Python Programming-An Introduction

Lecture 1

Department of Computer Science and Engineering, ITER
Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, India.

Contents

Introduction

- Python is a general purpose, high level and interpreted programming language.
- Python is an interactive programming language.
- Simple syntax, Easy to read and write
- Python was developed by **Guido Van Rossum** in 1991 at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Introduction (Cont.)

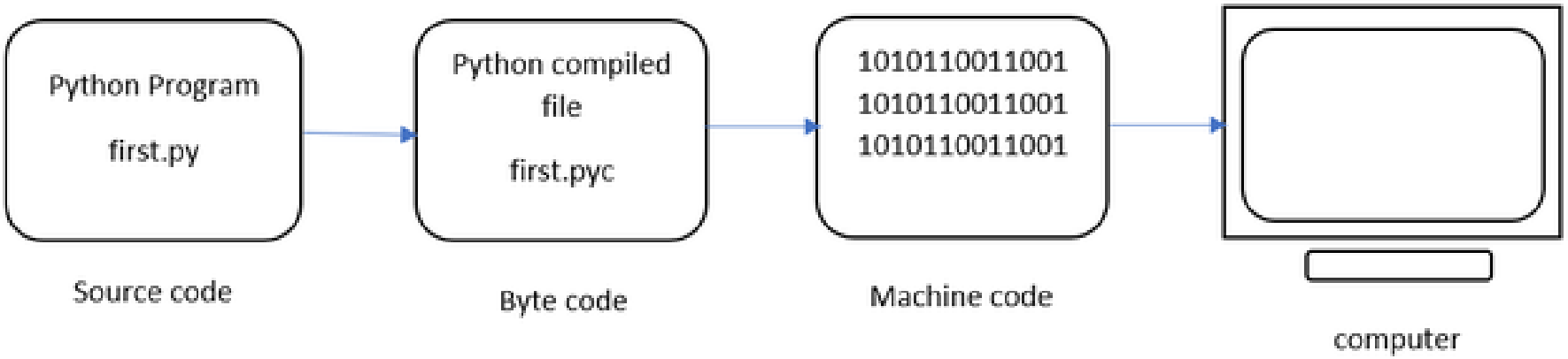


Figure 1: Steps of execution of a python program

Execution of a Python Program

- Normally, when we compile a python program, we cannot see .pyc file produced by python compiler and the machine code generated by Python Virtual Machine (PVM). This is done internally in the memory and the output is finally visible. For example, if our python program name is first.py, we can use python compiler to compile it as:
  python first.py
- In the preceding statement, *python* is the command for calling the python compiler. The compiler should convert the first.py file into its byte code equvalent file first.pyc. Instead of doing this, the compiler directly displays the output or result. To separately create .pyc file from the source code, we can use the following command:
  python -m py_compile first.py
- In order to interpret the .pyc file using PVM, the python compiler can be called using the following command:
  python first.cpython-38.pyc

IDLE - An Interpreter for Python

- IDLE stands for Integrated Development and Learning Environment.
- Python IDLE comprises **Python Shell** (An Interactive Interpreter) and **Python Editor** (Allows us to work in Script Mode).
- While using Python shell, we just need to type Python code at the ⟩⟩⟩ prompt and press Enter key. For Example:
  ⟩⟩⟩ print('Hello World')
  Hello World
- Python shell may also be used as a calculator. For Example:
  ⟩⟩⟩ 18+5
  23
  ⟩⟩⟩ 27//5
  5
  ⟩⟩⟩ 27.0//5
  5.0
  ⟩⟩⟩ 27%5
  2

IDLE - An Interpreter for Python

- We evaluate the foregoing expressions in Python shell. For Example:

```
>>> 3**2
9
>>> 6/3/2          #(6/3)/2
1.0
>>> 2**3**2        #2**(3**2)
512
```

- Left associative operators: +, -, *, /, //, %
- Right associative operators: **
- Precedence of Arithmetic operators are:

() (parentheses)	decreasing order
** (exponentiation)	
- (negation)	
/ (division) // (integer division) * (multiplication) % (modulus)	
+ (addition) - (subtraction)	

IDLE - An Interpreter for Python

- While the parentheses have the highest precedence, addition and subtraction are at the lowest level.
- Python complains when it encounters a wrongly formed expression. For Example
  ⟩⟩⟩ 7+3(4+5)
  Traceback (most recent call last):
  File "⟨*pyshell*#17⟩", line 1, in ⟨*module*⟩
  7+3(4+5)
  TyeError: 'int' object is not callable
- Similarly, Division by zero is a meaningless operation. For Example
  ⟩⟩⟩ 7/0
  Traceback (most recent call last):
  File "⟨*pyshell*#18⟩", line 1, in ⟨*module*⟩
  7/0
  ZeroDivisionError: division by zero

Python Strings

- A string is a sequence of characters.
- We can enclose a a sequence of characters between single, double, or triple quotes to specify.
- A string enclosed in single quotes may include double quotes marks and vice versa.
- A string enclosed in triple quotes (also known as docstring, i.e. documentation string) may include both single and double quote marks and may extend over several lines. For Example
  ⟩⟩⟩ 'Hello World'
  'Hello World'
  ⟩⟩⟩ print('Hello World')
  Hello World
  ⟩⟩⟩ """Hello
  World"""
  "Hello
  World"

Python Strings

- Escape sequence \ n marks a new line.
- Use of + as the concatenation operator.
- The operator * (multiplication) is used to repeat a string a specified number of times.
- For Example:
  ⟩⟩⟩ 'hello' + '!!'
  'hello !!'
  ⟩⟩⟩ 'how' + 'are' + 'you?'
  'how are you?'
  ⟩⟩⟩ 'hello' * 5
  'hellohellohellohellohello'

Relational Operators

- Relational Operators are used for comparing two expressions and yield True or False.
- Arithmetic operators have higher precedence than the relational operators.

==	(equal to)
<	(less than)
>	(greater than)
< =	(less than or equal to)
> =	(greater than or equal to)
! =	(not equal to)

- The relational operators are:
- A relational operator applied on expressions as:
  expression ⟨*comparisonoperator*⟩ expression
- For Example:
  ⟩⟩⟩ 23 != 23
  False
- ASCII vales of characters are used for string comparison.
- Python 3 does not allow string values to be compared with numeric values.

Logical Operators

- Logical operators not, and, and or are applied to logical operands True and False, also called Boolean values, and yield either rue or False.
- The operator not is unary, whereas other two are binary operator, those are as described below:

not	
True	False
False	True

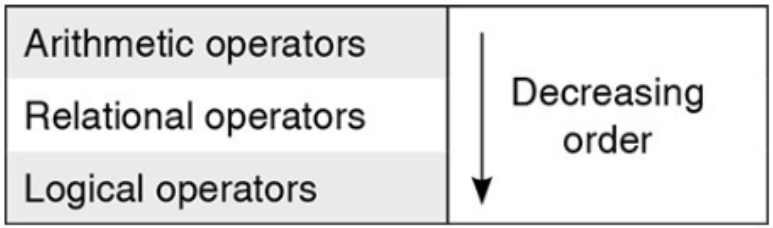
and	True	False
True	True	False
False	False	False

or	True	False
True	True	True
False	True	False

Logical Operators

- The precedence of logical operators are:

- The precedence of operators are:



- For Example: The expression (10 $\langle$ 5) and ((5/0) $\langle$ 10) yields False, the expression (10 $\rangle$ 5) and ((5/0) $\langle$ 10) yields an error.

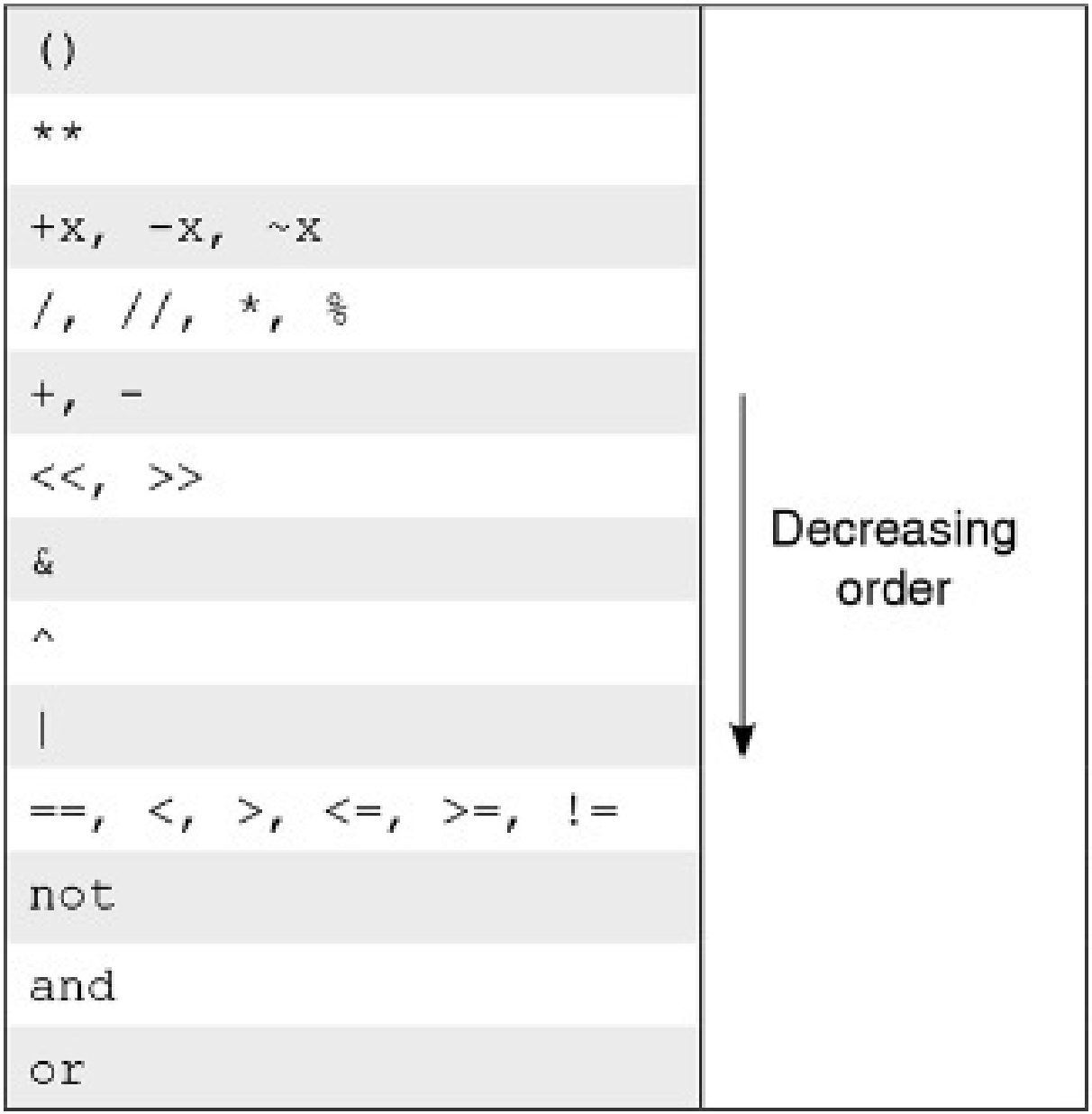
Bitwise Operators

- These operators operate on integers interpreted as strings of binary digits 0 and 1, also called bits, as:

Bitwise operator	Description	Example
Bitwise AND `x & y`	A bit in `x & y` is 1 if the corresponding bit in each of `x` and `y` is 1, and 0 otherwise.	`10 & 6` yields 2: 10: 00001010 6: 00000110 --- 2: 00000010
Bitwise OR `x | y`	A bit in `x | y` is 1 if at least one of the corresponding bits in `x` and `y` is 1, and 0 otherwise.	`10 | 6` yields 14: 10: 00001010 6: 00000110 --- 14: 00001110
Bitwise Complement `~ x`	A bit in `~x` is 1 if and only if the corresponding bit in `x` is 0. The result obtained from this operation is `-x-1`.	`~ 11` yields -12: 11: 00001011 --- -11: 11110100 Note that 11110100 is the two's complement of 12: 00001100.
Bitwise Exclusive OR `x ^ y`	A bit in `x ^ y` is 1 if exactly one of the corresponding bits in `x` and `y` is 1, and 0 otherwise.	`10 ^ 6` yields 12: 10: 00001010 6: 00000110 --- 12: 00001100
Left Shift `x << y`	Bits in the binary representation of `x` are shifted left by `y` places. Rightmost `y` bits of the result are filled with zeros. The result obtained from this operation is `x*2^y`.	`5 << 2` yields 20: 5: 00000101 --- 20: 00010100
Right Shift `x >> y`	Bits in the binary representation of `x` are shifted right by `y` places. Leftmost `y` bits of the result are filled with sign bit. The result obtained from this operation is `x/2^y`.	`5 >> 2` will yield 1 as the result as shown below: 5: 00000101 --- 1: 00000001

Bitwise Operators

- The precedence of operators are as:



Variables and Assignment Statements

- Variables provide a means to name values so that they can be used and manipulated later on. For Example:
  ⟩⟩⟩ english = 57
  When this statement is executed, Python associates the variable english with value 57.
- In Python style (often called Pythonic style or Pythonic way), variables are called names, and an assignment is called an association or a building.
  ⟩⟩⟩ english
  57
  ⟩⟩⟩ maths = 64
  ⟩⟩⟩ total = english + maths
  121

Variables and Assignment Statements (Contd..)

- Syntax for assignment statement:
  variable = expression
- Rules for naming variables:
  1. A variable must begin with a letter or _ (Underscore character).
  2. A variable may contain any number of letters, digits, or underscore characters. No other character apart from these is allowed.
- Always use meaningful variables names.
- Follow a consistent style in choosing variables.
- Examples of not valid variables as:

total_no.	#use of dot (.)
1st_number	#begins with a digit
AmountIn$	#use of dollar symbol ($)
Total Amount	#Presence of blank between two words

Variables and Assignment Statements (Contd..)

- Python is case-sensitive. For example, age and Age are not same, different.
- More than one variable may refer to the same object.
  ⟩⟩⟩ a = 5
  ⟩⟩⟩ b = a
  ⟩⟩⟩ b
  ⟩⟩⟩ a = 7
  ⟩⟩⟩ a
- The shorthand notation works for all binary mathematical operators. a = a ⟨*operator*⟩ b is equivalent to a ⟨*operator*⟩ = b. For example: a = a + b is equivalent to a + = b.
- Multiple assignments in a single statement as:
  ⟩⟩⟩ a, b, c = 'how', 'are', 'you?'
- Assigning same value to multiple variables in a single statement as:
  ⟩⟩⟩ a = b = 0

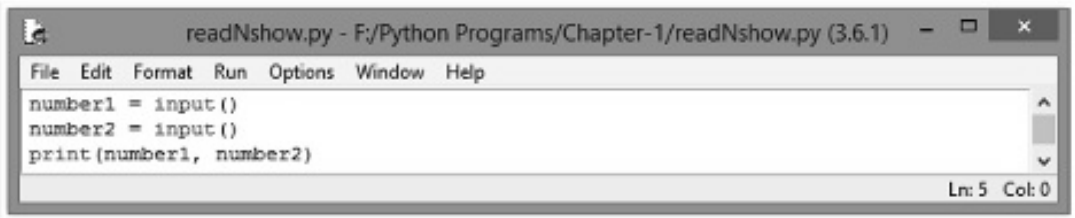
Keywords

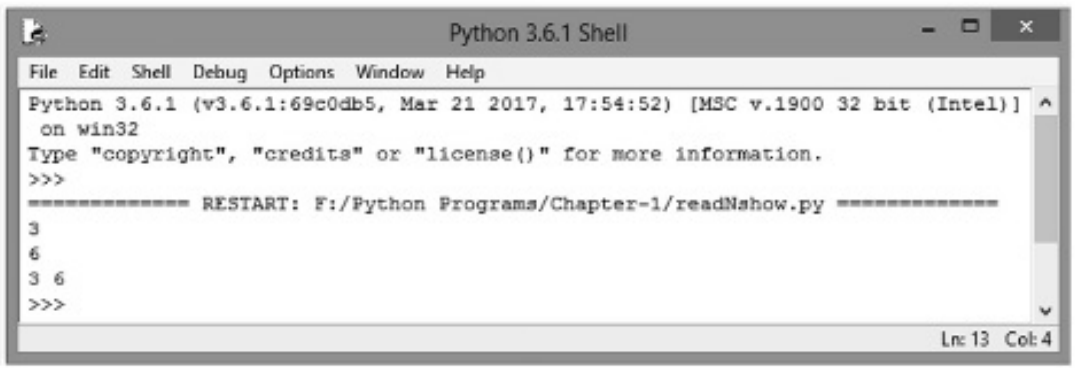
- Keywords are the reserved words that are already defined by the Python for specific uses.
- Keywords cannot be used for any other purposes.
- Keywords cannot be used for naming objects.
- List of Python keywords are as:

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

Script Mode

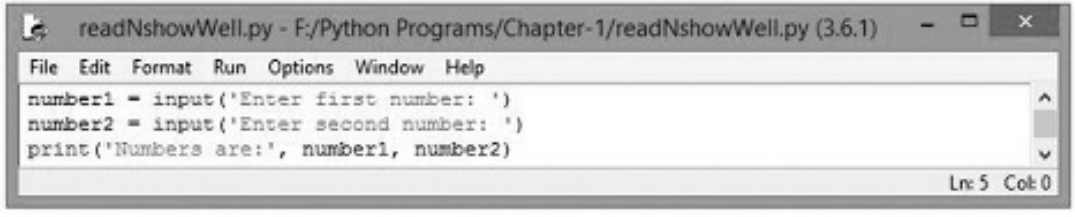
- When we exit an IDLE session, and start another IDLE session, we must redo all computations.This is not convenient mode of operation for most of the computational tasks.
- Python provides another way of working called script mode.
- In script mode, instructions are written in a file.
- A script should have extension .py or .pyw.
- For Example:

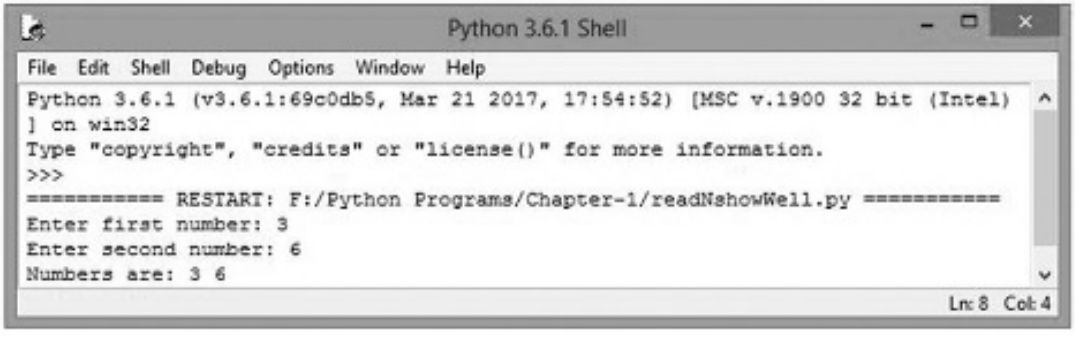




Script Mode (Contd..)

- We can modify the previous example as we may not remember that the program requires two numbers as the input.
- A good programming practice is to display to the user what data to be entered.
- For Example:





Summary

- Python interpreter executes one expression (command) at a time.
- A string is a sequence of characters. To specify a string, we may use single, double, or triple quotes.
- While evaluating a boolean expression involving *and* operator, the second sub-expression is evaluated only if the first sub-expression yields True.
- While evaluating an expression involving *or* operator, the second sub-expression is evaluated only if the first sub-expression yields False.
- A variable is a name that refers to a value. We may also say that a variable associates a name with data object such as number, character, string or Boolean.
- A variable name must begin with a letter or _.
- Python is case-sensitive.
- Python programming can be done in an interactive and script mode.

References

[1] Python Programming: A modular approach by Taneja Sheetal, and Kumar Naveen, *Pearson Education India, Inc.*, 2017.

Thank You
Any Questions?