



Introduction to Programming

Department of Computer Science and Information
Systems

Lecturer: Steve Maybank

sjmaybank@dcs.bbk.ac.uk

Autumn 2019 and Spring 2020

Week 1: First Program



Module Information

- Full time: 14.00-17.00 on Tuesdays in the autumn term.
 - 14.00 – 15.20 Lecture: MAL 421
 - 15.40 – 17.00 ITS Lab: MAL 109
- Part time: 18.00-21.00 on Tuesdays in the autumn term.
 - 18.00 – 19.20 Lecture: UCL Torrington Place 1-19, Room G13
 - 19.40 – 21.00 ITS Lab: MAL 109



Assessment

- Lab attendance (9 classes): 10%
- In lab test (Week 11): 20%
- Two hour written examination in summer 2020: 70%

- **Pass: an overall mark of at least 40%**
- Example:
 - 6 lab classes, 45% in lab test, 38% examination.
 - Overall mark:

$$((6/9)*100)*(1/10)+45*(2/10)+38*(7/10) = 42.27$$



Tests and Examinations

- Week 10 – first half: **mock** examination
- Week 10 – second half: **mock** in laboratory test
- **Week 11 – second half: in laboratory test**

- The mock examination and the mock in lab test are for **practice only**. They will not be marked.



Teaching Materials

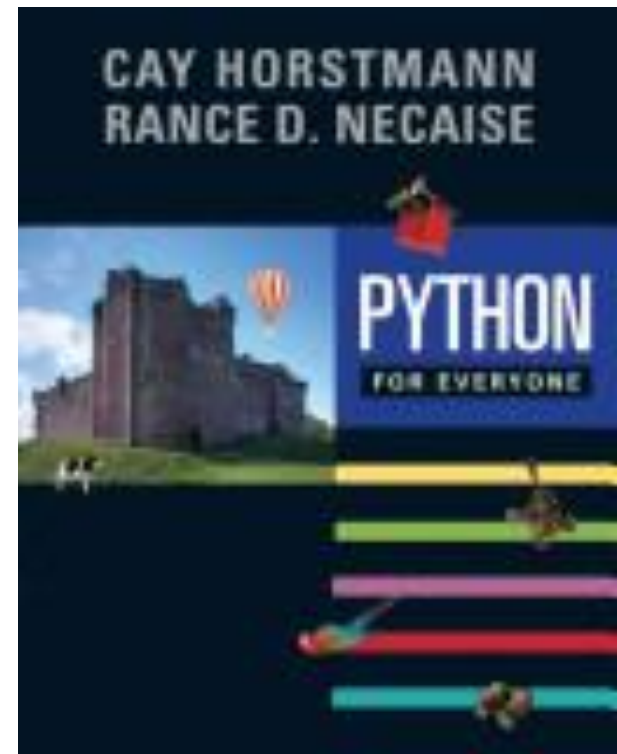
- Timetable, syllabus, slides, lab worksheets, example programs, etc. will be posted on my ITP web page:

<https://www.dcs.bbk.ac.uk/~sjmaybank/ITP/introduction%20to%20Programming.html>

- Moodle will be used for messages to the class

Textbook

- Essential: Cay Horstmann and Rance Necaise (2014)
Python for Everyone, Wiley
- Teaching is based on the **first six chapters** of PFE
- The lab classes are based on **exercises** in PFE





Syllabus

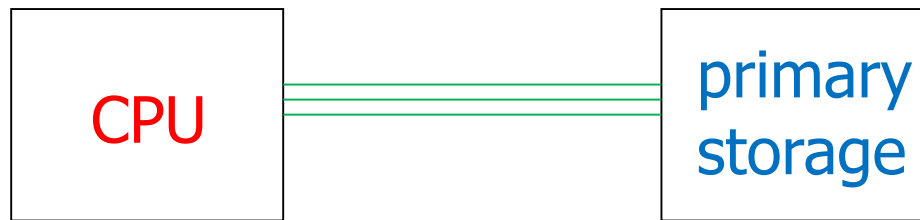
- First program: `print("Hello World")`
- Safe operation of equipment: avoid RSI (Repetitive Strain Injury)
- Variables: `q = 2`
- Pseudo code and algorithm design
- Arithmetic and Input: `(1+4)/5; input("Type a number")`
- Strings and Output: `print("Hello World"); print(q)`
- Relational operators and Boolean variables: `2 < 5`
- if statement: `if (2 < 5) :`
- Loops: `while (q < 3) :`
- Functions: `q = max(2, 3)` - encapsulate
- Lists: `[4, -5, 2]`



This Lecture

- Based on Ch. 1 of PFE
- Aim 1: provide background information on computing
- Aim 2: provide enough information to write a first Python program.

Structure of a Computer



- **Central Processing Unit:** executes in sequence small fragments of code known as *instructions*.
- **Primary storage (= main memory):** stores programs and data required by the CPU
- **Bus:** connects the CPU and the primary storage



Peripheral Devices

- Input devices:
 - mouse, keyboard, microphone, touchpad
- Output devices:
 - printer, monitor, speakers
- Input and Output device:
 - hard drive – secondary storage
 - large capacity storage of programs and data



Problem

- The CPU of a computer can only carry out a few simple instructions known as **machine code**

```
011010100110101000101101100100101011001010101001010101
01111000101011110001101110111000101010010101001101010100
01010100010010010110101000101001011100011001010100100110
00110101010111101011011110100100100010110101010100000101
00110101001101010001011011001001010110010101010100101010
1011110001010111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
0011010100110101000101101100100101011001010101010100101010
1011110001010111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
0011010100110101000101101100100101011001010101010100101010
1011110001010111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
0011010100110101000101101100100101011001010101010100101010
1011110001010111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
0011010100110101000101101100100101011001010101010100101010
1011110001010111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
0011010100110101000101101100100101011001010101010100101010
1011110001010111000110111011100010101001010100110101010
```

- It is **time consuming** and **error prone** to write programs using these simple instructions



Solution

- Write programs in a **high level language** which is easier to understand than **machine code**.
- Use another program to **convert high level programs** into lists of **machine code** instructions for the CPU.
- Python is a **high level programming language**.

Python

- Developed in the late 80s and early 90s by **Guido van Rossum**
 - National Research Institute for Mathematics and Computer Science (CWI), The Netherlands
 - Google (2005-2012)
 - Dropbox (2013-)
- Aim: to produce a language in which small programs can be written quickly



Python

- The name: from Monty Python's Flying Circus

Python 0.9.0: year 1991

Python 2.0: year 2000

Python 3.0: year 2008

Now:

Python 3.7.0 June, 2018





Advantages of Python

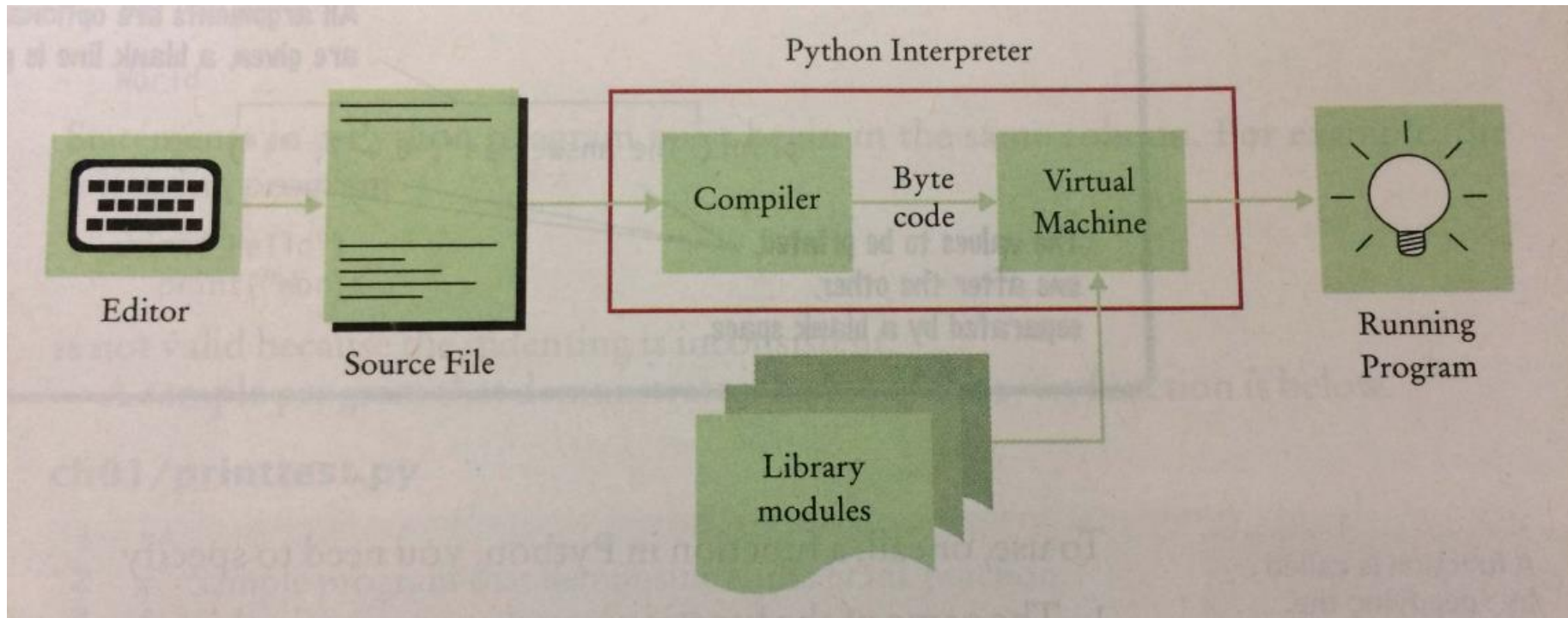
- **Simple syntax** (= grammar)
- **Portable** without change to different operating systems
 - Windows, UNIX, Linux and Mac
- Easy to write programs for **complex data**
- Very **large standard library**
 - text processing,
 - data compression,
 - file formats,
 - mathematical functions,
 - ...



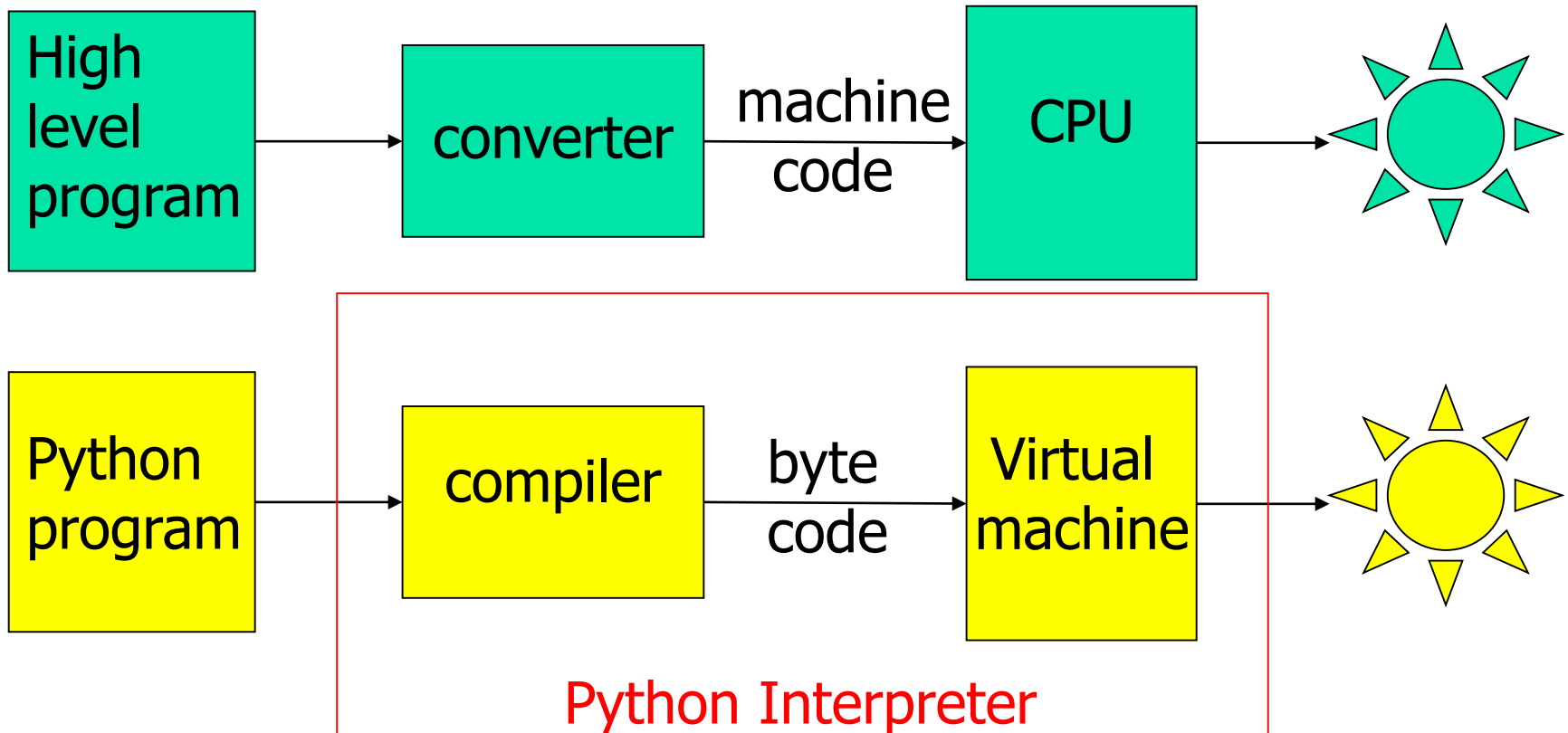
Python Interpreter

- The interpreter consists of a **compiler** and a **virtual machine**
- The **compiler** converts Python instructions to simpler instructions known as *byte code*
- The **virtual machine** is a software version of a CPU. It runs the byte code

From Source Code to Running Program



Comparison





Portability

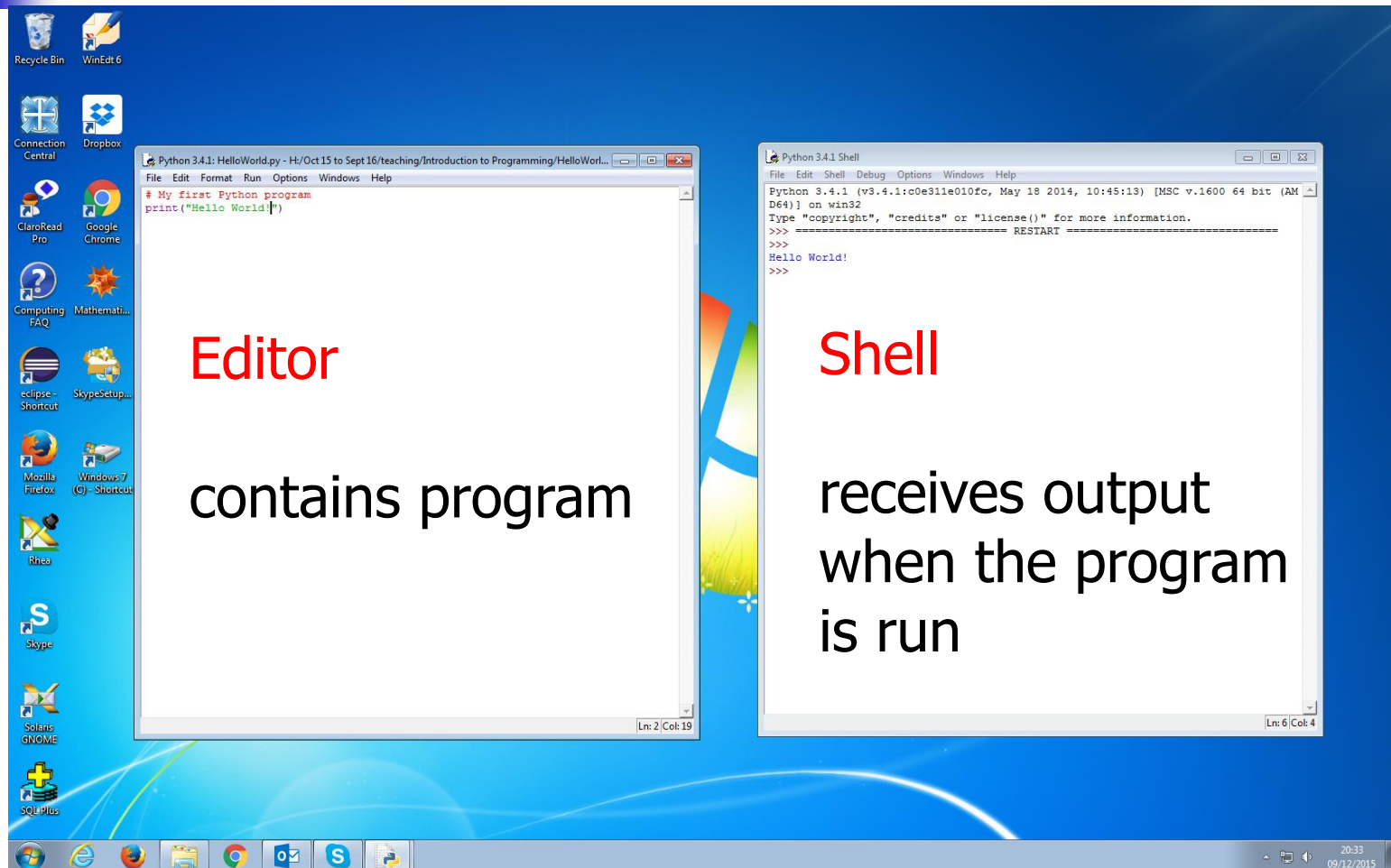
- The virtual machine is not portable
- Once the virtual machine is installed, it can run the byte code from any Python program



Integrated Development Environment (IDE)

- Our IDE for Python is **IDLE**
- IDLE facilities:
 - Create file for program
 - Edit program file
 - Run program
- See laboratory session

IDLE Editor and Shell



The image shows a Windows desktop with two windows from the IDLE Python environment. The left window is the 'Editor' window, titled 'Python 3.4.1: HelloWorld.py - H:/Oct 15 to Sept 16/teaching/Introduction to Programming/HelloWor...'. It contains the following code:

```
# My first Python program
print("Hello World!")
```

The right window is the 'Shell' window, titled 'Python 3.4.1 Shell'. It shows the output of running the program:

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Hello World!
>>>
```

Red text 'Editor' and 'Shell' is overlaid on the respective windows. Below the editor window, the text 'contains program' is written. Below the shell window, the text 'receives output when the program is run' is written.



My First Program

test.py - /Users/tingtinghan/Desktop/test.py (3.6.4)

```
# My first program  
print("Hello World!")
```

Ln: 3 Col: 0

Python 3.6.4 Shell

```
Python 3.6.4 (v3.6.4:d48ecebada5,  
Dec 18 2017, 21:07:28)  
[GCC 4.2.1 (Apple Inc. build 5666  
) (dot 3)] on darwin  
Type "copyright", "credits" or "l  
icense()" for more information.
```

```
>>>
```

```
===== RESTART: /Users/  
tingtinghan/Desktop/test.py =====
```

```
Hello World!
```

```
>>>
```

Ln: 7 Col: 4

When the above program is run in IDLE the string "Hello World!" appears in the shell screen



Commentary

- `# My first program` is a comment. It is ignored by the interpreter
- `print("Hello World!")` is a statement
- `print` is the name of a function
- `print("Hello World!")` is a function call
- The string `"Hello World!"` is an argument for the function `print()`

Colour Coding in IDLE

- Red for comments: `# My first program`
- Purple for functions: `print(...)`
- Green for data: `"Hello World!"`
- Blue for output: `"Hello World!"`

```
test.py - /Users/tingtinghan/Desktop/test.py (3.6.4)
# My first program
print("Hello World!")
Ln: 3 Col: 0
```

```
Python 3.6.4 Shell
Python 3.6.4 (v3.6.4:d48eceb5,
Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666
) (dot 3)] on darwin
Type "copyright", "credits" or "l
icense()" for more information.
>>>
===== RESTART: /Users/
tingtinghan/Desktop/test.py =====
Hello World!
>>>
Ln: 7 Col: 4
```




More About Functions

- A function is a collection of programming instructions that carry out a particular task
- Example: `print("Hello World!")`
- We know the **name** of the function, the **data supplied** to the function and the **data obtained** from the function (in this example as printed output)
- The programming instructions within the function `print` are hidden



Calls to print

```
print("The answer is", 6+7, "!") # three arguments
```

```
# The output is
```

```
The answer is 13 !
```

```
# Note added spaces
```

```
print("Hello") # one argument
```

```
print() # no arguments. A blank line is printed
```

```
print("World") # one argument
```

```
# The output is
```

```
Hello
```

```
World
```



Errors

- Python is case sensitive:

```
Print("Hello World!") # error if print is intended
```

```
PRINT("Hello World!") # error if print is intended
```

- Syntax errors:

```
print(Hello World!)
```

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



Indentation

- # Statements must begin in the **same column**
- # The following statements are in error

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



Compile Time Error

- An **error in the syntax** (grammar of Python) is detected by the **compiler**, e.g.

```
print(Hello World!)
```
- An error message is produced, in this case
SyntaxError: invalid syntax
- The error must be corrected `by hand`



Run Time Errors

- **Run time exception**: the program is compiled but the run time process stops when the error is encountered, e.g.

```
print(1/0)
```

- A run time exception produces an error message, in this case
ZeroDivisionError: int division or modulo by zero

- **Run time error** (but not an exception): the program runs but does not produce the desired result, e.g.

```
print("Helo World!")
```



Questions

- What does this program print?

```
print("39+3")  
print(39+3)
```
- What does this program print?

```
print("Hello", "World", "!")
```
- What is the error in this program?

```
print("Hello", "World!")
```

 - Is it a **compile time error** or
 - **run time exception** or
 - **run time error**?