Week 4: Data Structures and Conditional Statements in Python In-Class Activity:

LIST: Lists are ordered collections of items, which can be of any data type. They are mutable, meaning you can add, remove, or modify elements after the list is created. Lists are created using square brackets [] and can contain elements separated by commas.

LIST:

```
# Creating a list with some values assign to it
my_list = [1, 2, 3, 'hello', True]
```

```
# print the value of list
print(my_list)
```

```
[1, 2, 3, 'hello', True]
```

TUPLES: Tuples are similar to lists but are immutable, meaning their elements cannot be changed after creation. They are created using parentheses () and can contain elements separated by commas. Tuples are often used for storing fixed collections of related data.

TUPLES:

```
# tuple
my_tuple = (1, 2, 'world', 'jigawa', 2.5, 5.7)
```

```
new_my_tuple = (1,2, 'world', 'jigawa')
```

```
new_my_tuple
```

```
(1, 2, 'world', 'jigawa')
```

```
b_my_tuple = my_tuple
```

```
b_my_tuple
```

```
(1, 2, 'world', 'jigawa', 2.5, 5.7)
```

```
my_tuple
```

```
(1, 2, 'world', 'jigawa', 2.5, 5.7)
```

```
# printing the value of tuples
print(my_tuple)
```

```
(1, 2, 'world', 'jigawa', 2.5, 5.7)
```

SETS: Sets are unordered collections of unique elements. They do not allow duplicate elements, and the order of elements is not guaranteed. Sets are created using curly braces {} or the set() function.

SETS:

```
# write a script that store some values and assign it to a vriable my_set
my_set = {1, 2, 3, 4}
```

Displaying the vslues of the sets

```
# Displaying the vslues of the sets
print(my_set)
```

```
{1, 2, 3, 4}
```

```
new_sets = {1, 2, 3, 4, 1, 2, 6, 8, 1, 2, 3, 4, 1, 2, 6, 8, 1, 2, 3, 4, 1, 2, 6, 8}
```

```
new_sets
```

```
{1, 2, 3, 4, 6, 8}
```

```
an_set = {'name', 'number', 'address'}
```

```
an_set
```

```
{'address', 'name', 'number'}
```

DICTIONARIES: Dictionaries are unordered collections of key-value pairs. Each element in a dictionary consists of a key and its corresponding value, separated by a colon :. Dictionaries are created using curly braces {} and key-value pairs separated by commas.

EXAMPLE:

```
# write a Dictionaries script that store some values and assign it to a vriable my_dict
my_dict = {'name': 'Salisu', 'age': 30, 'city': 'New York'}
```

Displaying the Dict

```
# print the value of the dictionaries
print(my_dict)
```

```
{'name': 'Salisu', 'age': 30, 'city': 'New York'}
```

```
my_dict
```

```
{'name': 'Alice', 'age': 30, 'city': 'New York'}
```

Introduce conditional statements and functions through practical examples and coding exercises.

Conditional Statements: Conditional statements in Python allow you to execute different blocks of code based on whether a certain condition is true or false. The most commonly used conditional statements in Python are if, elif (short for "else if"), and else. Here's a simple example:

Conditional Statements:In this example, If the value of x is greater than 0, the message "x is positive" will be printed. If the value of x is equal to 0, the message "x is zero" will be printed. Otherwise, if none of the above conditions are met, the message "x is negative" will be printed.

```
# conditional statement
x = 4

if x < 0:
    print("x is positive")
elif x == 0:
    print("x is zero")
else:
    print("x is lower")
```

```
x is lower
```

```
age = 10
if age > 18:
  print("eligible to vote")
else:
  print("not eligble to vote")
```

```
not eligble to vote
```

Functions: Functions in Python allow you to encapsulate reusable blocks of code into named blocks, making your code modular and easier to understand. Here's an example of defining and using a function:

```
Function: In this example, greet is a function that takes one paramete
and prints a greeting message using that name. When you call the funct
("Maryam"), it will print "Hello, Maryam!".
```

Function: In this example, greet is a function that takes one parameter name and prints a greeting message using that name. When you call the function greet("Maryam"), it will print "Hello, Maryam!".

```
# Define a Function called greet
def greet(name):
    print("Hello, " + name)

# Call the function
greet("Maryam")
```

```
Hello, Maryam
```

Coding Exercise: Now, let's combine conditional statements and functions in a coding exercise. Write a Python function called check_even_odd that takes an integer as input and prints whether the number is even or odd. Here's a template to get you started:

```
# Coding example with function and conditonal statement
def check_even_odd(num):
    # Write your code here
    if num % 2 == 0:
        print(num, "is even")
    else:
        print(num, "is odd")

# Test the function
check_even_odd(10)
check_even_odd(7)
```

```
10 is even
7 is odd
```