FirstBitSolutions.com

...Learn IT Bit by Bit...

Variable, Identifier, Operators

Objectives

Topic Covered:

- Python basic syntax, Input/output functions
- Python Variables, Comments
- Identifiers, Reserved words, Data types
- Python built-in data types
- Python operators

Python Basic Syntax

- Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

- Python Indentation : Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
- Python uses indentation to indicate a block of code.

```
if 5 > 2:
    print("Five is greater than two!")
```

Input/Output - print(),input() functions

- We use print statement to show output to user.

```
>>> print("Hello Everyone!!")
Hello Everyone!!
```

```
>>> x=5
>>> y=10
>>> z = x+y
>>> print("Sum =",z)
Sum = 15
```

Input/Output - print(),input() functions

- We use input function to accept input from user

```
>>> x = input("Enter a number")
Enter a number12
```

- Data accepted is in string format.
- So we need to convert it into our desired format.

```
>>> x = int(input("Enter a number"))
Enter a number5
```

Python Variables

- Variables are like place holders for values
- In Python, variables are created when you assign a value to it:

```
x = 5
y = "Hello, World!"
```

- Python has no command for declaring a variable.
- Multiple Assignment : Python allows you to assign a single value to several variables simultaneously.

```
a = b = c = 1
print(a,b,c)
a,b,c = 1,2,"john"
print(a,b,c)
```

Python Comments

- Comments : Python has commenting capability for the purpose of in-code documentation. Comments start with a #, and Python will render the rest of the line as a comment:

```
#This is a comment.
print("Hello, World!")
```

- Multiline comment :

```
'''Python has an easy to
to learn syntax'''
var1 = 1
var2 = 10
```

Python Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Python Identifiers

- Here are naming conventions for Python identifiers −
  - Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
  - Starting an identifier with a single leading underscore indicates that the identifier is private.
  - Starting an identifier with two leading underscores indicates a strongly private identifier.
  - If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Python Reserved Words

- The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Data Types

- Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instances of these classes.
- Numbers : Integers, floating point numbers and complex numbers are numbers in Python.

```
a = 10
print(type(a))
b = 3.4
print(type(b))
c = 1 + 2j
print(type(c))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Python built-in data types

- The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has following standard data types −

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Python Numbers

- Number data types store numeric values. Number objects are created when you assign a value to them. For example −

```
var1 = 1
var2 = 10
```

- Python supports four different numerical types −
  - int (signed integers)
  - long (long integers, they can also be represented in octal and hexadecimal)
  - float (floating point real values)
  - complex (complex numbers)

Python Numbers

- Here are some examples of numbers −

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEI	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

- A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x and y are the real numbers and j is the imaginary unit.

Python Operators

- Operators are used to perform operations on variables and values.
- Python divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Logical operators
  - Identity operators
  - Membership operators
  - Bitwise operators

Arithmetic Operators

- Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

- Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3

Comparison Operators

- Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

- Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity Operators

- Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership Operators

- Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Bitwise Operators

- Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Summary

- Variables can be created by assigning values
- A Python identifier is a name used to identify a variable
- Comments are used for documentation
- Reserved words or key words are used for special purpose
- Python supports built-in data types like numbers, list, tuples etc.


THANK YOU!