### *Structural Questions*

1. What is the output of the following code?

fruits = ['banana', 'mango', 'apple', 'pineapple', 'watermelon', 'avocado']

for idx, fruit in enumerate(fruits, -2):

print(idx, fruit)

Output:

-2 banana

-1 mango

0 apple

1 pineapple

2 watermelon

3 avocado

Explaination:

- enumerate(fruits, -2) generates pairs of index and value from the fruits list.

- The second argument of enumerate(), which is -2, sets the **starting index**.

- **Loop Iteration:**

- enumerate() assigns an index to each element, starting from -2 and increasing by 1 for each item.

**- Printed Output:**

The loop iterates over fruits, printing each index and fruit.

1. What is the output of the following code snippet?

a = ['Geeks', 'for', 'Geeks']

b = set(a)

a = list(b)

print(a)

Output:

['for', 'Geeks']

Expalination:

- A list a is created with three string elements: ['Geeks', 'for', 'Geeks']

- Default indexing starts at 0.

1. Evaluate the code
2. In one line, explain what line 2 of the code snippet does.

a = ['Geeks', 'for', 'Geeks']

b = enumerate(a)

next\_val = next(b); next\_val = next(b)

print(next\_val)

Explaination:

- enumerate(a) returns an iterator that produces (index, value) pairs.

- next(b) retrieves the next item from the iterator.

**- Each call to** next() **moves forward in the sequence**.

**- The second** next(b) **call overwrites** next\_val, so the final printed value is (1, 'for')

- next(b) retrieves the first item from b, which is (0, 'Geeks').

- next\_val now holds (0, 'Geeks'), but it is immediately reassigned in the next line.

- The second call to next(b) retrieves the next item from b, which is (1, 'for').

- next\_val now holds (1, 'for').

1. What is the output?

(1, 'for')

10, How are range() and enumerate() different?

Both range() and enumerate() deal with numbers and iteration, but they serve different purposes.

1. range() **– Generates a Sequence of Numbers**

- range(start, stop, step) creates a sequence of numbers.

- It does **not** iterate over an existing iterable; it **generates numbers**.

### enumerate() ****– Adds an Index to an Iterable****

### - enumerate(iterable, start=0) assigns an ****index**** to each element in an iterable.

- It does **not generate numbers** but **returns tuples (index, value).**

1. Draw the flowcharts of the for-loop and while-loop
2. What is the purpose of the pass keyword in Python?

The pass keyword in Python is a **placeholder** that allows the interpreter to **skip execution** of a block of code without causing an error. It is often used in places where syntax requires a statement but no action is needed.

1. Give a list trainees = ['Arantis', 'Lois', 'Hans'] , what would be the output of the following:

A)

for trainee in trainees:

if trainee == 'Hans':

pass

print('Trainee:', trainee)

**The Program will not do anything to the code because of the pass keyword, so it will present every name Arantis, Lois, Hans.**

B)

for trainee in trainees:

if trainee == 'Lois':

continue

print('Trainee:', trainee)

- The continue statement **skips** the remaining code in the current loop iteration.

- The loop **moves to the next item** in trainees.

Hans and Arantis will be presented at the output but Lois wouldn’t.

C)

for trainee in trainees:

if trainee == 'Lois':

print('We have a girl!')

break

else:

print("Just guys!")

Once the name Lois is recognized the compiler stops and prints we have a girl