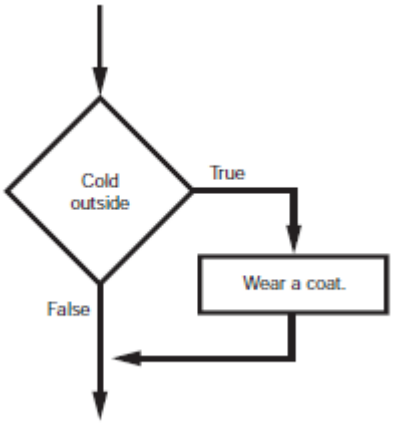
Decision Structure and Boolean Logic Python

If Statement

The if statement is used to create a decision structure, which allows a program to have more than one path of execution. The if statement causes one or more statements to execute only when a Boolean expression is true.



Simple Decision Structure

```
if condition:
    statement
    statement
    etc.
```

General Format of Decision Structure

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Relational Operators

1) Check the Operators
   x=1
   x>=1 ← This is an expression
   True
   Use of if
   sales=5000
   if sales==5000:
     print(sales)
2) Perform the problem. Calculate the average Test Score.
   high score=95

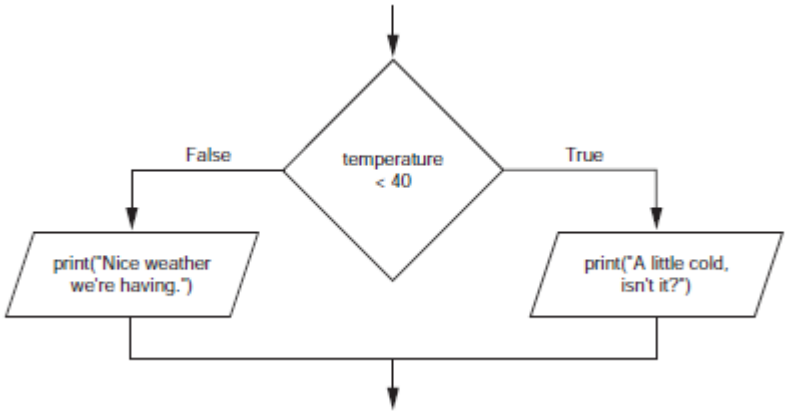
```
test1 = int(input('Enter the score for test 1:' ))
test2 = int(input('Enter the score for test 2:' ))
test3 = int(input('Enter the score for test 3:' ))
# Calculate the average test score.
average = (test1 + test2 + test3) / 3

# Print the average.
print('The average score is', average)
```

```
# If the average is a high score,
 # congratulate the user.
 if average >= high_score:
    print('Congratulations!')
    print('That is a great average!')
```

3) If-else statement

An if-else statement will execute one block of statements if its condition is true, or another block if its condition is false.




```
if condition:
    statement
    statement
    etc.
else:
    statement
    statement
    etc.
```

General format of if-else Structure

Eg.

```
sales=5000
if sales==5000:
   print(sales)
else:
   print("this is alternative")
```

4) Comparing Strings

Python allows you to compare strings. This allows you to create decision structures that test the value of a string.

```
>>> name1='mary'
>>> name2='mark'
>>> if name1==name2:
        print('the names are the same')
    else:
        print('the names are not same')
```

Nested if-elif-else statement **To test more than one condition, a decision structure can be nested**
inside another decision structure.

```
number = int(input('Please input a number in the range of 1 through 7: '))

if number == 1:
   print('Monday')
elif number == 2:
   print('Tuesday')
elif number == 3:
   print('Wednesday')
elif number == 4:
   print('Thursday')
elif number == 5:
   print('Friday')
elif number == 6:
   print('Saturday')
elif number ==7:
   print('Sunday')
else:
   print('Error')
```

Examples:

```
mass = float(input('Please enter an object\'s mass: '))

weight = mass * 9.8

if weight > 500:
   print('It is too heavy.')
elif weight < 100:
   print('It is too light.')
else:
   print(weight, 'N', sep='')
```

5) Logical operators

Operator	Meaning
and	The and operator connects two Boolean expressions into one compound expression. Both subexpressions must be true for the compound expression to be true.
or	The or operator connects two Boolean expressions into one compound expression. One or both subexpressions must be true for the compound expression to be true. It is only necessary for one of the subexpressions to be true, and it does not matter which.
not	The not operator is a unary operator, meaning it works with only one operand. The operand must be a Boolean expression. The not operator reverses the truth of its operand. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true.

```
>>> if not(temp>100):
                print("this is true")
```

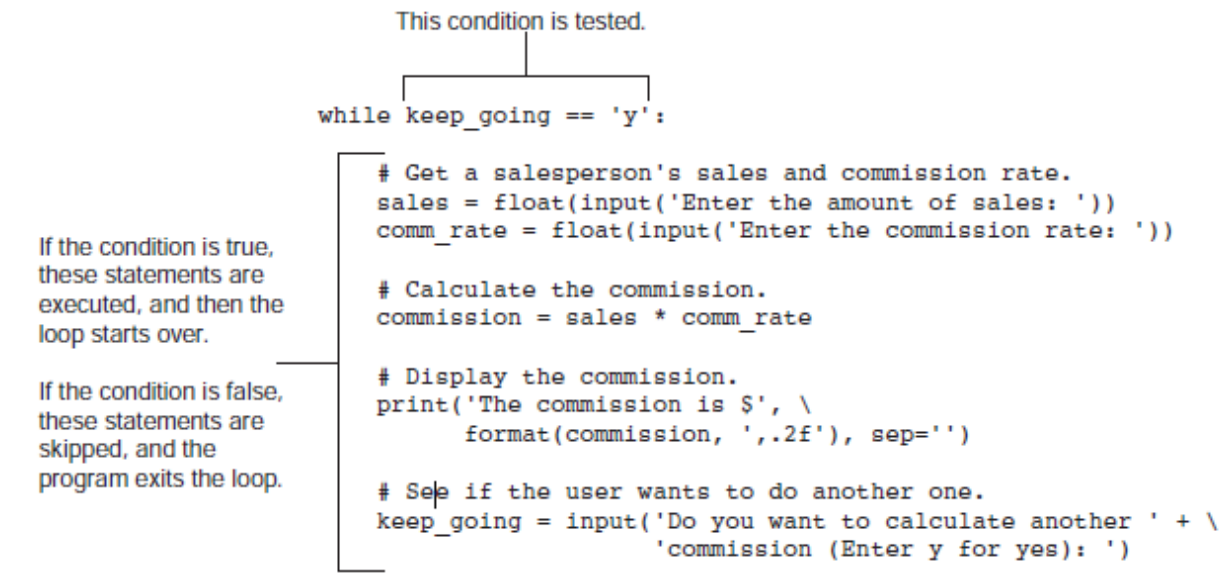
6) **Short Circuit Evaluations** If ist sub expression if false other sub expression will not checked
But in OR if ist expression is true other will not checked.

Boolean Variable
A Boolean Variable can reference one of two values: True or False
A flag is a variable that signals when
some condition exists in the program. When the flag variable is set to False, it indicates the condition does not exist. When the flag variable is set to True, it means the condition does exist.

```
Sales=4000.0
if Sales>=5000.0:
   sales_quota=True
else:
   sales_quota=false
```

7) Nested Loops
A condition-controlled loop uses a true/false condition to control the number of times that it repeats. A count-controlled loop repeats a specific number of times

```
while condition:
     statement
     statement
     etc.
```




```
number = float(input('Enter a positive number or enter a negative number '
                'if you want to end: '))
total = 0
```

```
        while number > 0:
            total += number
            number = float(input('Enter a positive number: '))
        print('The sum is', total)
```

Example to perform

```
# This program assists a technician in the process
 # of checking a substance's temperature.

 # Create a variable to represent the maximum
 # temperature.
max_temp = 102.5

 # Get the substance's temperature.
temperature = float(input("Enter the substance's Celsius temperature: "))

# As long as necessary, instruct the user to
 # adjust the thermostat.
while temperature > max_temp:
    print('The temperature is too high.')
    print('Turn the thermostat down and wait')
    print('5 minutes. Then take the temperature')
    print('again and enter it.')
    temperature = float(input('Enter the new Celsius temperature:'))



 # Remind the user to check the temperature again
# in 15 minutes.
print('The temperature is acceptable.')
print('Check it again in 15 minutes.')
```

8) For Loop : Count controlled Loop

```
for variable in [value1, value2, etc.]:
    statement
    statement
    etc.
```

Example

```
print('I will display the numbers 1 through 5.')
for num in [1, 2, 3, 4, 5]:
 print(num)
```

9) Using Range functions with for loop

```
>>> for x in range(5):
        print(x)
Eg. for number in range(1,11):
        square=number**2
        print(number, '\t', square)
```

Operator	Example Usage	Equivalent To
+=	x += 5	x = x + 5
-=	y -= 2	y = y - 2
*=	z *= 10	z = z * 10
/=	a /= b	a = a / b
%=	c %= 3	c = c % 3

```
>>> for num in range(1,10,2):
        print(num)
```

Highest to Lowest

range(10,0,-1)

10, 9 , 8, 7, 6 ,5,4,3,2,1

10) Nested Loops

A loop inside another loop is nested loop

```
>>> for hours in range(24):
        for minutes in range(60):
            for seconds in range(60):
                print(hours,':', minutes,':', seconds)
```

11) Perform the Problem

Print the following

```
**********
**********
**********
**********
**********
```

12) Functions

```
>>> def message():
        print('hello')
        print('Friend')
message()
```