Assignment

1. What are the two values of the Boolean data type? How do you write them?

The two values of the Boolean data type are:

- **True**: Represents a true value.
- **False**: Represents a false value.

In Python, you write them as True and False (case-sensitive).

2. What are the three different types of Boolean operators?

The three different types of Boolean operators are:

1. **AND**: Returns True if both operands are true.
2. **OR**: Returns True if at least one of the operands is true.
3. **NOT**: Returns True if the operand is false (negates the value).

3. Make a list of each Boolean operator's truth tables.

AND Operator:

A	**B**	**A AND B**
True	True	True
True	False	False
False	True	False
False	False	False

OR Operator:

A	**B**	**A OR B**
True	True	True
rue	False	True

A	B	A OR B
False	True	True
False	False	False

NOT Operator:

A	NOT A
True	False
False	True

4. What are the values of the following expressions?

1. **(5 > 4) and (3 == 5)** → False (True and False)
2. **not (5 > 4)** → False (not True)
3. **(5 > 4) or (3 == 5)** → True (True or False)
4. **not ((5 > 4) or (3 == 5))** → False (not True)
5. **(True and True) and (True == False)** → False (True and False)
6. **(not False) or (not True)** → True (True or False)

5. What are the six comparison operators?

The six comparison operators are:

1. **Equal to** (==): Checks if two values are equal.
2. **Not equal to** (!=): Checks if two values are not equal.
3. **Greater than** (>): Checks if the left value is greater than the right value.
4. **Less than** (<): Checks if the left value is less than the right value.
5. **Greater than or equal to** (>=): Checks if the left value is greater than or equal to the right value.
6. **Less than or equal to** (<=): Checks if the left value is less than or equal to the right value.

6. How do you tell the difference between the equal to and assignment operators? Describe a condition and when you would use one.

- The **equal to operator** (==) is used to compare two values for equality. It returns True if they are equal and False otherwise.
- The **assignment operator** (=) is used to assign a value to a variable.

Example Condition:

- Use = when you want to assign a value to a variable, e.g., x = 5.
- Use == when you want to check if two values are equal, e.g., if x == 5:

7. Identify the three blocks in this code:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

7.Identify the three blocks in this code:

```
spam = 0
if spam == 10:              # Block 1
   print('eggs')
if spam > 5:                # Block 2
   print('bacon')
else:                       # Block 3
   print('ham')
print('spam')
print('spam')
```

Block 1: The if spam == 10: statement and its associated action (print('eggs')).

Block 2: The if spam > 5: statement and its associated action (print('bacon')).

Block 3: The else: statement and its associated action (print('ham')).

8. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! if anything else is stored in spam.

8.spam = 0 # Change this value to test different outputs

```
if spam == 1:
    print('Hello')
elif spam == 2:
    print('Howdy')
else:
    print('Greetings!')
```

9. If your program is stuck in an endless loop, what keys will you press?

To stop a program that is stuck in an endless loop, you can usually press **Ctrl + C** in the terminal or command prompt. This sends a keyboard interrupt signal to the program, causing it to terminate.

10. How can you tell the difference between break and continue?

- **break**: This statement is used to exit a loop entirely. When break is encountered, the loop stops executing, and control is transferred to the statement immediately following the loop.
- **continue**: This statement skips the current iteration of the loop and moves on to the next iteration. When continue is encountered, the rest of the loop body for that iteration is skipped, and the loop proceeds to the next iteration.

11. In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?

- **range(10)**: Generates a sequence of numbers from 0 to 9 (10 numbers total). It starts from 0 by default and goes up to, but does not include, 10.
- **range(0, 10)**: Explicitly generates the same sequence of numbers as range(10), starting from 0 and ending at 9.
- **range(0, 10, 1)**: Generates the same sequence as the previous two, but includes a step parameter of 1, indicating that it increments by 1 for each step. The sequence is still from 0 to 9.

12. Write a short program that prints the numbers 1 to 10 using a for loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.

12.for i in range(1, 11):

```
    print(i)

i = 1

while i <= 10:

    print(i)

    i += 1  # Increment i by 1
```

13. If you had a function named bacon() inside a module named spam, how would you call it after importing spam?

13. import spam

```
spam.bacon()  # Calls the bacon() function from the spam module
```