Programing in Python

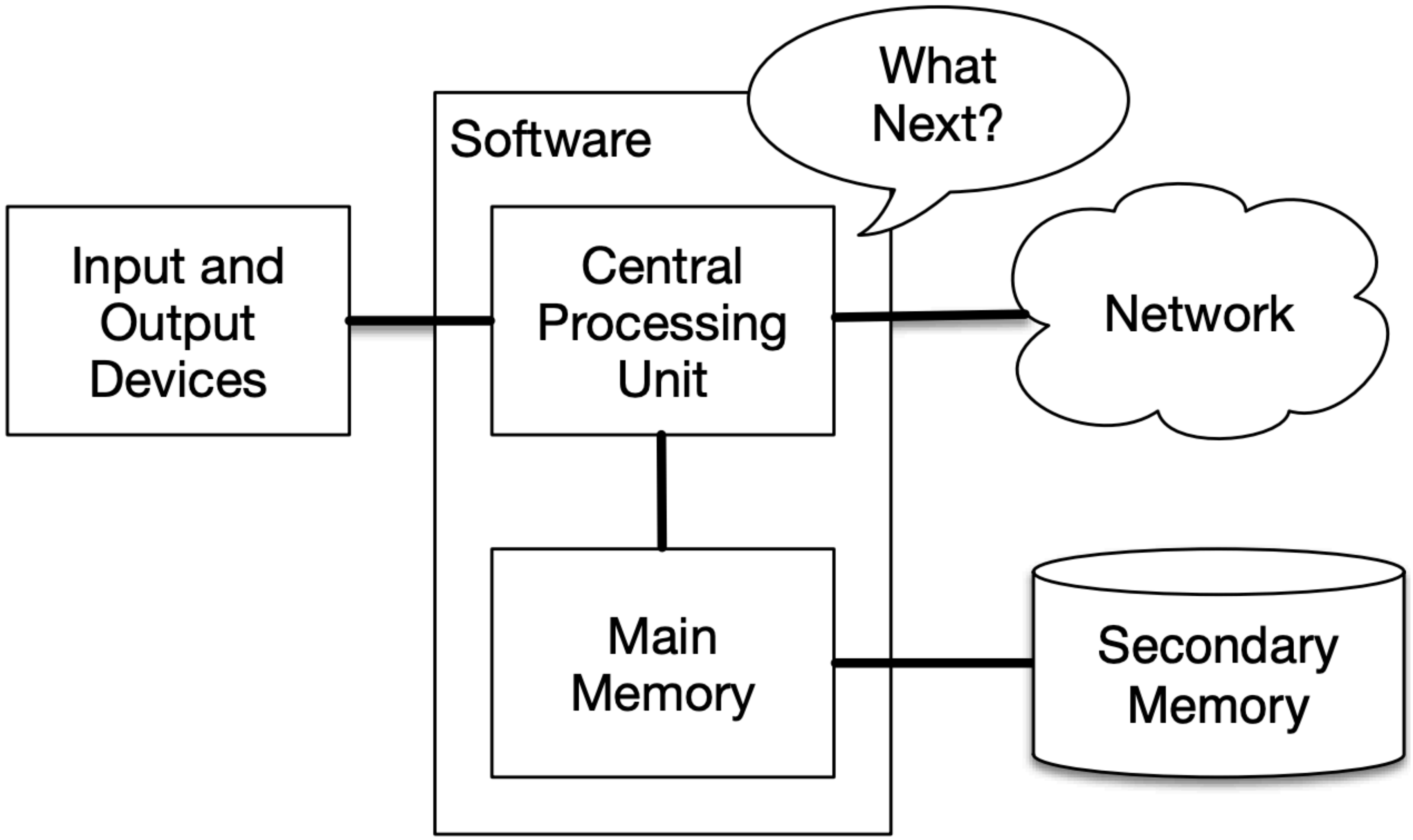
Lecture 2 - The Basics

Instructor: Zhandos Yessenbayev

Outline

- Computer Architecture
- What is a program?
- Basic Concepts

Computer Architecture



Computer Architecture

- The **Central Processing Unit** (or CPU) *executes instructions* (basic arithmetic, logic, controlling, and input/output (I/O) specified by the programs.
- The **Main Memory** *stores information* that the CPU needs immediately. It is nearly as fast as the CPU, but the information vanishes when the computer is turned off.
- The **Secondary Memory** also *stores information* on hard drives or flash memory. It is much slower than the main memory, but can store information even when there is no power to the computer.
- The **Input and Output Devices** are used to *interact with the computer*. Examples are screen, keyboard, mouse, microphone, speaker, touchpad, etc.
- **Network Connection** is used to *retrieve information over a network*. The network is a slower and at times unreliable form of secondary memory.

What is a Program?

- A **computer program** is a collection of instructions that can be executed by a computer to perform a specific task.
- **Programs**:
  - are written as *source code*, stored/loaded in *memory*, executed by *CPU*
  - can be *compiled* (translated) to machine code or *interpreted* immediately
  - can take some *input* and produce some *output*

High-level program (Python)	Low-level program (Assembly)

`print("Hello, World!")`	
	```
	0 LOAD_NAME          0 (print)
	2 LOAD_CONST         0 ('Hello, World!')
	4 CALL_FUNCTION      1
	6 POP_TOP
	8 LOAD_CONST          1 (None)
	10 RETURN_VALUE
	```

Compiler vs Interpreter

- **Compiler**
    - Converts high-level code to low-level machine code to create executable program
    - Executes very fast, but needs more time for testing and debugging
- **Interpreter**
    - Converts high-level code to low-level machine code and executes it line by line
    - Executes slow, but good for testing and debugging

What is Python?

- **Python** is a programming language which is:
  - general purpose (AI, data science, web, robotics, etc)
  - interpreted (executed on-the-fly)
  - object-oriented (can define **classes**)
  - high-level (human readable)
  - with dynamic semantics (dynamic **objects**).

Running Python

- **Python** can be run in *interactive mode*

```
$ python3

Python 3.8.2 (default, Dec 21 2020, 15:06:04)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
>>> "Hello, world!”
'Hello, world!'
>>>
>>> quit()
```

- **Python** can be run in *script mode*

```
$ echo “Hello, world!” > hello.py
$ python3 hello.py
```

Variables and Types

- **Variable** type depends on its value and can be:
  - Integers - 2, 4, 5
  - Floating point number - 2.5, 40.0
  - String - “Hello”, ‘World’
  - Boolean - True, False
- type(*variable*) - shows type of the variable

```
>>> type(2)          >>> type('Hello, World!')
<class ‘int’>        <class ‘str’>
>>> type(2.0)        >>> type(True)
<class ‘float’>       <class ‘bool’>
```

- To convert data from one type to another, use class name as a function:

```
>>> int(‘2’)    >>> bool(5)
2               True
>>> str(4)      >>> float(3)
‘4'             3.0
```

Input and Output

- To input some data to the Python program, **input()** function is used:
    - input(<some text>) — returns data as string

```
>>> n = input('Enter a number: ')
Enter a number: 5
>>> n
'5'
```

- To output some data to the Python program, print() function is used:
    - print(<comma separated data — text, variables, expressions>)

```
>>> print(2+3)
5
>>> print('Hello')
Hello
>>> a = 6
>>> print(‘a =', a)
a = 6
>>>
```

- * Functions will be discussed in the future lectures.

Arithmetic operators

- Python recognizes the following **arithmetic operators**:
    - Multiplication (*****) — 3*2=6
    - Division (**/**) — 3/2=1.5
    - Subtraction (**-**) — 3-2=1
    - Addition (**+**) — 3+2=5
    - Exponentiation (******) — 3**2=9
    - Bitwise operations (**<<, >>, |, &, ~, ^**) — 101&100=100

Assignment statements

- **Assignment statement** sets some value to a variable
  - Variable = Value
    - X = 5
    - Y = X
    - X = “String”
    - Z = X = 1.5
    - a, b = 1, 2

Y = ?   Z = ?

* Think of a variable as a box being filled with some value.

Variable Names

- **Variable names:**
- can be letters, numbers and special symbol (_)
- can be very long
- cannot start with numbers
- cannot be a reserved keyword (below)

`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`	

Expressions and Statements

- An **expression** is a combination of values, variables, and operators:

```
>>> 40+2
42
>>> “hello”
‘hello’
```

- A **statement** is a unit of code that has an effect, like creating a variable or displaying a value.

```
>>> n = 17
>>> print(n)
```

Order of operations

Order of operations follow the rule - **PEMDAS**:

- Parentheses — (5+3)*4 = 32
- Exponentiation — 2+2**4 = 18
- Multiplication and Division — 2*3-1= 5, 6+4/2=8
- Addition and Subtraction
- Equal operations evaluated from left to right — 6/2*3=9

String operations

- Two operations are important for strings (+, *)
  - ‘hello’ + ‘ ‘ + ‘world’ = ‘hello world’
  - ‘hello’ * 2 = ‘hellohello’

```
>>> first = 'silent'
>>> second = 'breeze'
>>> first + second
silentbreeze
```

Comments

```
# This is a single line comment
print(“This is not a comment”)

print(“This is not a comment”)   # This is also a comment

# These are
# multiple-line
# comments
print(“This is not a comment”)

“””
These are
multiple-line
comments
“””
print(“This is not a comment”)
```

Debugging

Three kinds of errors can occur in a program:

- **Syntax errors** - errors that violate the structure of a program, eg: (1+3 - is illegal. *Program will not run!*

- **Runtime errors** - error that appear during the execution of a program (also called *exceptions*), eg: 1/0. *Program will run, but fail at some point!*

- **Semantic errors** - errors in the logic or computation of a program, eg: sphere_perimeter = 3.14 * R.
  Program will run, but will not produce the right result!

- Use print() function to see intermediate results and debugging

Python Documentation

The official Python3 documentation can be found at:

- https://docs.python.org/3/

Thanks!