```
def count_letters(text):
    """
    Return a tuple (vowels, consonants) in the text.
    Count only English letters, ignore case.
    """
    pass
```

Explanation:

- Iterate over each character.
- If it's a letter, classify as vowel (a,e,i,o,u) or consonant.
- Keep separate counts.

Test Cases:

- `count_letters("Data Science")  # (5, 5)`
- `count_letters("IIT Bhilai")  # (4, 4)`
- `count_letters("1234!")  # (0, 0)`

Q6. Fibonacci Number

Filename: <YourName>_q6.py
Function:

```
def fibonacci(n):
    """
    Return the n-th Fibonacci number (F1=1, F2=1).
    Use iteration, not recursion.
    """
    pass
```

Explanation:

- Start with a=1, b=1.
- Loop from 3 to n, updating a, b = b, a+b.
- Return b for the n-th term.

Test Cases:

- `fibonacci(1)  # 1`
- `fibonacci(7)  # 13`
- `fibonacci(10)  # 55`

Q7. Swap Without Temporary Variable

Filename: <YourName>_q7.py
Function:

```
def swap(a, b):
    """
    Swap two integers without using a third variable.
    Return the new pair (a, b).
    """
    pass
```

Explanation:

- Use arithmetic: `a = a + b; b = a - b; a = a - b.`
- Or use tuple unpacking in Python.

Test Cases:

- `swap(3, 5)   # (5, 3)`
- `swap(-1, 10)   # (10, -1)`
- `swap(0, 0)   # (0, 0)`

Q8. Perfect Number
Filename: <YourName>_q8.py
Function:

```
def is_perfect(n):
    """
    A perfect number equals the sum of its proper divisors
    (excluding itself).
    """
    pass
```

Explanation:

- Find all divisors less than n.
- Sum them and compare to n.

Test Cases:

- `is_perfect(6)   # True (1+2+3=6)`
- `is_perfect(28)   # True (1+2+4+7+14=28)`
- `is_perfect(12)   # False`

Q9. Isogram Check
Filename: <YourName>_q9.py
Function:

```
def is_isogram(s):
    """
    Return True if the string has no repeating letters.
    Ignore case and non-letters.
    """
    pass
```

Explanation:

- Clean string to letters only, lowercase.
- Use a set to detect duplicates.

Test Cases:

- `is_isogram("Machine")  # False (two 'i's)`
- `is_isogram("Algorithm")  # True`
- `is_isogram("Dermatoglyphics")  # True`

Q10. Second Largest in List
Filename: <YourName>_q10.py
Function:

```
def second_largest(nums):
    """
    Return the second largest unique number in the list.
    If it doesn't exist, return None.
    """
    pass
```

Explanation:

- Convert list to a set to remove duplicates.
- Sort or iterate to find the top two values.

Test Cases:

- `second_largest([2,5,1,4,5])  # 4`
- `second_largest([7,7,7])  # None`
- `second_largest([10,9,8])  # 9`

Submission Criteria

- Ensure each function passes defined test cases.
- Include docstrings and comments where necessary.
- Zip all .py files into <YourName>_<College>.zip.

- Submit via this Google form link
- https://forms.gle/K29zj8N5r8JMhCz58.