PYTHON KEYWORDS

**Python Keywords:-**

Python Keywords are special reserved words that convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as a variable. Following is the List of Python Keywords.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| True | False | None | and | as |
| asset | def | class | continue | break |
| else | finally | elif | del | except |
| global | for | if | from | import |
| raise | try | or | return | pass |
| nonlocal | in | not | is | lambda |

Consider the following explanation of keywords.

1. **True -** It represents the Boolean true, if the given condition is true, then it returns "True". Non-zero values are treated as true.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A and B** |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

1. **False -** It represents the Boolean false; if the given condition is false, then it returns "False". Zero value is treated as false

**3. None -** It denotes the null value or void. An empty list or Zero can't be treated as **None**.

**4. and -** It is a logical operator. It is used to check the multiple conditions. It returns true if both conditions are true. Consider the following truth table.

**5. or** - It is a logical operator in Python. It returns true if one of the conditions is true. Consider the following truth table.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A and B** |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

**6. not** - It is a logical operator and inverts the truth value. Consider the following truth table.

|  |  |
| --- | --- |
| **A** | **Not A** |
| True | False |
| False | True |

**7. assert -** This keyword is used as the debugging tool in Python. It checks the correctness of the code. It raises an **AssertionError** if found any error in the code and also prints the message with an error.

**Example:**

**a = 10**

**b = 0**

**print('a is dividing by Zero')**

**assert b != 0**

**print(a / b)**

**Output:**

**a is dividing by Zero**

**Runtime Exception:**

**Traceback (most recent call last):**

**File "/home/40545678b342ce3b70beb1224bed345f.py", line 4, in**

**assert b != 0**

**AssertionError**

**8. def -** This keyword is used to declare the function in Python. If followed by the function name.

**def my\_func(a,b):**

**c = a+b**

**print(c)**

**my\_func(10,20)**

**Output:**

**30**

**9. class -** It is used to represents the class in Python. The class is the blueprint of the objects. It is the collection of the variable and methods. Consider the following class.

**class Myclass: #class definition**

**greet = "Hello "**

**def greet\_func(self, name):**

**print(self.greet + name)**

**obj = Myclass() #creating class object**

**obj.greet\_func("Krishna") #call class method using object**

**Output:**

**Hello Krishna**

**10. continue -** It is used to stop the execution of the current iteration. Consider the following example.

**a = 0**

**while a < 4:**

**a += 1**

**if a == 2:**

**continue**

**print(a)**

**Output:**

**1**

**3**

**4**

**11. break -** It is used to terminate the loop execution and control transfer to the end of the loop. Consider the following example.

**Example**

**for i in range(5):**

**if(i==3):**

**break**

**print(i)**

**print("End of execution")**

**Output:**

**0**

**1**

**2**

**End of execution**

**12. If -** It is used to represent the conditional statement. The execution of a particular block is decided by if statement. Consider the following example.

**Example**

**i = 12**

**if (i < 18):**

**print("I am less than 18")**

**Output:**

**I am less than 18**

**13. else -** The else statement is used with the if statement. When if statement returns false, then else block is executed. Consider the following example.

**Example:**

**n = 11**

**if(n%2 == 0):**

**print("Even")**

**else:**

**print("odd")**

**Output:**

**Odd**

**14. elif -** This Keyword is used to check the multiple conditions. It is short for **else-if**. If the previous condition is false, then check until the true condition is found. Condition the following example.

**Example:**

**marks = int(input("Enter the marks:"))**

**if(marks>=90):**

**print("Excellent")**

**elif(marks<90 and marks>=75):**

**print("Very Good")**

**elif(marks<75 and marks>=60):**

**print("Good")**

**else:**

**print("Average")**

**Output:**

**Enter the marks:85**

**Very Good**

**15. del** - It is used to delete the reference of the object. Consider the following example.

**Example:**

**a=10**

**b=12**

**del a**

**print(b)**

**# a is no longer exist**

**print(a)**

**Output:**

**12**

**NameError: name 'a' is not defined**

**16. try, except -** The try-except is used to handle the exceptions. The exceptions are run-time errors. Consider the following example.

**Example:**

**a = 0**

**try:**

**b = 1/a**

**except Exception as e:**

**print(e)**

**Output:**

**division by zero**

**17. raise -** The raise keyword is used to through the exception forcefully. Consider the following example.

**Example**

**a = 5**

**if (a>2):**

**raise Exception('a should not exceed 2 ')**

**Output:**

**Exception: a should not exceed 2**

**18. finally -** The **finally** keyword is used to create a block of code that will always be executed no matter the else block raises an error or not. Consider the following example.

**Example:**

**a=0**

**b=5**

**try:**

**c = b/a**

**print(c)**

**except Exception as e:**

**print(e)**

**finally:**

**print('Finally always executed')**

**Output:**

division by zero

Finally always executed

**19. for, while -** Both keywords are used for iteration. The **for** keyword is used to iterate over the sequences (list, tuple, dictionary, string). A while loop is executed until the condition returns false. Consider the following example.

**Example: For loop**

**list = [1,2,3,4,5]**

**for i in list:**

**print(i)**

**Output:**

1

2

3

4

5

**Example: While loop**

**a = 0**

**while(a<5):**

**print(a)**

**a = a+1**

**Output:**

**0**

**1**

**2**

**3**

**4**

**20. import -** The import keyword is used to import modules in the current Python script. The module contains a runnable Python code.

**Example:**

**import math**

**print(math.sqrt(25))**

**Output:**

**5**

**21. from -** This keyword is used to import the specific function or attributes in the current Python script.

**Example:**

**from** math **import** sqrt

**print**(sqrt(25))

**Output:**

**5**

**22. as -** It is used to create a name alias. It provides the user-define name while importing a module.

**Example:**

**import calendar as cal**

**print(cal.month\_name[5])**

**Output:**

**May**

**23. pass -** The **pass** keyword is used to execute nothing or create a placeholder for future code. If we declare an empty class or function, it will through an error, so we use the pass keyword to declare an empty class or function.

**Example:**

**class Myclass:**

**#we cal also use pass here if we want to left class empty**

**def newfunc(self):**

**pass**

**print("After pass")**

**obj = Myclass()**

**obj.newfunc()**

**Output:**

**After pass**

**24. return -** The **return** keyword is used to return the result value or none to called function.

**Example:**

**def sum(a,b):**

**c = a+b**

**return c**

**print("The sum is:",sum(25,15))**

**Output:**

**The sum is: 40**

**25. is -** This keyword is used to check if the two-variable refers to the same object. It returns the true if they refer to the same object otherwise false. Consider the following example.

**Example**

**x = 5**

**y = 5**

**a = []**

**b = []**

**print(x is y)**

**print(a is b)**

**Output:**

**True**

**False**

**Note: A mutable data-types do not refer to the same object.**

**26. global -** The global keyword is used to create a global variable inside the function. Any function can access the global. Consider the following example.

**Example**

**def my\_func():**

**global a**

**a = 10**

**b = 20**

**c = a+b**

**print(c)**

**my\_func()**

**def func():**

**print(a)**

**func()**

**Output:**

**30**

**10**

**27. nonlocal -** The **nonlocal** is similar to the **global** and used to work with a variable inside the nested function(function inside a function). Consider the following example.

**Example**

**def outside\_function():**

**a = 20**

**def inside\_function():**

**nonlocal a**

**a = 30**

**print("Inner function: ",a)**

**inside\_function()**

**print("Outer function: ",a)**

**outside\_function()**

**Output:**

**Inner function: 30**

**Outer function: 30**

**28. lambda -** The lambda keyword is used to create the anonymous function in Python. It is an inline function without a name. Consider the following example.

**Example**

**a = lambda x: x\*\*2**

**for i in range(1,6):**

**print(a(i))**

**Output:**

**1**

**4**

**9**

**16**

**25**

**29. yield -** The **yield** keyword is used with the Python generator. It stops the function's execution and returns value to the caller. Consider the following example.

**Example**

**def fun\_Generator():**

**yield 1**

**yield 2**

**yield 3**

**# Driver code to check above generator function**

**for value in fun\_Generator():**

**print(value)**

**Output:**

**1**

**2**

**3**

**30. with -** The **with** keyword is used in the exception handling. It makes code cleaner and more readable. The advantage of using **with**, we don't need to call **close()**. Consider the following example.

**Example**

**with open('file\_path', 'w') as file:**

**file.write('hello world !')**

**31. None -** The None keyword is used to define the null value. It is remembered that **None** does not indicate 0, false, or any empty data-types. It is an object of its data type, which is Consider the following example.

**Example:**

**def return\_none():**

**a = 10**

**b = 20**

**c = a + b**

**x = return\_none()**

**print(x)**

**Output:**

**None**

We have covered all Python keywords. This is the brief introduction of Python Keywords. We will learn more in the upcoming tutorials.