Python Fundamental

SiSoft

Token

- Keywords
- Literals
  - String literals
  - Numeric literals
  - Boolean Literals
  - Special literal ***None***
- Data Types
- Operators
- Identifiers
- Punctuators

SiSoft

Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 33 keywords in Python 3.3. This number can vary slightly in course of time.
- All the keywords except True, False and None are in lowercase and they must be written as it is. The list of all the keywords are given below

Sisoft

Keywords

Keywords in Python programming language

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

Sisoft

Data Types

- **Booleans**
  - are either `True` or `False`.
- **Numbers**
  - can be integers (`1` and `2`), floats (`1.1` and `1.2`), fractions (`1/2` and `2/3`), or even complex numbers.
- **Strings**
  - are sequences of Unicode characters, *e.g.* an html document.
- **Bytes** and **byte arrays**, *e.g.* a jpeg image file.
- **Lists** are ordered sequences of values
  - a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
- **Tuples** are ordered, immutable sequences of values.
  - a_tuple = ("a", "b", "mpilgrim", "z", "example")
- **Sets** are unordered bags of unique values
  - a_set = {1, 3, 6, 10, 15, 21, 28, 36, 45}
- **Dictionaries** are unordered bags of key-value pairs
  - a_dict = {'server': 'db.diveintopython3.org', 'database': 'mysql'}

SiSoft

Number Data Types

- **int (signed integers)**: They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
- **long (long integers)**: Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
- **float (floating point real values)** : Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = $2.5 \times 10^2 = 250$).
- **complex (complex numbers)** : are of the form a + bJ, where a and b are floats and J (or j) represents the square root of -1 (which is an imaginary number). The real part of the number is a, and the imaginary part is b. Complex numbers are not used much in Python programming.

String Data Type

- String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String
- Single line
- Multiple line (Using back slash or triple quotation)

SiSoft

Immutable and mutable data types

Immutable data type

- The Immutable types are those that can never change their values.
- NOTE: Although it may appear that the value of variable do change, but they don't change in the same place. A new variable space is assigned **at a new memory space**.
- Immutable types are:
  - Integers
  - Floating point numbers
  - Booleans
  - Strings
  - Tuples

Mutable data type

- The mutable type are those whose value **change in place**.
- Mutable types are:
  - Lists
  - Dictionaries

```
>>> list = [10,20,30]
>>> list[1] = 40
>>> list
>>> [10,40,30]
```

SiSoft

The type() method

The method **type()** returns the type of the passed variable. If passed variable is dictionary then it would return a dictionary type.

Syntax:

```
type(x) ##Where x is a already defined variable
```

To print on the output window:

```
print type(x)
```

Example:

```
>>> name = “Eddie”
>>> age = 34
>>> print “name is”,type(name),”age is”,type(age)
name is <type 'str'> age is <type 'int'>
```

Variables - Named labels whose values can be manipulated during program run.

Creating Variables-

```
>>> name = “Eddie”
>>> age = 34
```

Dynamic Typing

- A variable pointing to a value of certain type can be made to point to a value/object of different type. This is called *Dynamic Typing*.

```
>>> x = 10
>>> type(x)
<type 'int'>
>>> x = “summer”
<type 'str'>
```

Caution: Although Python is comfortable with changing types of a variable, the programmer is responsible for ensuring right types for certain type of operations.

Type Casting in Python

- To convert into integer
    - int(_)
- To convert into string
    - str(_)
- To convert into float
    - float(_)
- To convert into Boolean
    - bool(_)
    - Only 0 give false rest any number give true.
- To convert asci code into string
    - chr(_)
    - Not exactly type casting

SiSoft

Operators

- The operations being carried out on data are represented by operators. The symbols that trigger the operation / action on data are called **operators**. The operation (specific task) are represented by operators and the objects of the operations are referred to as *Operands.*

Type of operators in Python
Arithmetic operators
Comparison (Relational) operators
Logical (Boolean) operators
Bitwise operators
Assignment operators
Special operators

Arithmetic Operators

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Comparison Operator

Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	

Logical Operator

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

SiSoft

Bitwise Operator

Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000

Assignment Operator

Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Identity operators

- is and is not are the identity operators in Python. They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Membership operators

- in and not in are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).
- In a dictionary we can only test for presence of key, not the value.
- We can’t check an integer type in a string or in an character array.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

SiSoft

Identifiers

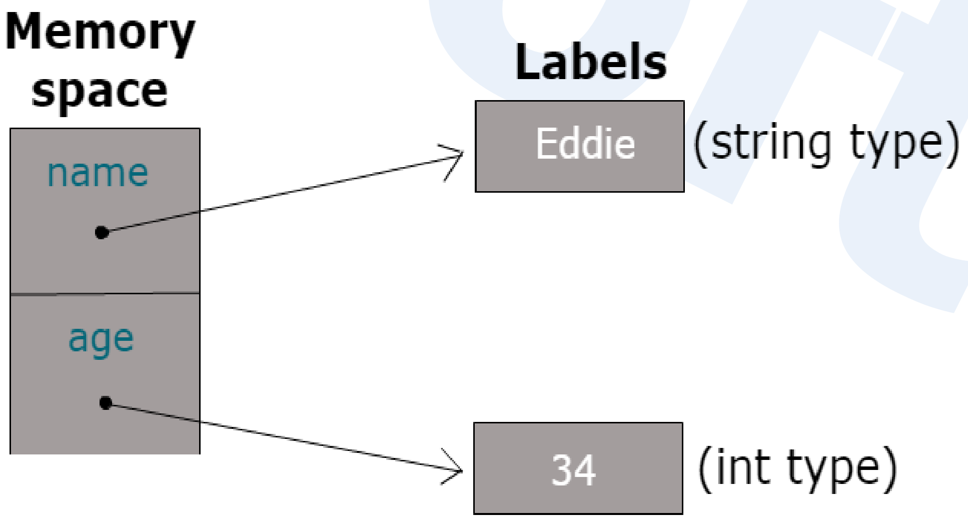
- Convention for identifier is:
  - An identifier is an arbitrary long sequence of letter and digits.
  - The first character must be a letter; the underscore (_) counts as a letter.
  - Upper and lower-case letters are different. All characters are significant.
  - Digits can be part of identifier but can't be as the first character.
  - An identifier must not be a keyword of python.
  - An identifier can't contain any special character.

SiSoft

Creating Variables

- **Variables** are Named labels whose values can be manipulated during program run.

```
>>> name = "Eddie"
>>> age = 34
```



SiSoft

Python Style Rules and Convention

- ❖ **Statement Termination**

  Python does not use to terminate a statement. When you end a physical code-line by pressing Enter key, the statement is considered terminated by default

- ❖ **Avoid multiple statement in one line**

  Multiple statements can be written in one line using semicolon (;) but should be avoided.

- ❖ **Maximum line length**

  Line length should be maximum 79 characters.

- ❖ **Line and Indentation**

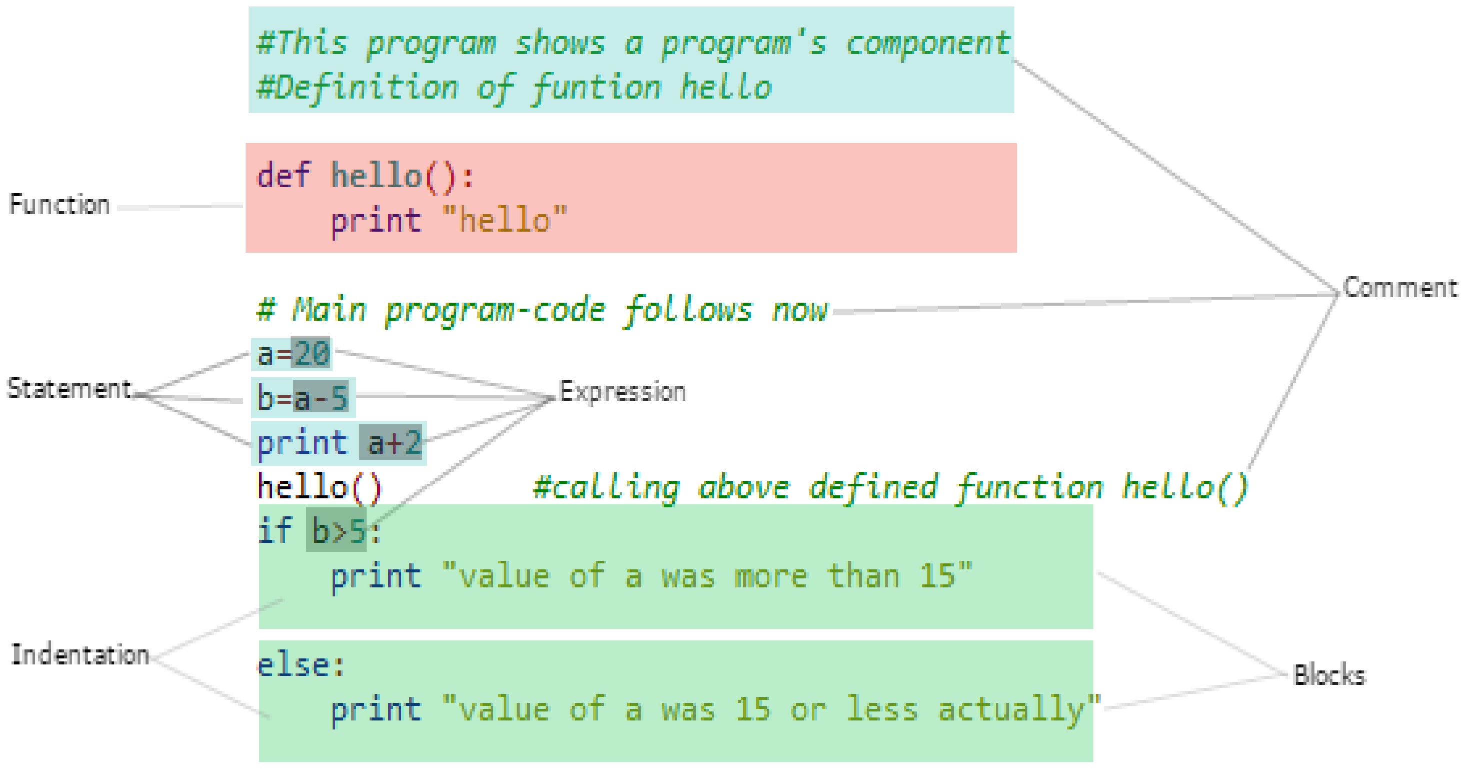
  Blocks of code are denoted by line indentation, which is enforced through 4 spaces (press Tab once) per indentation level.

- ❖ **Case Sensitivity**

  Python is case sensitive, so one has to be careful with keyword and identifier. Eg Print ”Hello” – will give error while, print ”hello” -- will print hello.

SiSoft

Barebones of a Python Program



```
#This program shows a program's component
#Definition of funtion hello

def hello():
    print "hello"

# Main program-code follows now
a=20
b=a-5
print a+2
hello()        #calling above defined function hello()
if b>5:
    print "value of a was more than 15"

else:
    print "value of a was 15 or less actually"
```

Multiple Assignment

1. Assigning same value to multiple variable.

```
>>> a=b=c=10
```

2. Assigning multiple values to multiple variable.

```
>>> a, b, c = 10,20,30
```

A Program For Swapping value variable

```
>>> a, b= 10,20
>>> a,b=b,a
>>> a, b
(20,10)
```

Comments

- In Python, we use the hash (#) symbol to start writing a comment.
- It extends up to the newline character.

- Triple quotes(‘’’ or “””) are generally used for multi-line strings. But they can be used as multi-line comment as well.

Sisoft

Sisoft

Object

- In python every entity that stores any value or any type of data is called Object.
- A object has three key attributes:
  - **The *type* of an object**: The type of an object defines the operation can be performed on an object .
  - **The *value* of an object**: It is the data -item contained in object.
  - **The *id* of an object**: The memory location of an object.

SiSoft