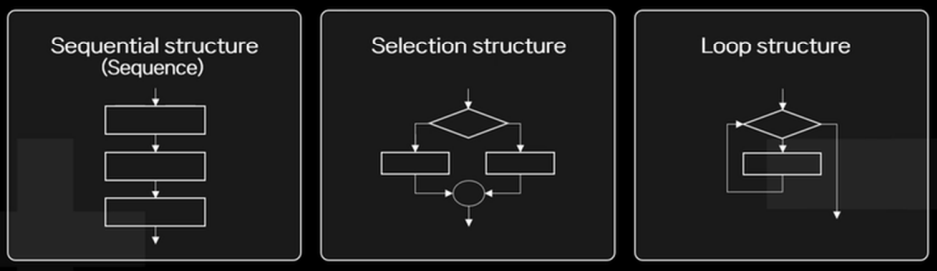
Conditional Statement-1: Condition and Decision Making

Topic 03 Key concept

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

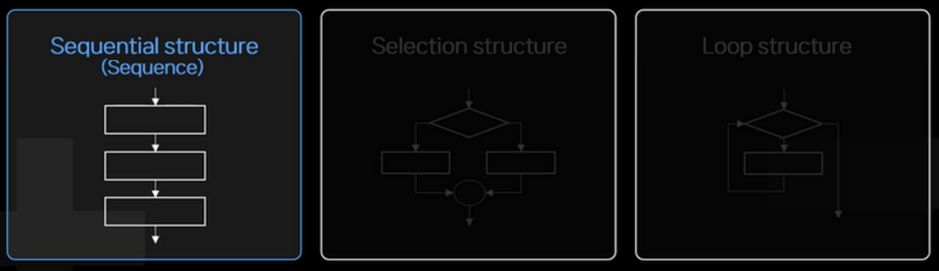
- We have learned that there are three main structures in which a program operates.



1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- We have learned that there are three main structures in which a program operates.

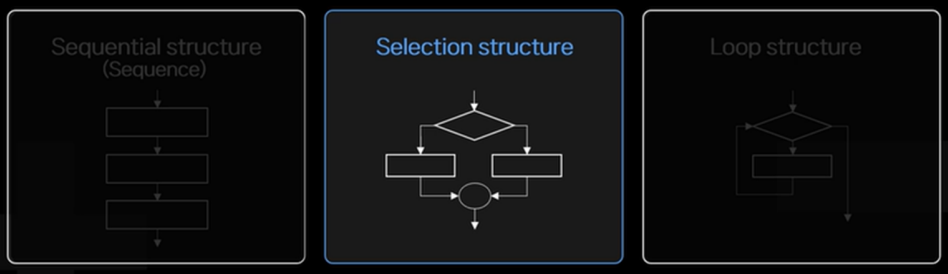


➡ A structure where commands are executed in a sequential order.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- We have learned that there are three main structures in which a program operates.

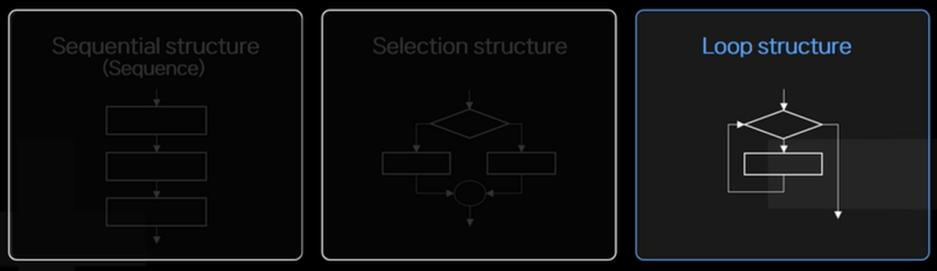


➡ A structure where one of several commands is selected and executed.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- We have learned that there are three main structures in which a program operates.



➡ A structure where the same command is repeatedly executed.

1. Programming with different flows depending on conditions

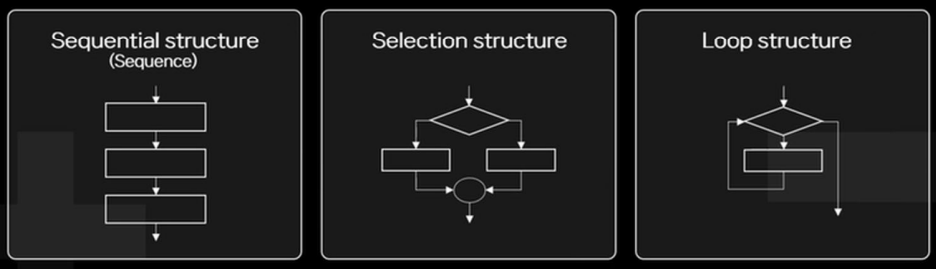
1.1 Three structures of a program represented by a flowchart

- The basic control structures of a program are similar to the basic blocks of Lego. Almost all Lego creations are made using just a few basic blocks.
- The same goes for programs. Even complex programs can be created with just three basic blocks.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

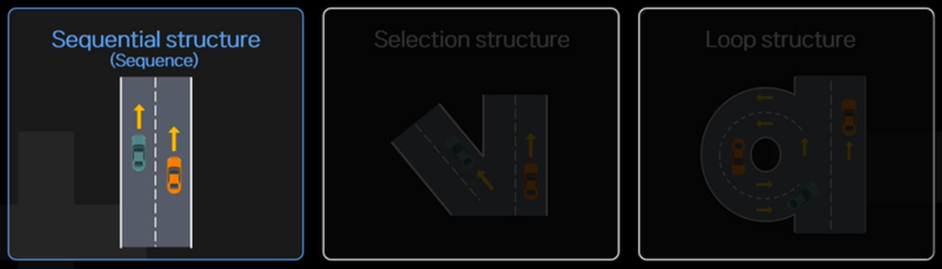
- To easily understand the basic blocks of a program, you can think of them as roads on which cars drive.



1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- To easily understand the basic blocks of a program, you can think of them as roads on which cars drive.

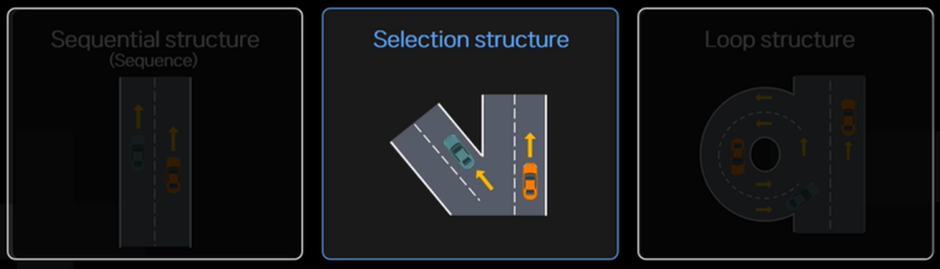


➡ The sequential structure can be compared to a straight road where the car drives forward.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- To easily understand the basic blocks of a program, you can think of them as roads on which cars drive.



➡ The selection structure can be compared to an intersection where the car chooses one of two roads to drive on.

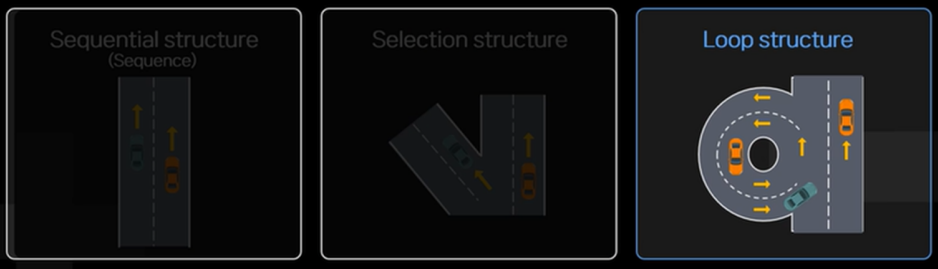
AnB_Prime

——>4<——

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- To easily understand the basic blocks of a program, you can think of them as roads on which cars drive.



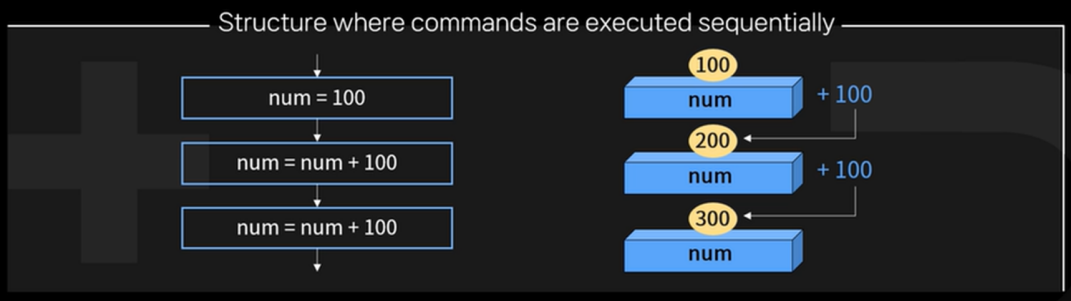
➡ The loop structure can be compared to a roundabout where the car drives in circles. This structure allows for repeating a set of commands multiple times.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- Sequential structure, or sequential statement, is a structure where the code appearing first is executed first.

e.g. whenever “num = num + 100” appears, “num” increases by 100.



1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- Sequential structure, or sequential statement, is a structure where the code appearing first is executed first.

e.g. whenever “num = num + 100” appears, “num” increases by 100.

```
1 num = 100
2 print('num = ', num)
3 num = num + 100
4 print('num = ', num)
5 num = num + 100
6 print('num = ', num)
```

```
num =  100
num =  200
num =  300
```

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

- The Python programs we have written so far follow a sequential structure where the code appearing first is executed first.
- The sequential structure, where commands are executed in order, is a common structure in most programming languages.
- However, programming languages are not limited to having only a sequential flow.
- There are also structures that control the sequential flow. These structures are the selection structure and the loop structure.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart

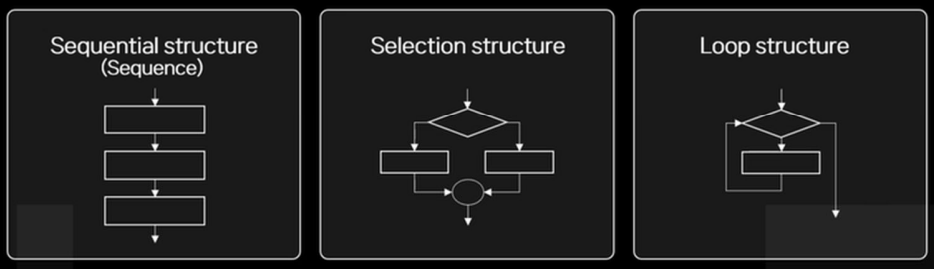
- In addition to sequential statements, statements that control the flow of a program are called control statements. There are three types of control statement:

> - Conditional statement: if statement, if-else statement, if-elif-else statement
> - Loop statement: for loop, while loop
> - Flow-altering command: break, continue

- The selection structure includes conditional statements. The loop structure corresponds to loop statements, and flow-altering commands are used in flows with the conditional statement inside the loop structure.

1. Programming with different flows depending on conditions

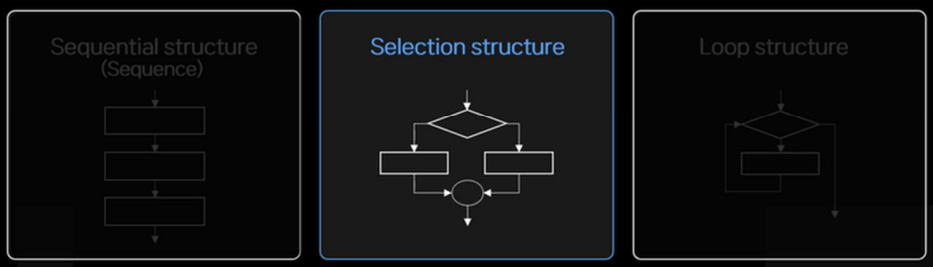
1.1 Three structures of a program represented by a flowchart



- In programming languages, it is possible to execute specific commands only when certain conditions are met or to perform different tasks depending on conditions.

1. Programming with different flows depending on conditions

1.1 Three structures of a program represented by a flowchart



- Therefore, let's explore the conditional statements among the selection structures, which are necessary for solving slightly more complex problems.

1. Programming with different flows depending on conditions

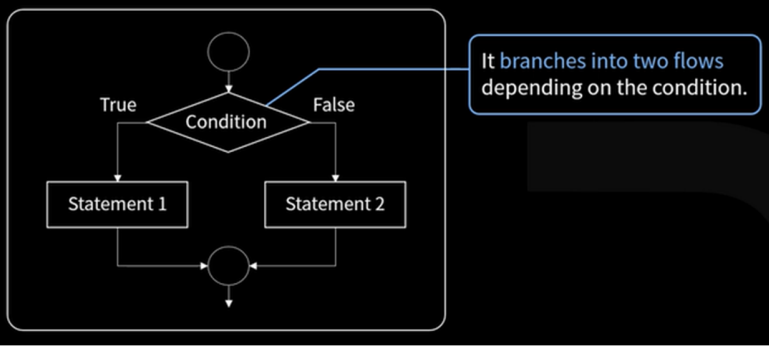
1.2 The need for conditional selective flow

- Why do we need conditional statements?
  - At certain stages of a program, if there is more than one possible path to proceed, we need to decide which path to choose.
  - Without selection structures, we cannot determine which path to choose, resulting in the program always following a single path.
    In that case, the program will perform the same actions and always reach a predetermined conclusion.

1. Programming with different flows depending on conditions

1.3 Flow of conditional statement

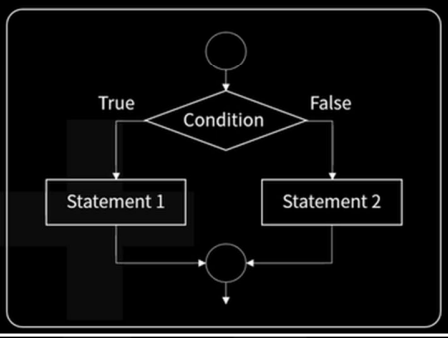
- If there are multiple executable statements and you want to execute only one of them, how can you do that? Let's represent this in a flowchart.



1. Programming with different flows depending on conditions

1.3 Flow of conditional statement

- If there are multiple executable statements and you want to execute only one of them, how can you do that? Let's represent this in a flowchart.



- Depending on a specific condition, either 'Statement 1' or 'Statement 2' is executed.
- The condition is determined by the 'condition expression.'
- Therefore, the condition expression should return a True or False value. In this case, we use the previously learned 'boolean expression' as the condition expression.

2. Conditional Statements

2.1 What is conditional statement?

- We want to create code that behaves differently depending on the situation, as follows:

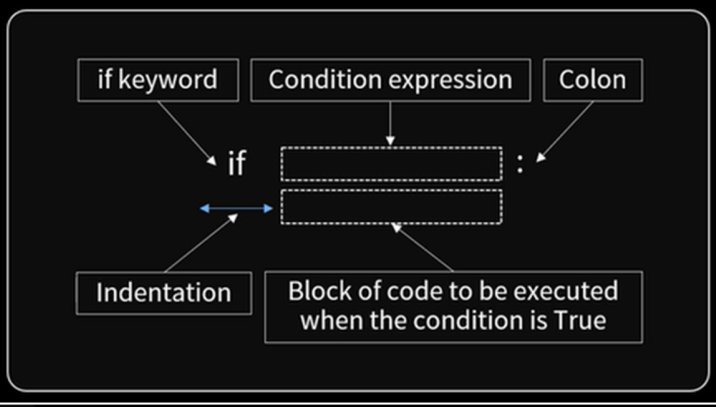
> - Situation 1: If the age is under 20, output 'Youth discount.'
> - Situation 2: If the step count is over 1000, output 'Goal achieved.'

- To implement this code, we need an expression that determines whether a condition is satisfied or not.
- This is called a conditional expression, and it evaluates to a bool type with a value of either True or False.
- The result of the previously learned 'boolean expression' is represented as either True or False. Therefore, it can be used in a condition expression that takes values of True or False.

2. Conditional Statements

2.2 If statement syntax

- Let's learn the syntax of the if statement.



2. Conditional Statements

2.2 If statement syntax

- Let's learn the syntax of the if statement.

Situation 1

If the age is under 20, output 'Youth discount.'

```
if age < 20 :
   print('Youth discount')
```

Situation 2

If the step count is over 1000, output 'Goal achieved.'

```
if walk_count >= 1000 :
   print('Achieved the goal')
```

(Situation 1)

- In the condition clause before the colon (:), using the < operator, the code print('Youth discount') is executed only when the age is less than 20.

2. Conditional Statements

2.2 If statement syntax

- Let's learn the syntax of the if statement.

Situation 1

If the age is under 20, output 'Youth discount.'

```
if age < 20 :
   print('Youth discount')
```

Situation 2

If the step count is over 1000, output 'Goal achieved.'

```
if walk_count >= 1000 :
   print('Achieved the goal')
```

(Situation 2)

- In the condition clause, using the >= operator, the code print('Achieved the goal') is executed only when the step count walk_count is equal to or greater than 1000.

2. Conditional Statements

2.2 If statement syntax

- Let's code using the if statement.

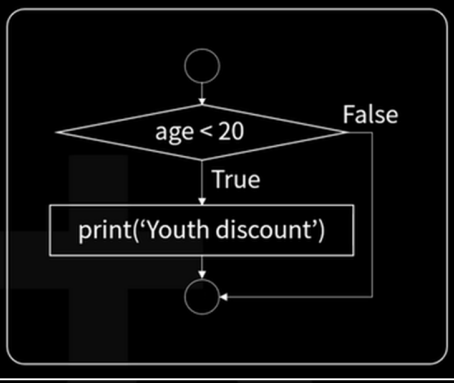
```
age = 18

if age < 20 :
    print('Youth discount')
```

```
Youth discount
```

2. Conditional Statements

2.2 If statement syntax

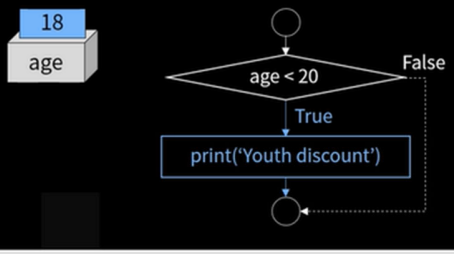
- Let's code using the if statement.



- When the age value is 18, the condition age < 20 evaluates to True, so 'Youth discount' is displayed on the screen.
- If the age value is 20 or above, nothing is printed.

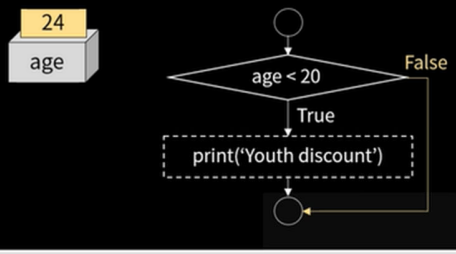
2. Conditional Statements

2.2 If statement syntax




```
age = 18

if age < 20 :
    print('Youth discount')
```

```
Youth discount
```




```
age = 24

if age < 20 :
    print('Youth discount')
```

In this way, the flow moves in different directions depending on the value of the 'age' variable.

2. Conditional Statements

2.3 Blocks and Python's indentation

- In programming, a block refers to a section where a series of code statements are grouped together to define a unit of execution. It is used to restrict the scope of the code.
- The block of code in a conditional statement is a chunk of code that can be executed when a certain condition is true.
- Unlike other programming languages, Python uses indentation to indicate blocks instead of using curly braces({ }). Therefore, in Python, it is necessary to indent these blocks.
- Most Python Integrated Development Environments(IDEs) provide automatic indentation features.

2. Conditional Statements

2.3 Blocks and Python's indentation

- The following code will produce an error.

IndentationError

```
1 age = 18
2
3 if age < 20 :
4 print('Youth discount')
```

```
Cell In[7], line 4
  print('Youth discount')
  ^
IndentationError: expected an indented block after 'if' statement on line 3
```

2. Conditional Statements

2.3 Blocks and Python's indentation

Focus

- Python is a programming language where indentation holds significant importance. Different results are produced based on the indentation.

```
1 age = 24
2 print('Age', age)
3 if age < 20 :
4     print('Youth Age')
5     print('Youth discount')
6 print('Welcome')
```

```
Age 24
Welcome
```

2. Conditional Statements

2.3 Blocks and Python's indentation

- If the fifth line, print('Youth discount'), is not indented in the code, it will result in an incorrect output.

Good coding

```
1 age = 24
2 print('Age', age)
3 if age < 20 :
4     print('Youth Age')
5     print('Youth discount')
6 print('Welcome')
```

```
Age 24
Welcome
```

Bad coding

```
1 age = 24
2 print('Age', age)
3 if age < 20 :
4     print('Youth Age')
5 print('Youth discount')
6 print('Welcome')
```

```
Age 24
Youth discount
Welcome
```

One Step Further

Rules for indented blocks

- The PEP-8 guidelines recommend writing the block code on the line following the colon.

Bad coding

```
1 age = 18
2 if age < 20 : print('Youth discount')
```

```
Youth discount
```

Good coding

```
1 age = 18
2 if age < 20 :
3     print('Youth discount')
```

```
Youth discount
```

One Step Further

Rules for indented blocks

- It is recommended to use the space key for indentation rather than tabs in Python. The size of indentation is recommended to be four spaces according to the Pythonic way.
- Blocks can be written on multiple lines. If there are multiple lines of commands, the indentation level should be the same for all of them.

One Step Further

Rules for indented blocks

IndentationError

```
1 age = 18
2 if age < 20 :
3     print('Youth Age', age)
4    print('Welcome')
5   print('Youth discount')
```

```
File <tokenize>:4
  print('Welcome')
  ^
IndentationError: unindent does not match any outer indentation level
```

2. Conditional Statements

2.4 The 'pass' Keyword

- The pass statement in Python is used to skip the code within a block.
- It is primarily used when you want to write the code later because the precise processing details are not yet available.

```
1 age = 18
2 if age < 20 :
3     pass
```

2. Conditional Statements

2.4 The 'pass' Keyword

- The pass statement in Python is used to skip the code within a block.
- It is primarily used when you want to write the code later because the precise processing details are not yet available.

```
1 age = 18
2 if age < 20 :
3     pass
```

Line 3

- It serves the purpose of leaving it empty for future implementation.

3. Various Conditional Statement Examples

3.1 Checking for value equality

Focus

- = operator and == operator are different operators.

- Although it is repeatedly mentioned, it is a common mistake made by beginners in coding.
- The = operator is used for assignment, i.e., assigning the value on the right-hand side to the variable on the left-hand side. On the other hand, the == operator is used to compare whether the values on both sides are equal or not, returning either true or false.

3. Various Conditional Statement Examples

3.1 Checking for value equality

- You can implement the code to check if the input string matches 'my_id' using the == operator as follows. In other words, you can determine the equality between the stored string and the input string using ==.

```
1 my_id = 'david'
2
3 s = input('Enter your id: ')
4 if s == my_id :
5     print('ID matches.')
```

```
Enter your id: david
ID matches.
```

3. Various Conditional Statement Examples

3.1 Checking for value equality

- You can implement the code to check if the input string matches 'my_id' using the == operator as follows. In other words, you can determine the equality between the stored string and the input string using ==.

```
1 my_id = 'david'
2
3 s = input('Enter your id: ')
4 if s == my_id :
5     print('ID matches.')
```

```
Enter your id: david
ID matches.
```

If you enter the string 'david', the two strings s and 'my_id' will match, and Line 5 will be executed.
If a different string is entered, Line 5 will not be executed.

3. Various Conditional Statement Examples

3.2 Checking for multiples of 3

- Let's write code to check if a number is a multiple of 3 based on user input.

```
1 number = int(input('Enter an integer : '))
2 if (number % 3) == 0 :
3     print(number, 'is a multiple of 3.')
```

```
Enter an integer : 6
6 is a multiple of 3.
```

3. Various Conditional Statement Examples

3.2 Checking for multiples of 3

- Let's write code to check if a number is a multiple of 3 based on user input.

```
1 number = int(input('Enter an integer : '))
2 if (number % 3) == 0 :
3     print(number, 'is a multiple of 3.')
```

```
Enter an integer : 6
6 is a multiple of 3.
```

Line 2

- If the remainder of dividing the number by 3 is zero, Line 3 will be executed.
- You can determine whether the number is a multiple of 3 using this approach.

3. Various Conditional Statement Examples

3.3 Checking for multiples of 3 and 5 simultaneously

- Let's check if a number is both a multiple of 3 and 5.

```
1 number = int(input('Enter an integer : '))
2 if (number % 3) == 0 and (number % 5) == 0:
3     print(number, 'is a multiple of 3 and a multiple of 5.')
```

```
Enter an integer : 15
15 is a multiple of 3 and a multiple of 5.
```

3. Various Conditional Statement Examples

3.3 Checking for multiples of 3 and 5 simultaneously

- Let's check if a number is both a multiple of 3 and 5.

```
1 number = int(input('Enter an integer : '))
2 if (number % 3) == 0 and (number % 5) == 0:
3     print(number, 'is a multiple of 3 and a multiple of 5.')
```

```
Enter an integer : 15
15 is a multiple of 3 and a multiple of 5.
```

Line 2

- If the remainder of dividing the number by 3 is zero and the remainder of dividing it by 5 is also zero, Line 3 will be executed.
- You can determine whether the number is both a multiple of 3 and 5 using this approach.

3. Various Conditional Statement Examples

3.4 Checking for even number

- Let's examine the code for checking even numbers.

```
1 n = int(input("Enter an integer : "))
2 print("n =", n)
3 if n % 2 == 0 :
4     print(n, "is an even number.")
```

```
Enter an integer : 50
n = 50
50 is an even number.
```

3. Various Conditional Statement Examples

3.4 Checking for even number

- Let's examine the code for checking even numbers.

```
1 n = int(input("Enter an integer : "))
2 print("n =", n)
3 if n % 2 == 0 :
4     print(n, "is an even number.")
```

```
Enter an integer : 50
n = 50
50 is an even number.
```

Line 3

- It checks whether the remainder of dividing the input n by 2 is zero. If the input value is 50, this expression will evaluate to True, and Line 4 will be executed.

3. Various Conditional Statement Examples

3.5 Checking for string equality

- Let's determine if two strings are equal.

```
1 str1 = 'aaa'
2 str2 = 'bbb'
3 if str1 == str2 :
4     print('The two strings are the same.')
5 if str1 != str2 :
6     print('The two strings are not the same.')
```

```
The two strings are not the same.
```

3. Various Conditional Statement Examples

3.5 Checking for string equality

- Let's determine if two strings are equal.

```
1 str1 = 'aaa'
2 str2 = 'bbb'
3 if str1 == str2 :
4     print('The two strings are the same.')
5 if str1 != str2 :
6     print('The two strings are not the same.')
```

```
The two strings are not the same.
```

Line 3

- If the values of the two strings are the same, Line 4 will be executed. In this case, since the values of str1 and str2 are different, it will not be printed.

3. Various Conditional Statement Examples

3.5 Checking for string equality

- Let's determine if two strings are equal.

```
1 str1 = 'aaa'
2 str2 = 'bbb'
3 if str1 == str2 :
4     print('The two strings are the same.')
5 if str1 != str2 :
6     print('The two strings are not the same.')
```

```
The two strings are not the same.
```

Line 5

- If the values of the two strings are different, Line 6 will be executed. In this case, since the values of str1 and str2 are different, it will be printed.

3. Various Conditional Statement Examples

3.6 Determining pass or fail based on scores

- Let's output whether a person passes the test and can receive a scholarship if their score is 90 or above.

```
1 score = int(input("Enter your score: "))
2 if score >= 90 :
3     print("Congratulations.")
4     print("You passed.")
5     print("You can also get a scholarship.")
```

```
Enter your score: 95
Congratulations.
You passed.
You can also get a scholarship.
```

3. Various Conditional Statement Examples

3.6 Determining pass or fail based on scores

- Let's output whether a person passes the test and can receive a scholarship if their score is 90 or above.

```
1 score = int(input("Enter your score: "))
2 if score >= 90 :
3     print("Congratulations.")
4     print("You passed.")
5     print("You can also get a scholarship.")
```

```
Enter your score: 95
Congratulations.
You passed.
You can also get a scholarship.
```

Line 2

- If the score input is 95, Lines 3, 4, and 5 will all be executed.

3. Various Conditional Statement Examples

3.7 Examples using logical operator

Focus

- Real programmers need to deal with more complex problems than this.
- To impose more sophisticated conditions, you can combine comparison operators and logical operators. This is called “logical operation”.
- Let’s look at an example of combining multiple operators.

3. Various Conditional Statement Examples

3.7 Examples using logical operator

```
1 a = 10
2 b = 14
3 if (a % 2 == 0) and (b % 2 == 0) :
4     print('Both numbers are even numbers.')
5 if (a % 2 == 0) or (b % 2 == 0) :
6     print('One or more of the two numbers are even.')
```

```
Both numbers are even numbers.
One or more of the two numbers are even.
```

- When a = 10 and b = 14, the following code prints the output based on two conditional statements.
- The if statement in this code uses logical operators called “and” and “or” to perform the output.
- In this code, (a % 2 == 0) is true, and (b % 2 == 0) is also true. Therefore, both print statements in Lines 4 and 6 will be executed.

3. Various Conditional Statement Examples

3.7 Examples using logical operator

- Let's modify the previous code by assigning 13 to ‘b’.

```
1 a = 10
2 b = 13
3 if (a % 2 == 0) and (b % 2 == 0) :
4     print('Both numbers are even numbers.')
5 if (a % 2 == 0) or (b % 2 == 0) :
6     print('One or more of the two numbers are even.')
```

```
One or more of the two numbers are even.
```

- In this case, ‘b’ is 13, so (b % 2 == 0) is false.
- Therefore, (a % 2 == 0) and (b % 2 == 0) becomes false, and Line 4 will not be printed.
- On the other hand, (a % 2 == 0) is true, so (a % 2 == 0) or (b % 2 == 0) becomes true.