



Centre for Differently Abled Persons Bharathidasan University

III BCA – V SEMESTER

PYTHON PROGRAMMING **(20UCA5CC6)** **UNIT – II**

Prepared by
Dr. M. Prabavathy
Ms. M. Hemalatha

UNIT – II

Identifier

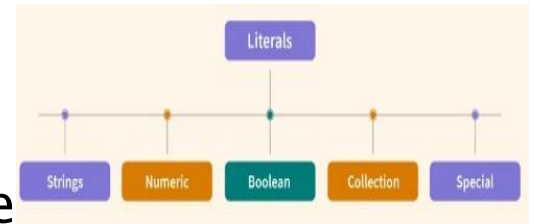
- Identifier is a name given to various programming elements such as a variable, function, class, module or any other object.

Rules are:

1. The allowed characters are a-z, A-Z, 0-9 and underscore (_)
 2. It should begin with an alphabet or underscore
 3. It should not be a keyword and can be of any size
 4. It is case sensitive and no blank spaces are allowed.
- Valid identifiers examples : si, rate_of_interest, student1, ageStudent
 - Invalid identifier examples : rate of interest, 1student, @age

Literals

- A literal is a constant that appear directly in a program and this value does not change during the program execution i.e. remain fixed.
- Example: `Num1 = 5`, `Principle_amount = 5000.00`
- We mainly have five types of literals which includes string literals, numeric literals, boolean literals, literal collections and a special literal `None`.



Keywords

- Keywords are the identifiers which have a specific meaning in python, there are 33 keywords inpython.

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

Comments

- Comments in Python is the inclusion of short descriptions along with the code to increase its readability.
- Comments in Python are identified with a hash symbol, #, and extend to the end of the line.
- Hash characters in a string are not considered comments, however.
- There are three ways to write a comment - as a separate line, beside the corresponding statement of code, or as a multi-line comment block.

Variables and Types

- When we create a program, we often need store values so that it can be used in a program

- We use variables to store data which can be manipulated by the computer program.
- Every variable has:
 - A. Name and memory location where it is stored.
 - A variable name:
 1. Can be of any size
 2. Have allowed characters, which are a-z, A-Z, 0-9 and underscore (_)
 3. Should begin with an alphabet or underscore
 4. Should not be a keyword
 - It is a good practice to follow these identifier naming conventions:
 1. Variable name should be meaningful and short
 2. Generally, they are written in lower case letters
 - B. A type or datatype which specifies the nature of variable. We can check the type of the variable by using type command

```
>>>type(variable_name)
```
 - C. A value

Datatypes

Various datatypes available in python are as follow:

- **Number**

- Number data type stores Numerical Values.
- These are of three different types:

1. Integer & Long

- Integers are the whole numbers consisting of + or – sign like 100000, -99, 0, 17.

2. Float/floating point

- Numbers with fractions or decimal point are called floating point numbers.
- A floating point number will consist of sign (+,-) and a decimal sign(.)
- Example: temperature= -21.9,growth_rate= 0.98333328
- The floating point numbers can be represented in scientific notation such as
- -2.0X 10⁵ will be represented as -2.0e5
- 2.0X10⁻⁵ will be 2.0E-5

Complex

- Complex number is made up of two floating point values, one each for real and imaginary part. For accessing different parts of a variable `x` we will use `x.real` and `x.imag`.
- Imaginary part of the number is represented by `j` instead of `i`, so `1+0j` denotes zero imaginary part.
- Example:

```
>>> x = 1+0j, >>> print x.real,x.imag
```

 - ▶ Output: 1.0 0.0

- **None**

- This is special data type with single value. It is used to signify the absence of value/false in a situation. It is represented by None.

- **Sequence**

- A sequence is an ordered collection of items, indexed by positive integers. Three types of sequence data type available in Python are Strings, Lists & Tuples.

- 1.String:

- String is an ordered sequence of letters/characters. They are enclosed in single quotes (‘’) or double (‘’”).
 - Example: >>> a = 'Ram', >>>a=’’Ram’’

- 2.Lists:

- List is also a sequence of values of any type.
 - Values in the list are called elements /items.
 - The items of list are accessed with the help of index (index start from 0).

- List is enclosed in square brackets.
- Example: Student = ["Ajay", 567, "CS"]

3. Tuples:

- Tuples are a sequence of values of any type, and are indexed by integers.
- They are immutable i.e. we cannot change the value of items of tuples.
- Tuples are enclosed in ().
- Example: Student = ("Ajay", 567, "CS")

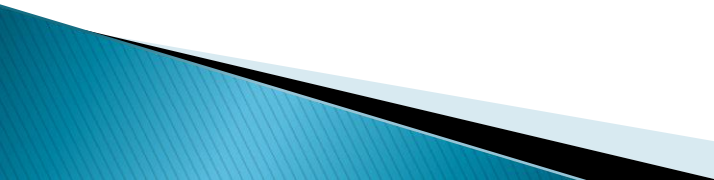
4. Sets:

- Set is an unordered collection of values, of any type, with no duplicate entry.
- Sets are immutable.
- Example: s = set ([1,2,3,4])

5. Dictionaries:

- Dictionaries store a key – value pairs, which are accessed using key.
- Dictionary is enclosed in curly brackets.
- Example: d = { 1:'a', 2:'b', 3:'c' }

Data type Conversion

- Type conversion is the process of converting a data type into another data type.
 - There are mainly two types of type conversion methods in Python.
 - Implicit type conversion is performed by a Python interpreter only.
 - Explicit type conversion is performed by the user by explicitly using conversion functions in the program code.
- 

- Explicit type conversion is also known as typecasting.

- Example:

```
a = 5
```

```
b = 5.5
```

```
sum = a + b print (sum)
```

```
print (type (sum))
```

- Output: 10.5

```
<class 'float'>
```

Operators

- Operators are special symbols which represents specific operations.
- They are applied on operand(s), which can be values or variables.
- Operators when applied on operands form an expression.
- **Arithmetic Operators :**
 - Arithmetic operators are used to apply arithmetic operation.

Symbol	Description	Example 1	Example 2
+	Addition	>>>55+45 100	>>> 'Good' + 'Morning' GoodMorning
-	Subtraction	>>>55-45 10	>>>30-80 -50
*	Multiplication	>>>55*45 2475	>>> 'Good'* 3 GoodGoodGood
/	Division	>>>17/5 3 >>>17/5.0 3.4 >>> 17.0/5 3.4	>>>28/3 9
%	Remainder/ Modulo	>>>17%5 2	>>> 23%2 1

- **Relational Operators :**

- Relational operators are used to compare two items & the result is True or False.

Symbol	Description	Example 1	Example 2
<	Less than	<pre>>>>7<10 True >>> 7<5 False >>> 7<10<15 True >>>7<10 and 10<15 True</pre>	<pre>>>>'Hello'< 'Goodbye' False >>>'Goodbye'< 'Hello' True</pre>
>	Greater than	<pre>>>>7>5 True >>>10<10 False</pre>	<pre>>>>'Hello'> 'Goodbye' True >>>'Goodbye'> 'Hello' False</pre>
<=	less than equal to	<pre>>>> 2<=5 True >>> 7<=4 False</pre>	<pre>>>>'Hello'<= 'Goodbye' False >>>'Goodbye' <= 'Hello' True</pre>

<pre>>=</pre>	<pre>greater than equal to</pre>	<pre>>>>10>=10 True >>>10>=12 False</pre>	<pre>>>>'Hello'>= 'Goodbye' True >>>'Goodbye'>= 'Hello' False</pre>
<pre>!=, <></pre>	<pre>not equal to</pre>	<pre>>>>10!=11 True >>>10!=10 False</pre>	<pre>>>>'Hello'!= 'HELLO' True >>>'Hello'!= 'Hello' False</pre>
<pre>==</pre>	<pre>equal to</pre>	<pre>>>>10==10 True >>>10==11 False</pre>	<pre>>>>'Hello' == 'Hello' True >>>'Hello' == 'Good Bye' False</pre>

- **Logical Operators**

- Logical Operators give the logical relationship based upon the truth table.

Symbol	Description
or	If any one of the operand is true, then the condition becomes true.
and	If both the operands are true, then the condition becomes true.
not	Reverses the state of operand/condition.

- **Membership operator**

- It is test for membership in a sequence, such as strings, lists and tuples.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

- **Identity Operator**

- It compares the memory locations of two objects.

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

Operator Precedence

- It is in descending order and the operator precedence in Python is listed below:

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

THANK YOU

