Fundamentals of Programming

Unit 1
Introduction to Python

CUSCS

Course Introduction

Say “Hello” to everyone in our class!

Hello & Welcome

- Class Schedule
  - 6:45 p.m. – 9:45 p.m.
  - 10 lectures
  - Exam will be conducted at the last lecture
  - Important Note: Please note the class arrangements from the school.

Assessments

10%
Participation
Class Activities / Questions

50%
Assignments
Two individual assignments

40%
Examination
A two hour closed book exam

<u>Note</u>

1. Marks will be deducted for late submission or inappropriate submission format.
2. Total 100% for this course

Assessments

- Submission Requirements
  - Submit your assessments to the Moodle on or before the submission deadline
  - Write down and indicate your full name in your assessments
  - Submit the assessment in the required formats (Assessments in Inappropriate format may not be marked)
  - Marks will be deducted for late submission, or inappropriate file formats

Assessments

❑ Late Submission Penalty

- The submission time is based on the record shown in the Moodle.
- Deduct 10% of your assignment score for 1-day late submission
- Deduct 20% of your assignment score for 2-day late submission
- Deduct 50% of your assignment score for 3-day late submission
- **NO marks** for assignments submitted more than 3 days after the submission deadline

Introduction to Programming

Programming everywhere?

Programming

- Programming
  - the process of taking an algorithm and encoding it into a notation, a programming language
  - it can be executed by a computer
  - many programming languages and many different types of computers exist
- Before Working With Programming
  - the need to have the solution
  - without an algorithm there can be no program

Algorithm

Algorithm

describe the solution to a problem in terms of the data needed to represent the problem instance and the set of steps necessary to produce the intended result

❑ Meaning of Algorithms in Different Aspects

- a computer program can be viewed as an elaborate algorithm
- mathematics and computer science → an algorithm usually means a small procedure that solves a recurrent problem

Algorithm

❑ Programming and Algorithm

- step-by-step procedure to resolve any problem from programming point of view

$$\begin{tikzcd}\boxed{\text{Algorithm}} \arrow[r] & \boxed{\text{Programming}}\end{tikzcd}$$

- an effective method expressed as a finite set of well-defined instructions

- a computer programmer lists down all the steps required to resolve a problem before writing the actual code

Program Planning

Analyze

- define the problem and decide boundaries of problem

Design

- develop a step by step procedure to solve the problem

Interface Choosing

- gather the required resources to solve the problem

Code

- use a programming language to implement the instructions

Test & Debug

- check whether the code written is solving the specified problem or not

Documentation

- allow other people to understand the program

Maintenance

- program is actively used by the users
- if any enhancements found, all the phases are to be repeated to make the enhancements

Programming Environments

❑ Features

- the first step to be followed before setting on to write a program
- but environment Setup is not an element of any Programming Language

❑ Common Components for Setup

Text Editor

- create computer programs

Compiler

- compile the programs into binary format

Interpreter

- execute the programs directly

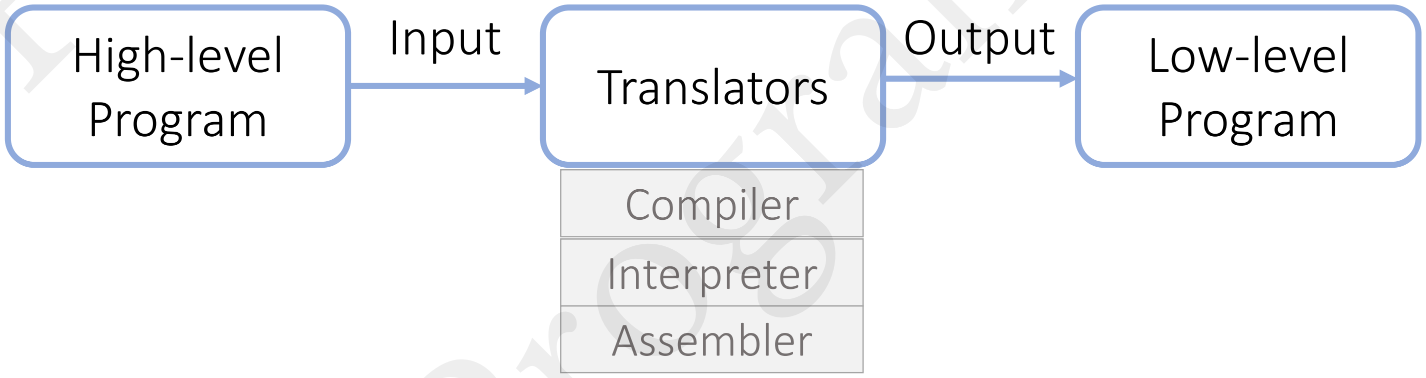
Text Editor

- Text Editor
  - a software that is used to write computer programs
- Use with Programming
  - use this software to type the computer program and save it in a file at any location
- Example
  - Notepad, Notepad++

Language Translators

❑ Translators

- the piece of software that translate a computer program written in some specific programming language into another programming language
- usually from a high-level programming language translated into low-level programming language / machine code



Introduction to Python

Then, let's move to Python!

Say ‘Hello’ to Python

Python

a general-purpose interpreted, interactive, object-oriented, and high-level programming language

Welcome Python?

- it was created by Guido van Rossum during 1985- 1990
- Python source code is also available under the GNU General Public License (GPL)

Main Features of Python

- Easy to Learn and Use
  - easy to learn as compared to other programming languages
  - its syntax is straightforward and much the same as the English language
  - the recommended programming language for beginners

- Free and Open Source
  - Python is freely available for everyone
  - freely available on its official website www.python.org

Main Features of Python

- Object-Oriented Language
  - Python supports object-oriented language and concepts of classes and objects come into existence
  - object-oriented procedure helps to programmer to write reusable code and develop applications in less code

- Extensible
  - other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code
  - it converts the program into byte code, and any platform can use that byte code

Python History and Versions

Python 1.0

In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce

Python 2.0

Python 2.0 added new features such as list comprehensions, garbage collection systems

Python 3.0

On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language

Python 2 vs. Python 3

- Common Practices
  - in most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language
  - easier for the projects to switch in the newer version

- Python 2 vs. Python 3
  - in the case of Python, the two versions Python 2 and Python 3 are very much different from each other

Python 2 vs. Python 3

- ❑ Major Differences between Python 2 and Python 3

Print Statement

- Python 2 uses print as a statement
- Python 3 uses print as a function

Implicit String Type

- ASCII in Python 2
- Unicode in Python 3

Accept User's Input

- Python 2 uses the function raw_input() and returns the string representing the value
- Python 3 uses input() function which automatically interpreted the type of input entered by the user

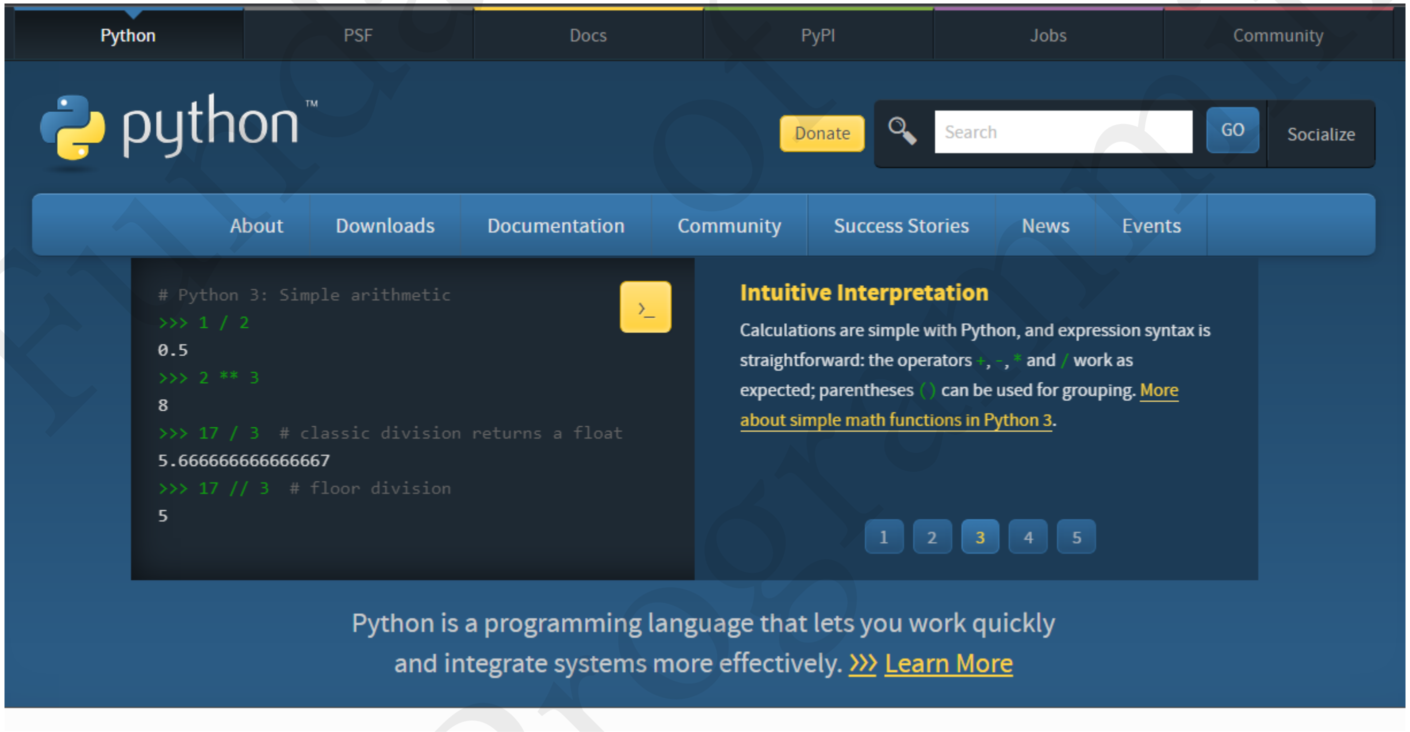
Setup Python

- ❑ Download Python
  - available on the official website of Python https://www.python.org/
- ❑ Download Python Documentation
  - from https://www.python.org/doc/
  - documentation is available in HTML, PDF, and PostScript formats

Setup Python

❑ Python Official Website

- https://www.python.org/



```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>> 17 // 3  # floor division
5
```

Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators +, -, * and / work as expected; parentheses () can be used for grouping. More about simple math functions in Python 3.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Python IDLE

Python IDLE

IDLE (Integrated Development and Learning Environment)
an integrated development environment (IDE) for Python

❑ Usages

- execute a single statement just like Python Shell and also to create, modify, and execute Python scripts
- provide a fully-featured text editor to create Python script
- has a debugger with stepping and breakpoints features

Python IDLE

❑ Python IDLE in Windows

- Python installer for Windows contains the IDLE module by default

❑ Start with Your Python IDLE

- search for the IDLE icon in the start menu and double click on it to start an IDLE interactive shell
- open IDLE and then we can write and execute the Python scripts

Python IDLE

❑ Approaches to Run Programs

1. Using Interactive interpreter prompt
2. Using a script file

❑ Interactive Interpreter Prompt

- Python provides us the feature to execute the Python statement one by one at the interactive prompt
- preferable in the case where we are concerned about the output of each line of our Python program

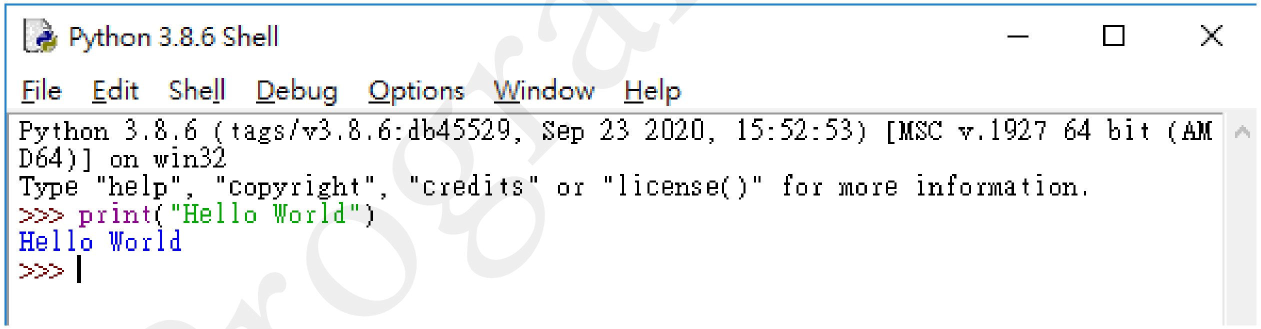
Interactive Interpreter Prompt

- Working Approaches
  - open the terminal / IDLE to write our Python statement
  - press the ***Enter*** key after writing the Python statement

- Example
  - Write ***print("Hello World")*** and press ***Enter*** in the IDLE

Programming Code

```
print("Hello World")
```



```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 
```

Interactive Interpreter Prompt

- Class Activity
  - Display “Hello everyone! I am Chan Tai Man!” where “Chan Tai Man” should be replaced as your name.

Python IDLE

❑ Limitation of Interactive Interpreter Prompt

- interpreter prompt is best to run the single-line statements of the code
- we cannot write the code every-time on the terminal
- not suitable to write multiple lines of code

❑ Script Mode Programming

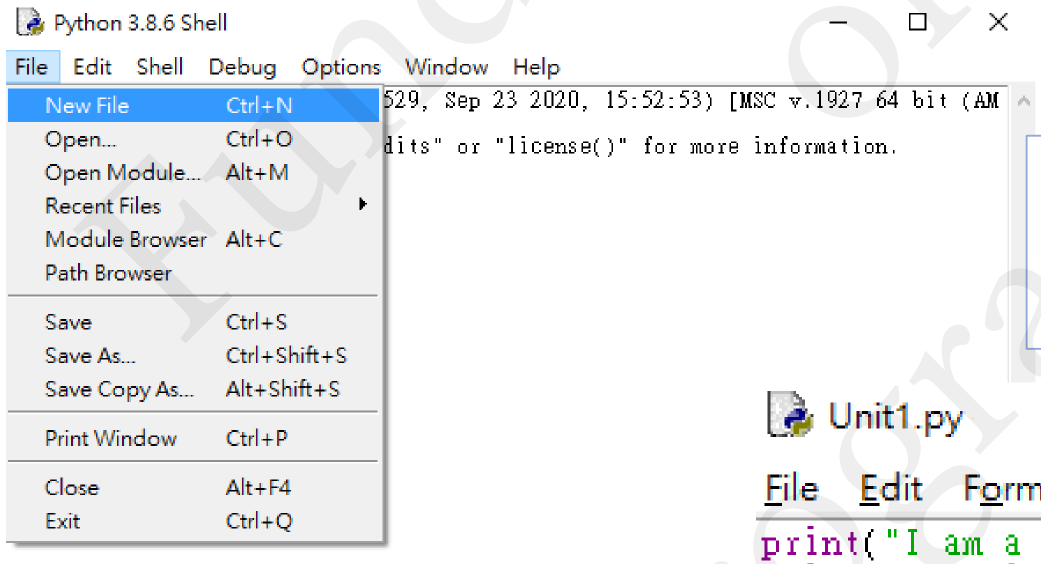
- using a script file
- write multiple lines code into a file which can be executed later
- file with .py extension, which stands for "Python"

Script Mode Programming

- Working Approaches
  - under File menu, select New File or press Ctrl + N to create Python code file
  - insert code and save Python code file that will bring up a new window called Untitled
  - enter the following command in the new window
  - save your Python code in a .py file
  - run Python code by simply clicking Run -> Run Module

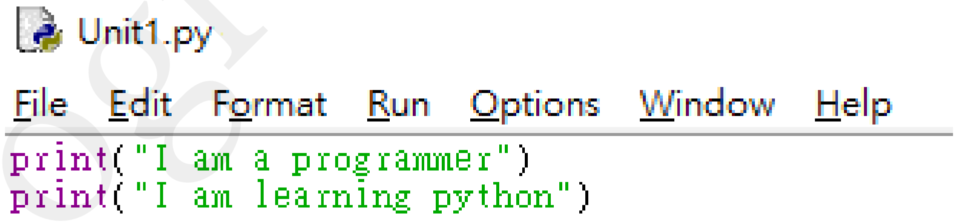
Script Mode Programming

- ❑ Example
  - Create a Python file called "Unit1.py"
  - Input the following code and run the program after saving the file



```
529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
dits" or "license()" for more information.
```

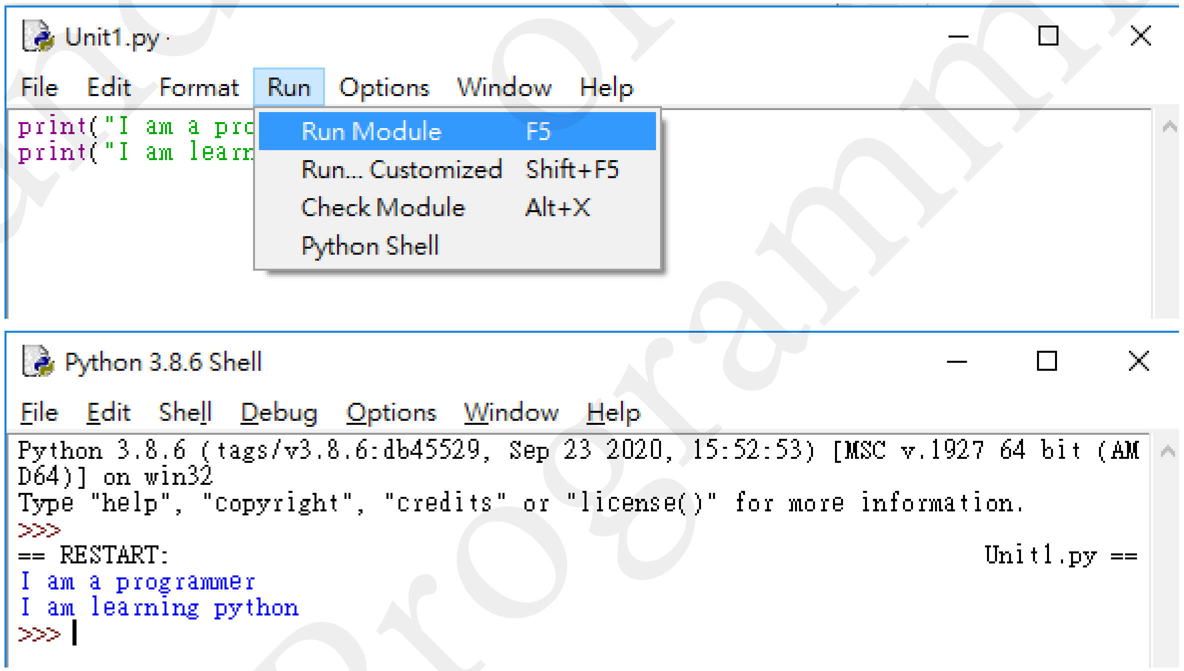
Programming Code
```
print("I am a programmer")
print("I am learning python")
```



```
print("I am a programmer")
print("I am learning python")
```

Script Mode Programming

- ❑ Example
  - Create a Python file called “Unit1.py”
  - Input the following code and run the program after saving the file

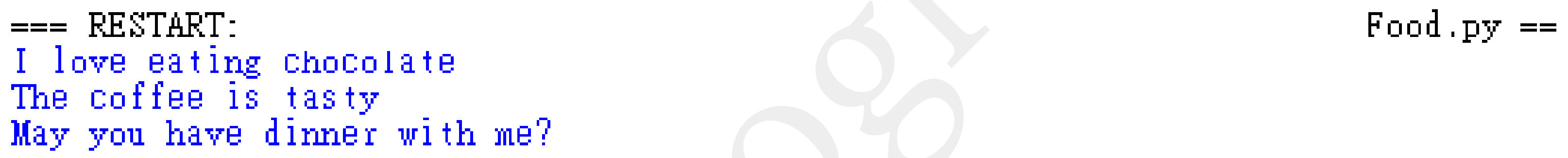

Unit1.py
File Edit Format Run Options Window Help
Run Module F5
Run... Customized Shift+F5
Check Module Alt+X
Python Shell
Python 3.8.6 Shell
File Edit Shell Debug Options Window Help

```
print("I am a pro
print("I am learn
```

```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART:                                                          Unit1.py ==
I am a programmer
I am learning python
>>> |
```

Script Mode Programming

- Class Activity
  - Create a Python file called “Food.py”
  - Display three statements which are:
    1. I love eating chocolate
    2. The coffee is tasty
    3. May you have dinner with me?
  - Run your Python script to show the results



```
=== RESTART:                                                   Food.py ==
I love eating chocolate
The coffee is tasty
May you have dinner with me?
```

Script Mode Programming

- ❑ Advantages
  - can run multiple lines of code
  - debugging is easy in script mode
  - appropriate for beginners and also for experts

- ❑ Disadvantages
  - have to save the code every time if we make any change in the code
  - can be tedious when we run a single or a few lines of code

Basic Input & Output

Display our information and ask for user's information

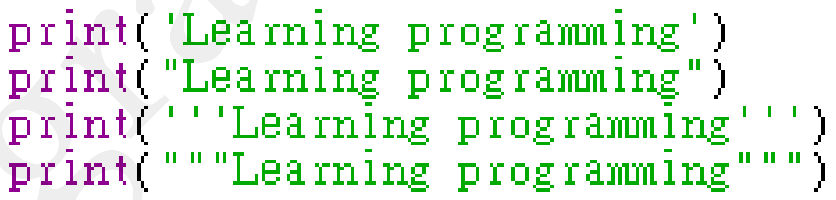
Python Basic Output

- ❑ print() function
  - the print() function displays the string enclosed inside the single quotation
  - simply use the print() function to print output

- ❑ Example

<u>Programming Code</u>

```
print('Learning programming')
print("Learning programming")
print('''Learning programming''')
print("""Learning programming""")
```



```
print('Learning programming')
print("Learning programming")
print('''Learning programming''')
print("""Learning programming""")
```

Python Basic Output

❑ Syntax of print() function

```
print(object= separator= end= file= flush=)
```

Parameter	Description
object	▪ value(s) to be printed
sep (optional)	▪ allows us to separate multiple objects inside print()
end (optional)	▪ allows us to add add specific values like new line "\n", tab "\t"
file (optional)	▪ where the values are printed. It's default value is sys.stdout (screen)
flush (optional)	▪ boolean specifying if the output is flushed or buffered. Default: False

Python Basic Output

❑ Example

- Create a test.txt in the same directory of your python file

```
print("Python")
print("Python", "Programming", sep = "*")
print("Python", "Programming", sep = "*", end = "@")

myFile = open('test.txt', 'w')
print("Python", "Programming", sep = "*", end = "@", file = myFile)
myFile.close()

print("Python", "Programming", sep = "*", end = "@", flush = False)
```

Python Basic Output

- Example
  - Create a test.txt in the same directory of your python file

<u>**Programming Code**</u>

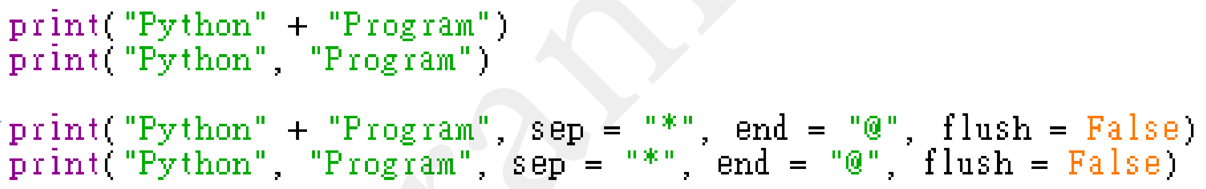
```
print("Python")
print("Python", "Programming", sep = "*")
print("Python", "Programming", sep = "*", end = "@")

myFile = open('test.txt', 'w')
print("Python", "Programming", sep = "*", end = "@", file = myFile)
myFile.close()

print("Python", "Programming", sep = "*", end = "@", flush = False)
```

Python Basic Output

❑ Concatenation of print() function

- string concatenation means add strings together
- Approach 1: using + operator
- Approach 2: using , operator

❑ Example



```
print("Python" + "Program")
print("Python", "Program")

print("Python" + "Program", sep = "*", end = "@", flush = False)
print("Python", "Program", sep = "*", end = "@", flush = False)
```

<u>Programming Code</u>

```
print("Python" + "Program")
print("Python", "Program")
print("Python" + "Program", sep = "*", end = "@", flush = False)
print("Python", "Program", sep = "*", end = "@", flush = False)
```

Python Basic Input

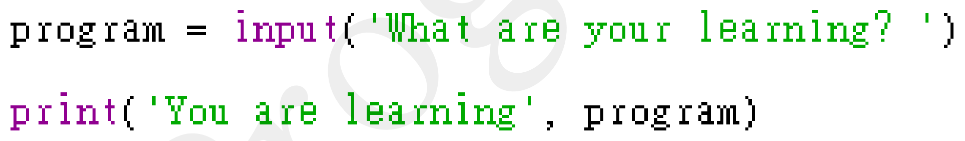
❑ input() function

- take the input from the user. In Python using the input() function
- Input(prompt) function → prompt = the string we want to display on the screen

❑ Example

<u>**Programming Code**</u>

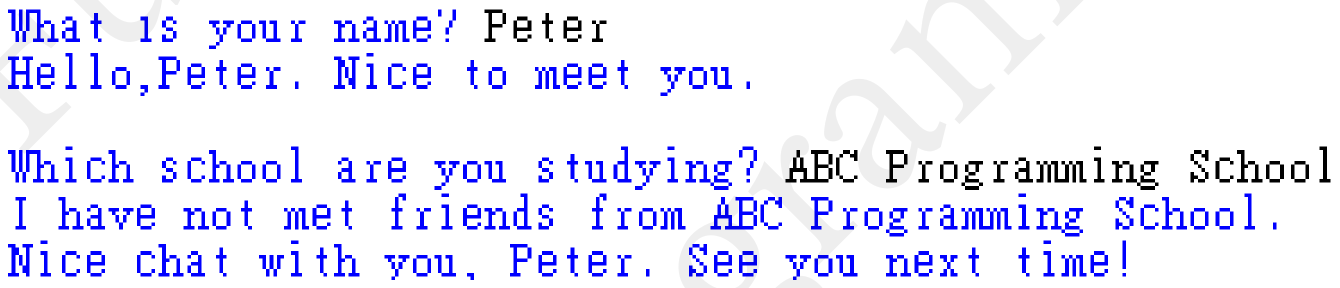
```
program = input('What are your learning? ')
print('You are learning', program)
```



```
program = input('What are your learning? ')

print('You are learning', program)
```

Python Basic Input

- Class Activity
  - Create a Python file called “Welcome.py”
  - Ask the user for their names and their schools
  - Display welcome messages with their names and schools
  - The sample output is shown below



```
What is your name? Peter
Hello,Peter. Nice to meet you.

Which school are you studying? ABC Programming School
I have not met friends from ABC Programming School.
Nice chat with you, Peter. See you next time!
```

Hope you enjoy the class

Thank you