**IDE**

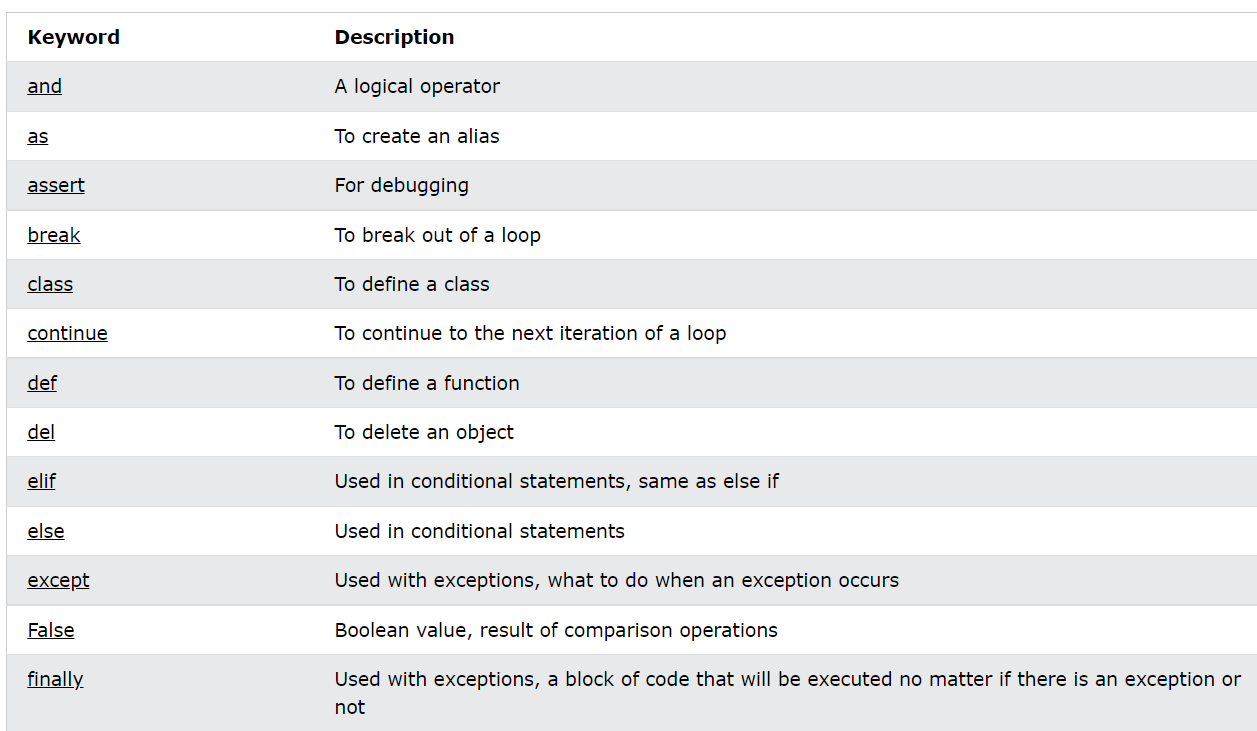
It stands for integrated development environment. An IDE consists of tools that help a programmer throughout the phases of writing, executing and testing a computer program. An IDE has a Graphical User Interface (GUI), meaning that a user can interact with it using windows and buttons to provide input and output. There are different types of IDEs available for development process. Example: Visual Studio, Code Block, Dev C++ etc. PyCharm is cross-platform IDE, hence it can be installed on different operating systems. PyCharm is the product of JetBrains.

To install PyCharm on Windows operating system, visit the link <https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows> to download the executable installer. Double click the installer (.exe) file and install PyCharm by clicking next at each step.

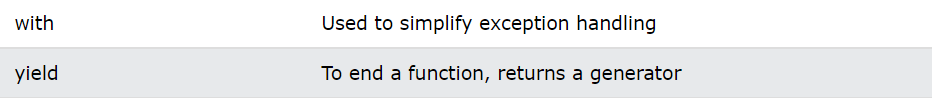
**Keywords**

* reserved words or predefined words
* keywords have its own purpose
* they are used to define the syntax and structure of the python language
* keywords cannot be used as identifiers
* 35 keywords in python

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

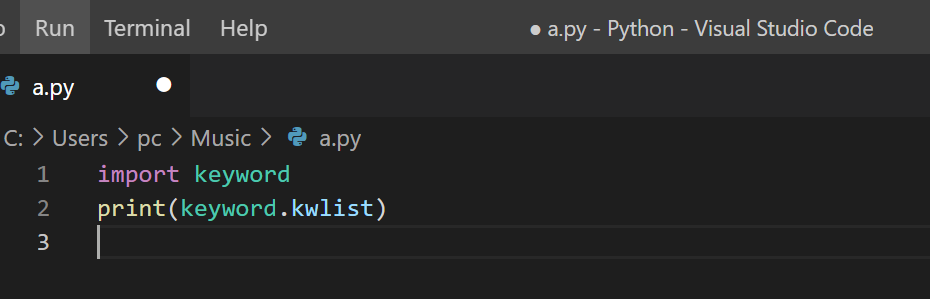




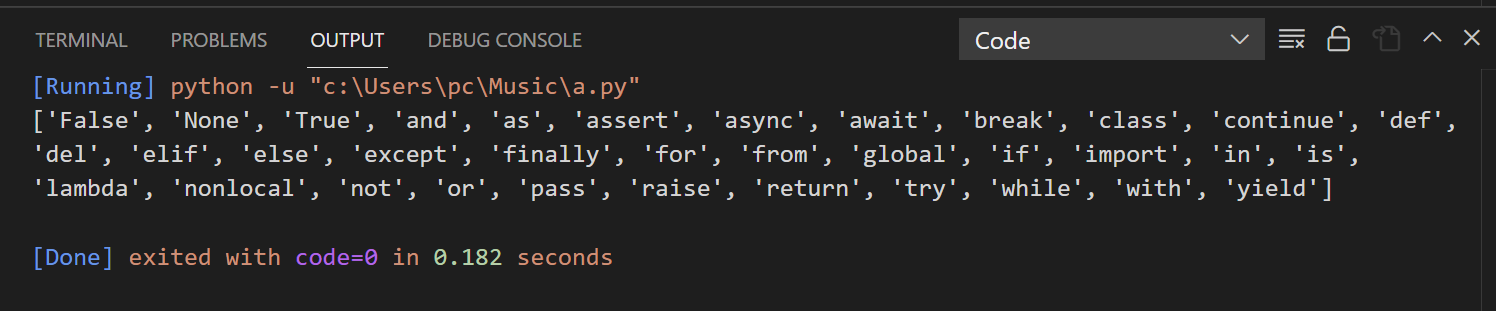


**async and await are also python keywords**

We can see the list and numbers of keyword available in Python by importing keyword module as below:



**OUTPUT**



**Identifiers**

* A python identifier is a name used to define a variable, function etc.
* Combination of characters, digits and underscore
* Python is a case sensitive programming language. Thus, Ram and ram are two different identifiers in Python.

**Identifier naming:**

Variables are the example of identifiers. The rules to name an identifier are given below:

* The first character of the variable must be an alphabet or underscore ( \_ ).
* All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
* Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, \*).
* Identifier name must not be similar to any keyword defined in the language.
* Identifier names are case sensitive; for example, myname, and MyName is not the same.

**Examples of valid identifiers**

a123, \_n, n\_9, etc.

**Examples of invalid identifiers:**

3rdGraph - Name can’t start with a digit

Roll#No - Special symbol # is not allowed

First Name - Spaces are not allowed.

D.O.B. - Special symbol. (dots) are not allowed.

while - Keyword not allowed.

**Variable**

* Variables are the containers to store some data or value
* Eg. x=25, Name=”Nepal”
* The equal (=) operator is used to assign value to a variable.

**Object References:**

Everything is an object. Variables are references. It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.

Consider the following example.

print("John")

output: John

In the above print statement, we have created a string object.

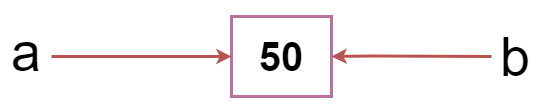
Let's check the type of it using the Python built-in type() function.

type('John')

output: <class -'str'>

In Python, variables are a symbolic name that is a reference

or pointer to an object. The variables are used to denote objects by that name.

a=50                    

b = a

print(a)

print(b)

We assigned the b = a, a and b both point to the same object.

When we checked by the id() function it returned the same number.

We reassign a to 500; then it referred to the new object identifier.

**Object Identity**

In python, every created object identifies uniquely in python. Python provides the guaranteed that no two objects will have the same identifier.

The built-in **id()** function, is used to identify the object identifier. Consider the following example.

a=10

b=20

print(id(a))

print(id(b))

# Reassigned variable a

a = 500

print(id(a))

Output:

140734982691168

140734982691168

2822056960944

**Variable Name:**

We have already discussed how to declare the valid variable. Variable names can be any length can have uppercase, lowercase (A to Z, a to z), the digit (0-9), and underscore character(\_). Consider the following example of valid variables names.

name = "Devansh"

age = 20

marks = 80.50

print(name)

print(age)

print(marks)

Output:

Devansh

20

80.5

**Consider the following valid variables name.**

name = "A"

Name = "B"

naMe = "C"

NAME = "D"

n\_a\_m\_e = "E"

\_name = "F"

name\_ = "G"

\_name\_ = "H"

na56me = "I"

print(name,Name,naMe,NAME,n\_a\_m\_e, NAME, n\_a\_m\_e, \_name, name\_,\_name, na56me)

Output:

A B C D E D E F G F I

In the above example, we have declared a few valid variable names such as name, \_name\_ , etc. But it is not recommended because when we try to read code,

it may create confusion. The variable name should be descriptive to make code more readable.

**Assign values to variables**

* A=10 #An integer assignment
* B=2.28 #A float assignment
* C=” Hello” #A string assignment

**Multiple Assignment**

We can assign a value to multiple variables in a single statement which is also known as multiple assignment. We can give multiple assignments in two ways: -

1. Assigning a single value to multiple variables

**Example: -**

x=y=z=50

print(x)

print(y)

print(z)

1. Assigning multiple values to multiple variables

**Example: -**

a,b,c=5,10,15

print(a)

print(b)

print(c)

**The values will be assigned in the order in which variable appears.**

**The multi-word keywords can be created by the following method.**

**Camel Case -** In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace.

For example - nameOfStudent, valueOfVaraible, etc.

**Pascal Case -** It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.

**Snake Case -** In the snake case, Words are separated by the underscore.

For example - name\_of\_student, etc.

**Memory Allocation**

**Garbage Collection:** When a value in memory is no longer referenced by a variable, the python interpreter automatically removes it from memory. This process is known as garbage collection.

**String in Python**

**String:** String represents group of characters. Strings are enclosed in double quotes or single quotes. The str data type represents a String.

Ex: - “Hello”, ‘Sahil’

**Creating String**

**Single Quotes**

Ex: - a = ‘Python’

**Double Quotes**

Ex: - a = “Python”

**Triple Single Quotes**

Ex: - a = ‘’’Python’’’

**Triple Double Quotes**

Ex: - a = “””Python”””

**Double Quote inside Single Quotes**

Ex: - a = ‘Python “is a programming language”’

**Single Quote inside Double Quotes**

Ex: - a = “Python ‘is a programming’ language “

**Using Escape Characters**

Ex: - a = ‘Python \n is a programming language’

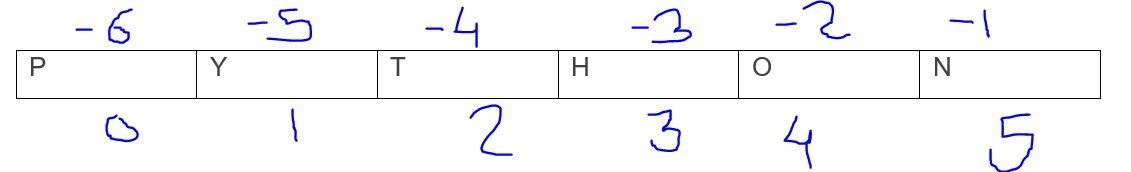
**Raw String -** is used to nullify the effect of escape characters

Ex: - a = r‘Python \n is a programming language’

**Index**

Index represents the position number of characters in a string.

**Example: -** name = “Python”



**Accessing Character and String**

name = “Python”

print(name[0])

print(name[1])

print(name[-4])

print(name[-6])

**Indentation**

The other programming languages such as C, C++, and Java use curly braces {}, whereas Python uses an indentation. Whitespaces are used as indentation in Python. Indentation refers to spaces that are used in the beginning of a statement. By default, python puts 4 spaces but it can be changed by programmers.

**Example:**

