

Python Keywords

Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers:

and Keyword

The and keyword is a logical operator.

Logical operators are used to combine conditional statements.

The return value will only be True if both statements return True, otherwise it will return False.

1st example

```
x = (5 > 3 and 5 < 10)      print(x)      result True
```

```
x = (3 > 5 and 10 < 5)      print(x)      result True
```

2nd example

```
if 5 > 3 and 5 < 10: print("Both statements are True")
```

```
else:    print("At least one of the statements are False")    result Both statements are True.
```

as Keyword

The as keyword is used to create an alias.

In the example above, we create an alias, `c`, when importing the calendar module, and now we can refer to the calendar module by using `c` instead of `calendar`.

```
import calendar as c      print(c.month_name[1])      result January
```

assert Keyword

```
x = "welcome"
```

```
assert x != "hello", "x should be 'hello'"
```

```
#if condition returns False,      result AssertionError is raised:
```

break Keyword

The break keyword is used to break out a for loop, or a while loop.

```
i = 1
```

```
while i < 9: print(i)
```

```
    if i == 3:    break    i += 1      result 1 2 3
```

class Keyword

The class keyword is used to create a class.

A class is like an object constructor. See the example below to see how we can use it to create an object.

```
class Person:    name = "John"    age = 36
```

```
p1 = Person() print(p1.name)      result John
```

continue Keyword

The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

```
for i in range(9): if i == 3:
    continue print(i)
if i == 3: continue print(i) for i in range(9):
result
0
1
2
4
5
6
7
8
```

def Keyword

The def keyword is used to create, (or define) a function.

```
def my_function(): print("Hello from a function") my_function()
result Hello from a function
```

del Keyword

The del keyword is used to delete objects. In Python everything is an object, so the del keyword can also be used to delete variables, lists, or parts of a list etc.

```
x = ["apple", "banana", "cherry"] del x[0] print(x) result ['banana', 'cherry']
```

elif Keyword

The elif keyword is used in conditional statements (if statements), and is short for else if.

```
for i in range(-5, 5): if i > 0: print("YES")
elif i == 0: print("WHATEVER")
else: print("NO")
result
NO
NO
NO
NO
NO
NO
WHATEVER
YES
```

else Keyword

The else keyword is used in conditional statements (if statements), and decides what to do if the condition is False.

The else keyword can also be use in try...except blocks, see example below.

```
x = 2
if x > 3: print("YES")
else: print("NO")          result NO
```

except Keyword

The except keyword is used in try...except blocks. It defines a block of code to run if the try block raises an error.

You can define different blocks for different error types, and blocks to execute if nothing went wrong, see examples below.

```
x = "hello" try: x > 3
except NameError: print("You have a variable that is not defined.")
except TypeError: print("You are comparing values of different type")
result You are comparing values of different type
x = "hello" try: x > hello
result You have a variable that is not defined
```

False Keyword

The False keyword is a Boolean value, and result of a comparison operation.

The False keyword is the same as 0 (True is the same as 1).

```
print(5 > 6)          result False
print(4 in [1,2,3])   result False
print("hello" is "goodbye") result False
```

Finally Keyword

The finally keyword is used in try...except blocks. It defines a block of code to run when the try...except...else block is final.

The finally block will be executed no matter if the try block raises an error or not.

This can be useful to close objects and clean up resources.

```
try: x > 3
except: print("Something went wrong")
else: print("Nothing went wrong")
finally: print("The try...except block is finished")
```

Result

Something went wrong

The try...except block is finished

for Keyword

The for keyword is used to create a for loop.

It can be used to iterate through a sequence, like a list, tuple, etc.

```
for x in range(1, 9): print(x)
```

result

1

2

3

4

5

6

7

8

```
fruits = ["apple", "banana", "cherry"]
```

```
for x in fruits: print(x)
```

result

apple, banana, cherry

from Keyword

The from keyword is used to import only a specified section from a module.

```
from datetime import time
```

```
x = time(hour=15)
```

```
print(x)
```

result 15:00:00

global Keyword

The global keyword is used to create global variables from a no-global scope, e.g. inside a function.

#create a function:

```
def myfunction(): global x x = "hello"
```

#execute the function:

```
myfunction()
```

#x should now be global, and accessible in the global scope.

```
print(x)
```

Result hello