**Variables in Python**

**Identifiers:** An identifier is a name given to the entities in a program such as variables, classes, functions and all the other objects in python.

* Identifiers are case sensitive.
* There is no length restriction for the identifiers but you have to make it reasonably short for better code readability.

**Examples:**

1. Value=42
2. classname=NewClass()

**Variable**: variable is a name that is used to store and refer to a memory location

Examples:

1. m=99
2. name=”akash”

**Note:** All variables are identifiers whereas, vice versa is not true.

**Some additional info about variables:**

* In many of the programming languages, variables are statically typed, i.e a variable is initially declared with a specific data type, and any value assigned to it.
* But variables in python are not restricted to this.
* In python a variable is initially assigned to one type and later can change its datatype.
* If you didn’t specify the data type of the variable to which you are taking the input then the python interpreter will consider it as the “string” datatype.
* If you do not assign the data type while assigning a value to the variable python interpreter will automatically assign the datatype.

Example:

**INPUT:**

### # type() is used to check the datatype of the variable.

### k=10.0

### print("The Datatype of :", k, "is ",type(k))

### m=10

### print("The Datatype of :", m, "is ",type(m))

### n="python"

### print("The Datatype of :", n, "is ",type(n))

**OUTPUT:**

### The Datatype of : 10.0 is <class 'float'>

### The Datatype of : 10 is <class 'int'>

### The Datatype of : python is <class 'str'>

In the example we didn't specify the datatype of the variable and also its datatype is assigned correctly by the python interpreter.

**Rules for naming the variables:**

1. A variable name must start with the underscore character or a letter

 Ex1. var (Accepted)

 Ex2. \_var(Accepted)

 Ex3. 1\_var(Not Accepted)

1. A variable name cannot start with a number or any special character like @,#etc.

 Ex1. #name (Not Accepted)

 Ex2. 95variable (Not Accepted)

 Ex3. py(Accepted)

1. A variable name can only contain alpha-numeric characters and underscores
* All letters (A to Z and a to z)
* Digits (0 to 9)
* underscore(\_)
1. Should not use the python keywords to name the variable

 Ex: if,elif,else,def etc.

**FOR MULTI\_WORD VARIABLE:**

The most commonly used methods of constructing a multi-word variable name are the last three examples:

**Camel Case:** Second and subsequent words are capitalized, to make word boundaries easier to see.

 Ex : classNameVariable

**Pascal Case:** Identical to Camel Case, except the first word is also capitalized.

 Ex: ClassNameVariable

**Snake Case:** Words are separated by underscores.

 Ex: class\_name\_variable

**OBJECT REFERENCES:**

* Python is an object oriented programming language.That means virtually every item of data in a Python program is an object of a specific class.
* When you create an object in python, a block of memory is allocated to store the object data.
* The variables are the names that are associated with these references.
* Python uses automatic garbage collection to reclaim the memory that is occupied by the objects that are no longer needed.

**Ex1:** single value (m=100)

In the above statement an integer object with the value 100 will be assigned to the variable m to point to that object.



 Variable Assignment

**Example2:** Multiple variables with same value ( m=n=100)



 Single object with multiple references

If we change any one of the value then the reference of one variable will be changed

**Example3:** different values (m=100 and n=99)



**OBJECT IDENTITY**

* In Python, whenever you create an object that is given a number that uniquely identifies that object.
* It is guaranteed that no two objects will have the same identifier except for the immutable objects like numbers,strings etc.
* When the number of references to an object drops to zero, it is no longer accessible. At that point, its lifetime is over.
* Python will eventually notice that it is inaccessible whenever the number of references to the object is zero and reclaim the allocated memory so it can be used for some other.
* To check that unique number we use the built-in Python function “ id()” that returns an object’s integer identifier.

Example1:

INPUT:

#### a=10

#### b=10

#### print("The ID of a is:",id(a)," and ""The ID of b is:",id(b))

#### m=75

#### n=85

#### print("The ID of m is:",id(m)," and ""The ID of n is:",id(n))

OUTPUT:

#### The ID of a is: 139701301793128 and The ID of b is: 139701301793128

#### The ID of m is: 139701301795208 and The ID of n is: 139701301795528

=============================================================

Example2:

INPUT:

#### a=[1,2,3,4,5,6,7,8,9,10]

#### b=[1,2,3,4,5,6,7,8,9,10]

#### print("The ID of a is:",id(a)," and ""The ID of b is:",id(b))

OUTPUT:

#### The ID of a is: 140334410624960 and The ID of b is: 140334407014016

==================================================================

**Note:**

* From the above two examples you can say that the object reference number is the same whenever the value is the same and the object is immutable.
* In case if it is a mutable object then the object reference number is different even if they hold the same content.

 **KEYWORDS IN PYTHON**

**Keywords:**

* Keywords are reserved words that have a predefined meaning and they are an integral part of the language syntax.
* These keywords will serve a specific purpose and are used to define the flow, structure and behavior of a program.
* The keyword cannot be used as an identifier, function, and variable name.
* There are 33 keywords in the python programming language.
* Python keywords are the fundamental building blocks of any Python program.

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