OPEN SOURCE SOFTWARE LAB (15B17CI575)
Lab Assignment 2 (Practice Lab)
ODD 2025
Topic Coverage: Python

Question 1: Create a Python program that initializes a list of five different fruits. Use indexing to print
the first and last fruit from the list. Then, use slicing to print the middle three fruits.

• Example:
o First fruit: "Apple" (or whichever fruit is first in your list)
o Last fruit: "Mango" (or whichever fruit is last in your list)
o Middle fruits: ["Banana", "Orange", "Grapes"] (or whichever fruits are in the middle of your list)

CODE:

```
fruits = ["Apple", "Banana", "Orange", "Grapes", "Mango"]

print("First fruit:", fruits[0])
print("Last fruit:", fruits[-1])
print("Middle fruits:", fruits[1:4])
```

Question 2: Write a Python program that creates a dictionary with five key-value pairs, where the keys are the names of students and the values are their ages. Access and print the age of a specific student using their name as the key. Also, add a new student to the dictionary and print the updated dictionary.
• Example
o Age of the specific student: 20 (or the age of the student you accessed)
o Updated dictionary: {'Alice': 21, 'Bob': 22, 'Charlie': 20, 'David': 23, 'Eve': 19, 'Frank': 24} (or your updated dictionary with the new student added)

CODE:

```
students = {'Alice': 21, 'Bob': 22, 'Charlie': 20, 'David': 23, 'Eve': 19}

print("Age of the specific student:", students['Charlie'])

students['Frank'] = 24

print("Updated dictionary:", students)
```

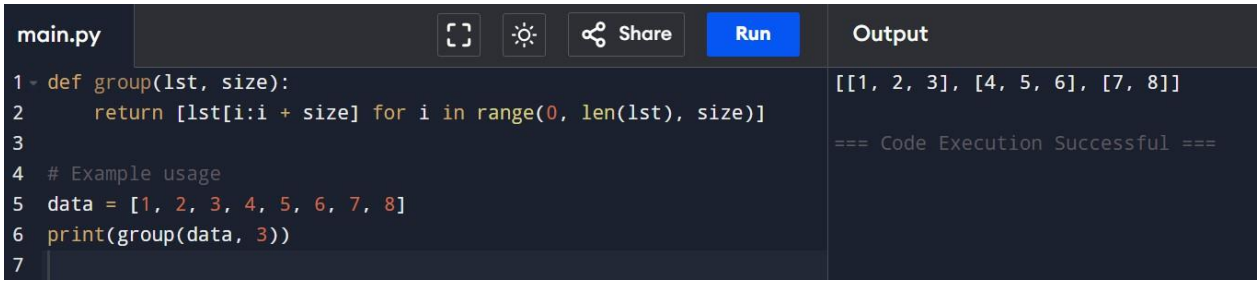
Question 3: Write a function “duplicate” to find all duplicates in a list of 10 integers.
CODE:

```
def duplicate(numbers):
    duplicates = []
    for num in numbers:
        if numbers.count(num) > 1 and num not in duplicates:
            duplicates.append(num)
    return duplicates

# Example usage
nums = [1, 2, 3, 4, 5, 2, 6, 7, 3, 8]
print("Duplicates:", duplicate(nums))
```

Question 4: Write a function group (list, size) that takes a list and splits into smaller lists of given size.

CODE:

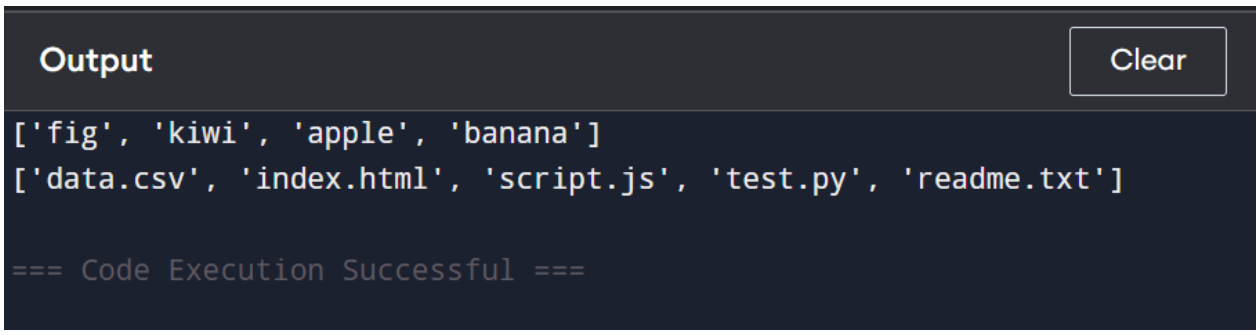
```
def group(lst, size):
    return [lst[i:i + size] for i in range(0, len(lst), size)]

# Example usage
data = [1, 2, 3, 4, 5, 6, 7, 8]
print(group(data, 3))
```



Question 5: Write a function “lensort” to sort a list of strings based on length. 4. Write a function extsort to sort a list of files based on extension.

CODE:

```
def lensort(strings):
    return sorted(strings, key=len)

def extsort(files):
    return sorted(files, key=lambda x: x.split('.')[-1] if '.' in x else '')

# Example usage
words = ["apple", "kiwi", "banana", "fig"]
print(lensort(words))

files = ["test.py", "readme.txt", "script.js", "data.csv", "index.html"]
print(extsort(files))
```



Question 6: Use Python Built-in Functions 'open', 'read', 'readline', 'write', 'writeline' to work with files.

Code

```
import io

file_sim = io.StringIO()

file_sim.write("Hello world\n")

file_sim.writelines(["Line 2\n", "Line 3\n"])

file_sim.seek(0)

print("Read all content:\n" + file_sim.read())

file_sim.seek(0)
print("\nRead one line using readline():")
print(file_sim.readline(), end="")
print(file_sim.readline(), end="")

file_sim.close()
```

Question 7: Write a function to compute the number of characters, words and lines in a file.

```
def file_statistics():
    content = """This is line one.
This is line two.
Third line here."""
    num_chars = len(content)
    num_words = len(content.split())
    num_lines = content.count('\n') + 1
    return num_chars, num_words, num_lines

chars, words, lines = file_statistics()
print("Characters:", chars)
print("Words:", words)
print("Lines:", lines)
```

main.py Share Run Output Clear

```
def file_statistics():
    content = """This is line one.
This is line two.
Third line here."""
    num_chars = len(content)
    num_words = len(content.split())
    num_lines = content.count('\n') + 1
    return num_chars, num_words, num_lines

chars, words, lines = file_statistics()
```

```
Characters: 52
Words: 11
Lines: 3

=== Code Execution Successful ===
```

Question 8: Write a program named ‘reverse.py’ to print lines of a file in reverse order.

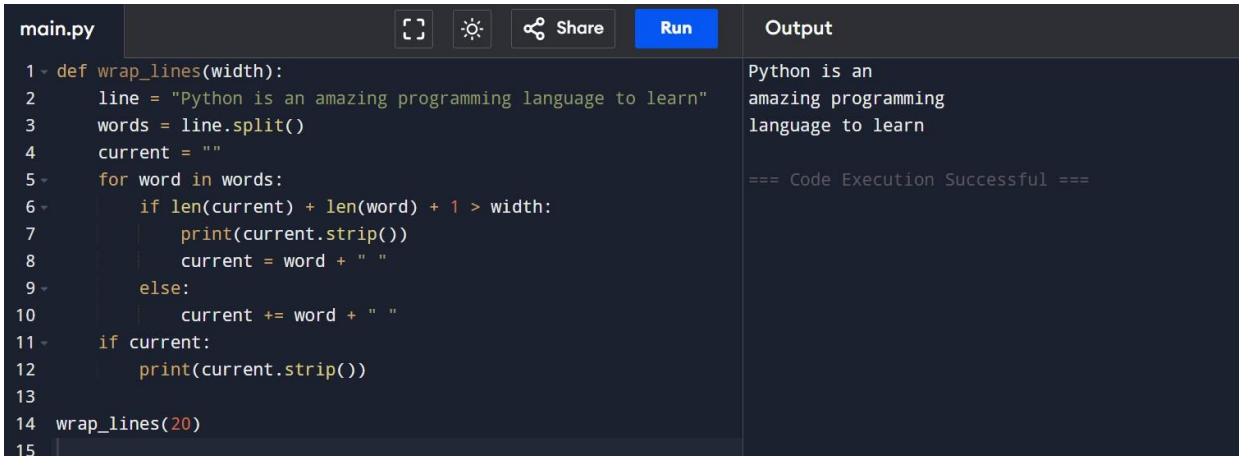
```
def reverse_lines():
    lines = [
        "First line\n",
        "Second line\n",
        "Third line\n"
    ]
    for line in reversed(lines):
        print(line.strip())

reverse_lines()
```

Question 9: Write a program to print each line of a file in reverse order.

```
def reverse_each_line():
    lines = [
        "Hello World",
        "Python is fun",
        "ChatGPT rocks"
    ]
    for line in lines:
        print(line[::-1])

reverse_each_line()
```

Question 10: Write a program 'wrap.py' that takes filename and width as arguments and wraps the lines longer than width.

```
def wrap_lines(width):
    line = "Python is an amazing programming language to learn"
    words = line.split()
    current = ""
    for word in words:
        if len(current) + len(word) + 1 > width:
            print(current.strip())
            current = word + " "
        else:
            current += word + " "
    if current:
        print(current.strip())

wrap_lines(20)
```



Question 11: Python provides a built-in function map that applies a function to each element of a list. Provide an implementation for map using list comprehensions.

Code:

```
def my_map(func, lst):
    return [func(x) for x in lst]
```

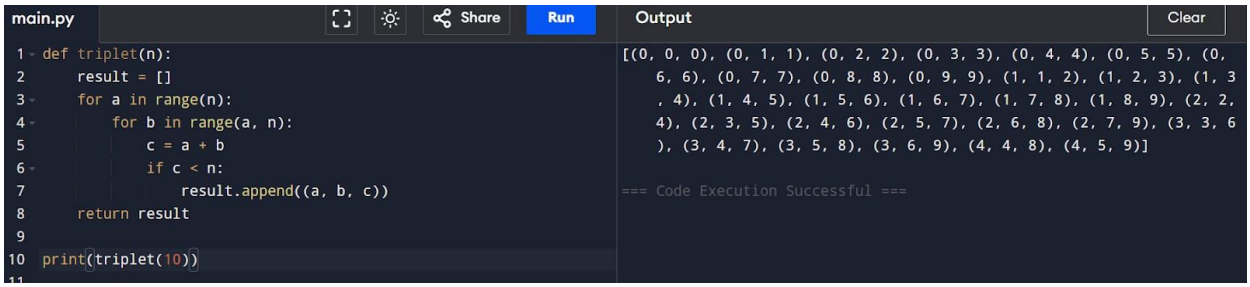
```
# Example
result = my_map(lambda x: x.upper(), ["hello", "world"])
print(result)
```

Question 12: Python provides a built-in function filter (f, a) that returns items of the list a for which

```
def my_filter(func, lst):
    return [x for x in lst if func(x)]

# Example
result = my_filter(lambda x: x > 10, [4, 11, 8, 20, 3])
print(result)
```

Question 13: Write a function triplet that takes a number n as an argument and returns a list of triplets
such that the sum of first two elements of the triplet equals the third element using numbers below n.
Please note that (a, b, c) and (b, a, c) represent the same triplet.
Code:

```
def triplet(n):
    result = []
    for a in range(n):
        for b in range(a, n):
            c = a + b
            if c < n:
                result.append((a, b, c))
    return result

print(triplet(10))
```



Question 14: Write a python function parse_csv to parse csv (comma separated values) files.
14. Write a function mutate to compute all words generated by a single mutation on a given word. A mutation is defined as inserting a character, deleting a character, replacing a character, or swapping 2 consecutive characters in a string. For simplicity consider only letters from a to z.
Code:

```
from io import StringIO

def parse_csv_from_string(data):
    result = []
    file = StringIO(data)
    for line in file:
        line = line.strip()
        values = line.split(',')
        result.append(values)
    return result

csv_data = """name,age,city
Alice,21,Delhi
Bob,22,Mumbai"""

print(parse_csv_from_string(csv_data))
```

Question 15: Write a function nearly_equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on b.

```
def nearly_equal(a, b):
    if a == b:
        return False
```

```
    len_a = len(a)
    len_b = len(b)
    if len_a == len_b + 1:
        for i in range(len_a):
            if a[:i] + a[i+1:] == b:
                return True
    if len_a + 1 == len_b:
        for i in range(len_b):
            if b[:i] + b[i+1:] == a:
                return True

    if len_a == len_b:
        mismatch = 0
        for ch1, ch2 in zip(a, b):
            if ch1 != ch2:
                mismatch += 1
            if mismatch > 1:
                break
        if mismatch == 1:
            return True

    if len_a == len_b:
        for i in range(len_a - 1):
            if (a[i] != b[i] or a[i+1] != b[i+1]) and \
               a[i] == b[i+1] and a[i+1] == b[i]:
                if a[:i] == b[:i] and a[i+2:] == b[i+2:]:
                    return True

    return False
print(nearly_equal("cat", "bat"))
print(nearly_equal("cat", "cta"))
print(nearly_equal("cat", "cat"))
```

Question 16: Write a program to count the frequency of characters in a given file. Can you use character frequency to tell whether the given file is a Python program file, C program file or a text file?

Code:

```
def count_char_frequency_from_string(content):
    freq = {}
    for ch in content:
        freq[ch] = freq.get(ch, 0) + 1
    return freq

def detect_file_type(freq_dict):
    if not freq_dict:
        return "Unknown"

    combined = ''.join(freq_dict.keys())

    if 'def' in combined or ':' in freq_dict and '#' in freq_dict:
        return "Python Program File"
    elif '#include' in combined or ';' in freq_dict and '{' in freq_dict:
        return "C Program File"
    else:
        return "Text File"

content = """#include <stdio.h>
int main() {
    printf("Hello, World!");
    return 0;
}
"""

freq = count_char_frequency_from_string(content)
print("Character Frequencies:\n", freq)
print("\nFile Type Detected:", detect_file_type(freq))
```

Question 17: Write a program to find anagrams in a given list of words. Two words are called anagrams if one word can be formed by rearranging letters of another. For example, ‘eat’, ‘ate’ and ‘tea’ are anagrams.

Code:

```
from collections import defaultdict

def find_anagrams(word_list):
    anagram_groups = defaultdict(list)

    for word in word_list:
        key = ''.join(sorted(word))
        anagram_groups[key].append(word)

    return [group for group in anagram_groups.values() if len(group) > 1]

words = ['eat', 'tea', 'tan', 'ate', 'nat', 'bat', 'tab', 'hello', 'silent', 'listen']
anagram_sets = find_anagrams(words)

for group in anagram_sets:
    print(group)
```