Introduction and Programming Terminologies

Lecture 1

CPT023 – Fundamentals of Programming

- Non-contact study hours:
  - At least 8 hours / week

- Attendance is compulsory
  - All lectures and tutorials
  - Attendance < 80% → not eligible to sit final exams

Assessment Scheme

- The assessment for the subject will be as follows:

- 20% - Completion of all the labs
- 20% - Mid-term (week 8)
- 60% - Final Exam

- To pass, students must:
  - Submit all assessments
  - Obtain a minimum composite mark of 50%

Lecture Schedule

Lectures	Week	Topic
1	1	Introduction to the subject. Some programming terminologies
2	2	Variable names and keywords. Operators and operands.
3	3	Conditional statements
4	4	Nesting Conditionals
5	5	Repetition
6	6	Nesting repetitions
7	7	Preparation for the Midterm exam.
8	8	Methods / functions / library functions / procedures
9	9	Methods and argument passing to functions
10	10	Arrays / Lists
11	11	Tuples and Dictionaries
12	12	Testing Methods
13	13	Developing longer programs
14	14	Revision

Tutorial Schedule

Tutorials	Week	Topic
1	1	No labs during this week
2	2	Running Python. IDLE. Running simple programs. Writing programs with variables.
3	3	Write programs with conditional statements.
4	4	Nested conditionals
5	5	Repetitions
6	6	Develop programs with more than one level of repetitions
7	7	Midterm test
8	8	Introduce functions, call functions, passing and returning controls from functions and procedures.
9	9	Functions and variable passing
10	10	One dimensional arrays / lists
11	11	Tuples and Dictionaries
12	12	Writing own test cases
13	13	Debugging techniques
14	14	Revision

Lecture 1 - Objectives

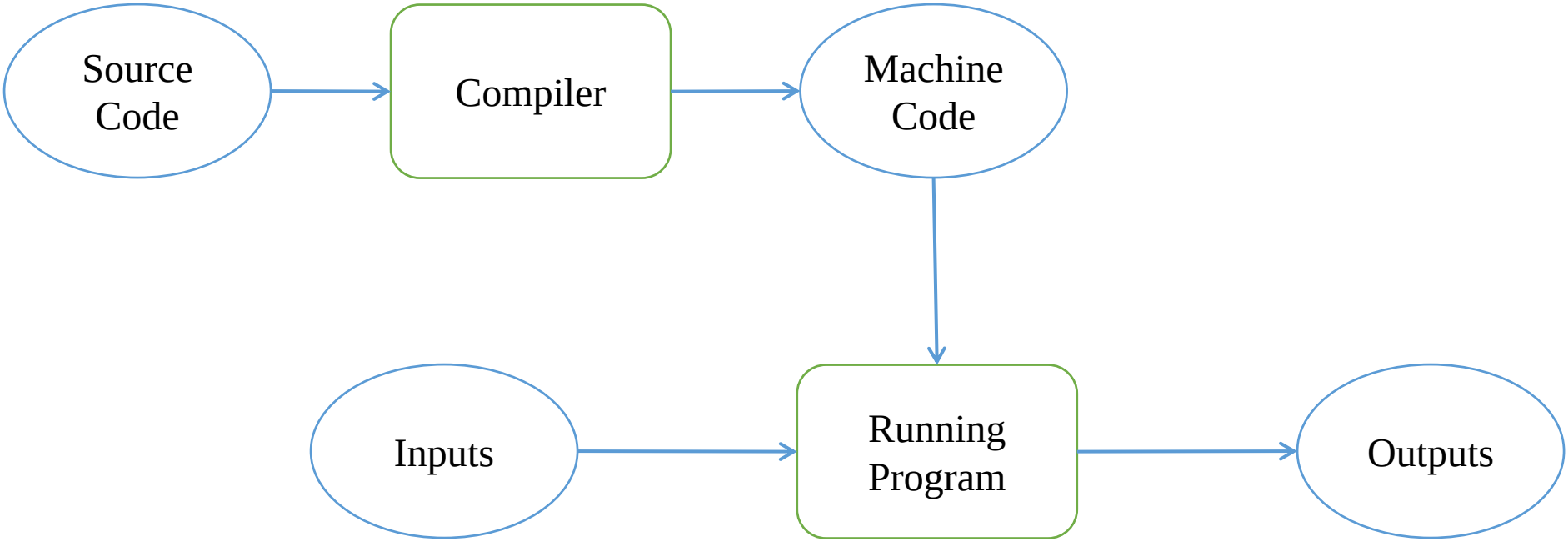
- After completing the lesson, the student will be able to:
  - State what topics the subject covers
  - State some concepts of programming
  - Download and install Python
  - Differentiate between compiler and interpreter
  - Have some idea of the process of debugging
  - Differentiate between syntax errors and run time errors
  - List different types of programming languages.
  - Write, compile and display the output of some programming statements using python IDLE environment.

What is programming?

- Code
- Coding
- Source code
- Object code
- Program
- Programming language
- Compiling
- Compiler
- Interpreter

Compiler

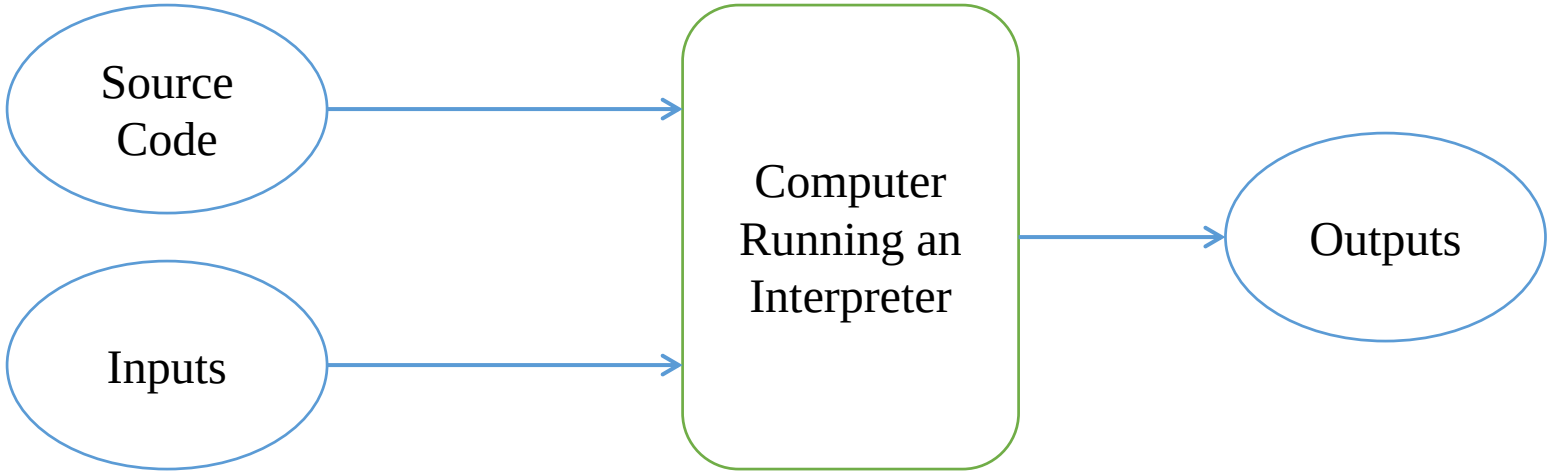
- Compiler is used to translate high level language into machine language.



Compiling a High Level Language

Interpreter

- Interpreter is also used to translate high level language into machine language.



Interpreting a High Level Language

Compiler and Interpreter

Compiler	Interpreter
Compiling is a one-shot translation	Interpreter analyzes and executes the source code instruction by instruction as necessary.
Once a program is compiled, it may be run over and over again	The interpreter and the source are needed every time the program runs.
Compiled programs tend to be faster	Interpreted languages lend themselves to a more flexible programming environment as programs can be developed and run interactively

Debugging

- Errors that occur in programs are called bugs
- The process of locating and fixing errors is called debugging a program.
- Following is the typical debugging process:
  - Describe the bug
  - Get the program snapshot when the bug 'appears'.
  - Analyze the snapshot and search for the cause of the bug.
  - Fix the bug.

Syntax Errors

- Syntax error refers to an error in the syntax of the programming statement.
- For compiled languages syntax errors occur strictly at compile-time.
- For interpreted languages, all syntax errors cannot be detected until run time, so it is not necessary to differentiate a syntax error from a logical error.
- A syntax error may also occur when an invalid equation is entered into a calculation.
- Example: opening brackets without closing them.

Run Time Errors

- Run time is the time during which a program is running.
- An error that occurs during the execution of a program.
- Example: running out of memory will often cause a runtime error.

Run Time Errors

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32
bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informat
ion.
>>> a= 5
>>> b = 0
>>> c= a/b

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    c= a/b
ZeroDivisionError: integer division or modulo by zero
>>>
```

Ln: 8 Col: 41

Programming Languages

- Machine Language refers to the “1s and 0s" that digital processors use as instructions.
- Example:
  - One pattern of bits (such as 11001001) add two numbers
  - A different pattern (such as 11001010) subtracts one from the other.
- Machine Language is very difficult to work with, and almost never worth the effort anymore.

Programming Languages

- An assembly language is a low-level programming language for computers, microprocessors, microcontrollers, and other programmable devices.
- It implements a symbolic representation of the machine codes and other constants needed to program a given CPU architecture.
- An assembler is used to translate assembly language statements into machine code.

Programming Languages

- High-level languages are relatively sophisticated sets of statements utilizing words and syntax from human language.
- They are more similar to normal human languages than assembly or machine languages.
- Larger and more complicated programs can be developed faster by using these languages.

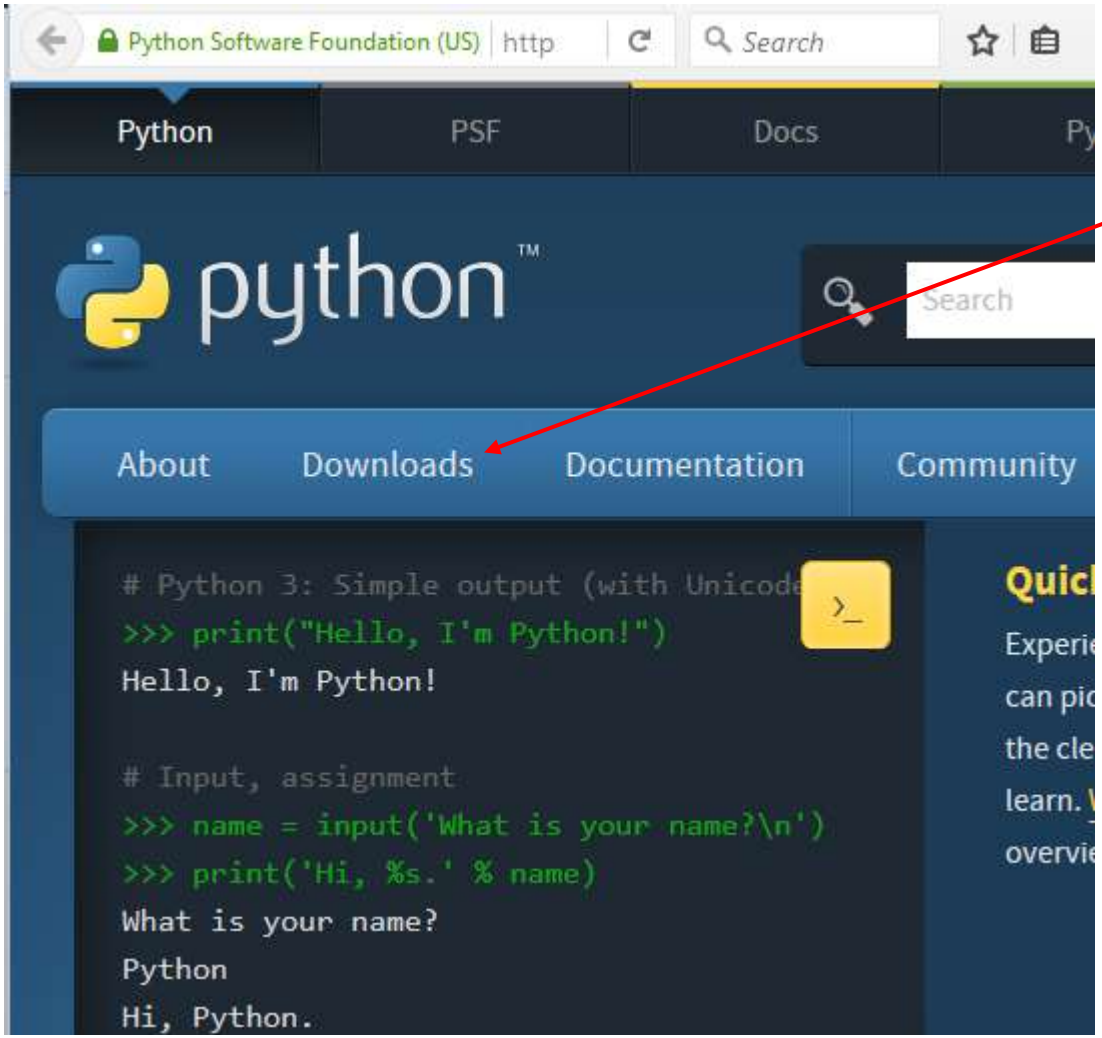
Programming Languages

- It must be translated into machine language by using a compiler.
- It may take longer to execute and use up more memory than programs written in an assembly language.
- Example: Visual Basic, C/C++, Pascal, BASIC etc..

Compiling Python Code

- Python source code is automatically compiled into Python byte code by the CPython interpreter.
- Compiled code is usually stored in PYC files, and is regenerated when the source is updated.
- To distribute a program to people who already have Python installed, you can send either the PY files or the PYC files.

Download and Install Python



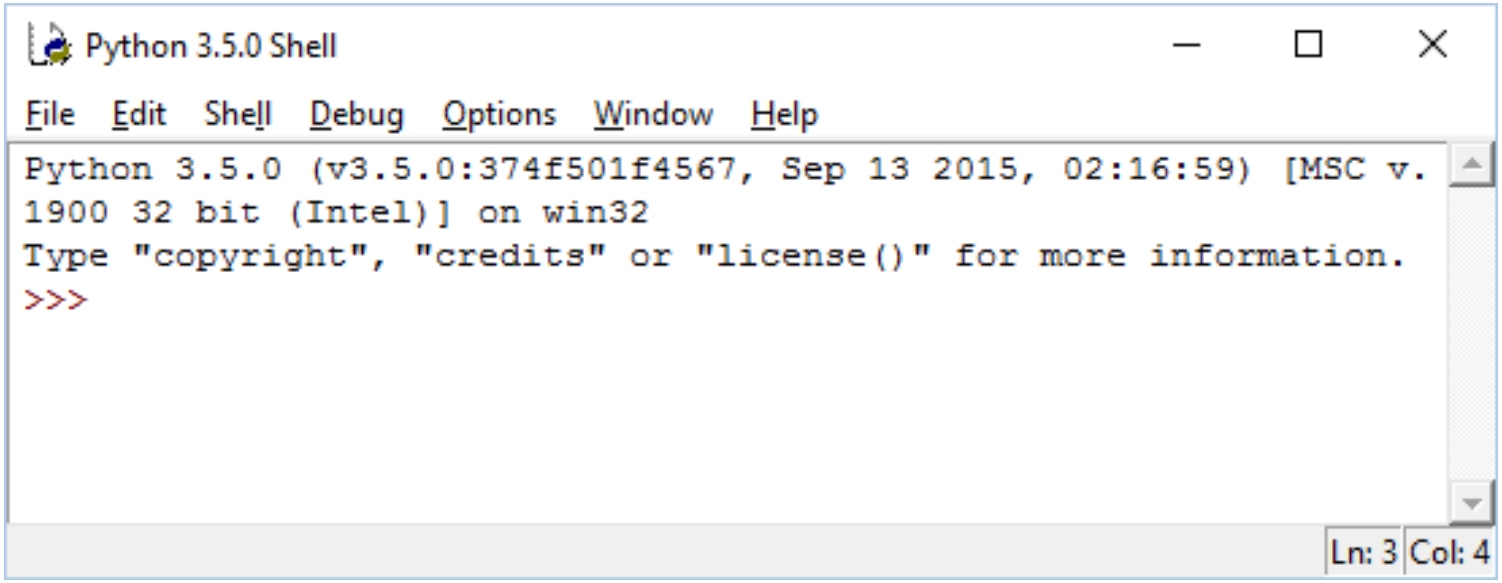
Click Download

Download and Install Python

- Click the Save File button to save Python application in your system
- Locate the file and then double click to start to install the Python application
- Click the Run button

Start Python Application

- Click Start > All Programs > Python 3.5
  - Or alternatively, press Windows key and start typing “IDLE”
- Select IDLE (Python GUI) from the list
- The following window appears:



Write Programming Statements and Display Output

- The >>> is a Python prompt
- To display a message simply type the message within double quotes and press enter key.
- Or use the print keyword before the message in double quotes.
- Examples:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Welcome to Fundamentals of Programming")
Welcome to Fundamentals of Programming
>>> "Welcome to Fundamentals of Programming"
'Welcome to Fundamentals of Programming'
>>> Welcome to Fundamentals of Programming
SyntaxError: invalid syntax
>>> Welcome
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    Welcome
NameError: name 'Welcome' is not defined
>>> |
```

Write Programming Statements and Display Output

- Following methods can be used to print numbers.

```
>>> 2
2
>>> 2+4
6
>>> ' 5 + 6 '
' 5 + 6 '
>>> print(2)
2
>>> print(2+4)
6
>>> print("5 + 6")
```

Write Programming Statements and Display Output

- To save and run a python file
  - Click File > New Window from the Python Shell editor.
  - Enter the statements in the new window and save it.
  - Click Run Menu and then select Run Module to run the programming statements in the file.
  - The output of the programming statements will be displayed in the Python Shall editor.

Write Programming Statements and Display Output

my_first.py - G:/my_first.py (3.5.0)

File Edit Format Run Options Window Help

```
x = 2
y = 4
print(x + y)
```

Python 3.5.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59
tel)] on win32
Type "copyright", "credits" or "license()" for more info
>>>
========================== RESTART: G:/my_first.py =====
6
>>>
```