



# **PROGRAMMING WITH PYTHON**

Karuna sheel

NIELIT kurukshetra

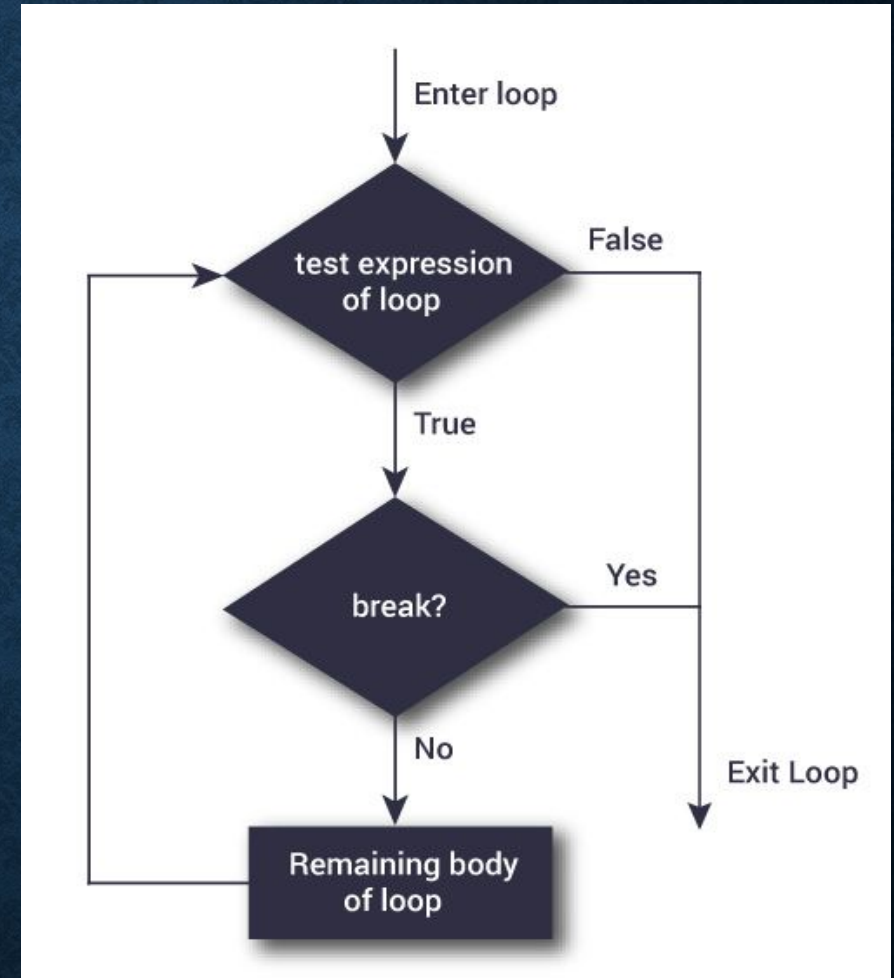
# PYTHON BREAK STATEMENT

The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. In other words, we can say that break is used to abort the current execution of the program and the control goes to the next line after the loop.

The break is commonly used in the cases where we need to break the loop for a given condition.

The syntax of the break is given below.


```
#loop statements  
break;
```





# WORKING OF BREAK STATEMENT IN FOR AND WHILE LOOP


```
for var in sequence:  
    # codes inside for loop  
    if condition:  
        break
```



```
    # codes inside for loop  
# codes outside for loop
```

---

```
while test expression:  
    # codes inside while loop  
    if condition:  
        break
```



```
    # codes inside while loop  
# codes outside while loop
```

### Example 1

```
i = 0;
while 1:
    print(i, " ", end=""),
    i=i+1;
    if i == 10:
        break;
print("came out of while loop");
```

Output:

```
0 1 2 3 4 5 6 7 8 9 came out of while loop
```

### Example 2

```
for letter in 'Python':    # First Example
    if letter == 'h':
        break
    print 'Current Letter :', letter
```

```
var = 10                # Second Example
while var > 0:
    print 'Current variable value :', var
    var = var -1
    if var == 5:
        break

print "Good bye!"
```



### Example 3

```
n=2
while 1:
    i=1;
    while i<=10:
        print("%d X %d = %d\n"%(n,i,n*i));
        i = i+1;
    choice = int(input("Do you want to continue printing the table, press 0 for no?"))
    if choice == 0:
        break
    n=n+1
```



# EXAMPLES

## # Program to check if a number is prime or not

```
num = int(input("Enter a number: "))
```

```
# define a flag variable
```

```
flag = False
```

```
# prime numbers are greater than 1
```

```
if num > 1:
```

```
    # check for factors
```

```
    for i in range(2, num):
```

```
        if (num % i) == 0:
```

```
            # if factor is found, set flag to True
```

```
            flag = True
```

```
            # break out of loop
```

```
            break
```

```
# check if flag is True
```

```
if flag:
```

```
    print(num, "is not a prime number")
```

```
else:
```

```
    print(num, "is a prime number")
```



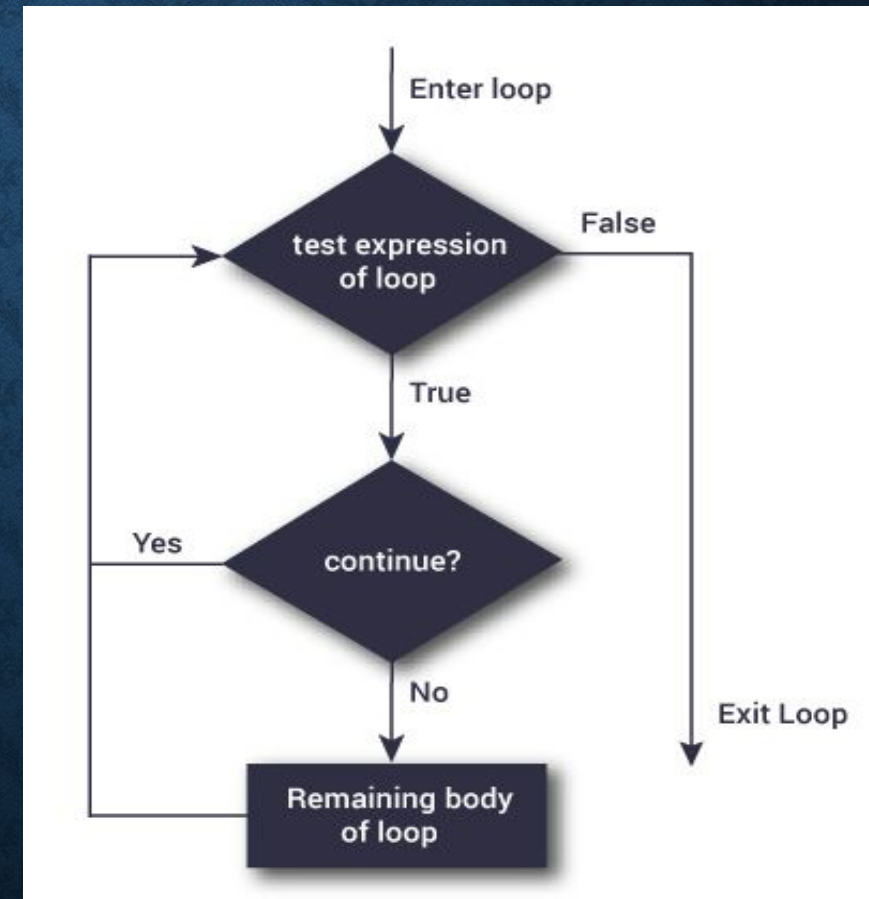
# PYTHON CONTINUE STATEMENT

The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.

Syntax of Continue

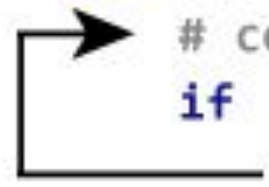
`continue`

## FLOWCHART OF CONTINUE



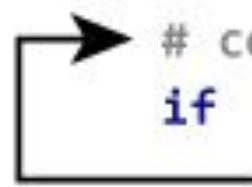
# THE WORKING OF CONTINUE STATEMENT IN FOR AND WHILE LOOP IS SHOWN BELOW.

```
for var in sequence:  
    # codes inside for loop  
    if condition:  
        continue  
    # codes inside for loop  
  
# codes outside for loop
```



---

```
while test expression:  
    # codes inside while loop  
    if condition:  
        continue  
    # codes inside while loop  
  
# codes outside while loop
```





# EXAMPLE

```
i=1; #initializing a local variable
```

```
#starting a loop from 1 to 10
```

```
for i in range(1,11):
```

```
    if i==5:
```

```
        continue
```

```
    print("%d"%i)
```

Output:

1  
2  
3  
4  
6  
7  
8  
9  
10



# PASS STATEMENT

- The pass statement is a null operation since nothing happens when it is executed. It is used in the cases where a statement is syntactically needed but we don't want to use any executable statement at its place.
- For example, it can be used while overriding a parent class method in the subclass but don't want to give its specific implementation in the subclass.
- Pass is also used where the code will be written somewhere but not yet written in the program file.

The syntax of the pass statement is given as

```
list = [1,2,3,4,5]
flag = 0
for i in list:
    print("Current element:",i,end=" ")
    if i==3:
        pass;
        print("\nWe are inside pass block\n");
        flag = 1;
    if flag==1:
        print("\nCame out of pass\n");
        flag=0;
```

Output:

Current element: 1 Current element: 2 Current element: 3

We are inside pass block

Came out of pass

Current element: 4 Current element: 5



# PYTHON STRING

Till now, we have discussed numbers as the standard data types in python. In this section of the tutorial, we will discuss the most popular data type in python i.e., string.

In python, strings can be created by enclosing the character or the sequence of characters in the quotes. Python allows us to use single quotes, double quotes, or triple quotes to create the string.

Consider the following example in python to create a string.

```
str = "Hi Python !"
```

Here, if we check the type of the variable str using a python script

```
print(type(str)), then it will print string (str).
```

In python, strings are treated as the sequence of strings which means that python doesn't support the character data type instead a single character written as 'p' is treated as the string of length 1.



# STRINGS INDEXING AND SPLITTING

Like other languages, the indexing of the python strings starts from 0. For example, The string "HELLO" is indexed as given in the below figure.

str = "HELLO"				
H	E	L	L	O
0	1	2	3	4
str[0] = 'H'				
str[1] = 'E'				
str[2] = 'L'				
str[3] = 'L'				
str[4] = 'O'				



As shown in python, the slice operator `[]` is used to access the individual characters of the string. However, we can use the `:` (colon) operator in python to access the substring. Consider the following example.

str = "HELLO"				
H	E	L	L	O
0	1	2	3	4
str[0] = 'H'		str[:] = 'HELLO'		
str[1] = 'E'		str[0:] = 'HELLO'		
str[2] = 'L'		str[:5] = 'HELLO'		
str[3] = 'L'		str[:3] = 'HEL'		
str[4] = 'O'		str[0:2] = 'HE'		
		str[1:4] = 'ELL'		

Here, we must notice that the upper range given in the slice operator is always exclusive i.e., if `str = 'python'` is given, then `str[1:3]` will always include `str[1] = 'p'`, `str[2] = 'y'`, `str[3] = 't'` and nothing else.



# REASSIGNING STRINGS

Updating the content of the strings is as easy as assigning it to a new string. The string object doesn't support item assignment i.e., A string can only be replaced with a new string since its content can not be partially replaced. Strings are immutable in python.

Consider the following example.

## Example 1

```
str = "HELLO"  
str[0] = "h"  
print(str)
```

## Output:

Traceback (most recent call last):

File "12.py", line 2, in <module>

str[0] = "h";

TypeError: 'str' object does not support item assignment



# STRING OPERATORS

Operator	Description
+	It is known as concatenation operator used to join the strings given either side of the operator.
*	It is known as repetition operator. It concatenates the multiple copies of the same string.
[]	It is known as slice operator. It is used to access the sub-strings of a particular string.
[:]	It is known as range slice operator. It is used to access the characters from the specified range.
in	It is known as membership operator. It returns if a particular sub-string is present in the specified string.
not in	It is also a membership operator and does the exact reverse of in. It returns true if a particular substring is not present in the specified string.
r/R	It is used to specify the raw string. Raw strings are used in the cases where we need to print the actual meaning of escape characters such as "C://python". To define any string as a raw string, the character r or R is followed by the string.
%	It is used to perform string formatting. It makes use of the format specifiers used in C programming like %d or %f to map their values in python. We will discuss how formatting is done in python.



# EXAMPLE

```
str = "Hello"  
str1 = " world"  
print(str*3) # prints HelloHelloHello  
print(str+str1)# prints Hello world  
print(str[4]) # prints o  
print(str[2:4]); # prints ll  
print('w' in str) # prints false as w is not present in str  
print('wo' not in str1) # prints false as wo is present in str1.  
print(r'C://python37') # prints C://python37 as it is written  
print("The string str : %s"%(str)) # prints The string str : Hello
```

Output:

HelloHelloHello

Hello world

o

ll

False

False

C://python37

The string str : Hello



# PYTHON FORMATTING OPERATOR

Python allows us to use the format specifiers used in C's printf statement. The format specifiers in python are treated in the same way as they are treated in C. However, Python provides an additional operator % which is used as an interface between the format specifiers and their values. In other words, we can say that it binds the format specifiers to the values.

Consider the following example.

```
Integer = 10;
```

```
Float = 1.290
```

```
String = "Ayush"
```

```
print("Hi I am Integer ... My value is %d\n Hi I am float ... My value is %f\n Hi I am string ... My value is %s"%(Integer,Float,String));
```

Output:

```
Hi I am Integer ... My value is 10
```

```
Hi I am float ... My value is 1.290000
```

```
Hi I am string ... My value is Ayush
```