The Center for the Development of the Gifted and Talented
香港科技大學 資優教育發展中心

D002 Python for Everyone

Lecture 1 Python Basics

Dr. Kevin WANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

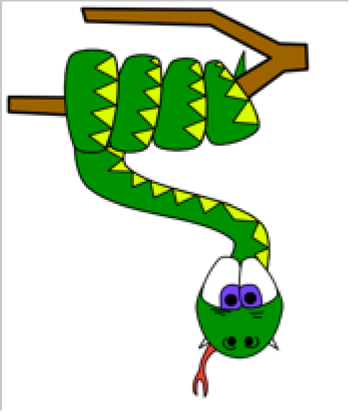
Course administration issues

- Instructor: Dr. Kevin Wang
- Teaching assistants: David Lin, Terry Lam, Yubo Tang
- Course schedule
  - Computer Barn A (Room 4402, Lift 17-18)
  - Lecture & Practice: Aug 7,9,12,14,15,16 10:00am – 12:30pm
- Online learning system: ~~https://epst.ust.hk/~~ github.com/khwang0/D002-2019
- Grade
  - Certificate will be awarded to those who attend over 80% of the class (prior approval needed for leave application)
  - The certificate is with Distinction, Merit and Completion classifcations

Course administration issues

- ❑ Open a Github account to save your work. Search **D002-2019**
  - ➢ We will grade your work on Github.
- ❑ You are suggested to use our PC here
- ❑ You need to do some homework with a PC/Laptop (either a mac/**Windows**/Linux)
  - ➢ Install Python at home ← **Don't use Python 2.x. Use Python 3.6 or higher**
- ❑ You might bring your laptop
  - ➢ Don't forget your charger

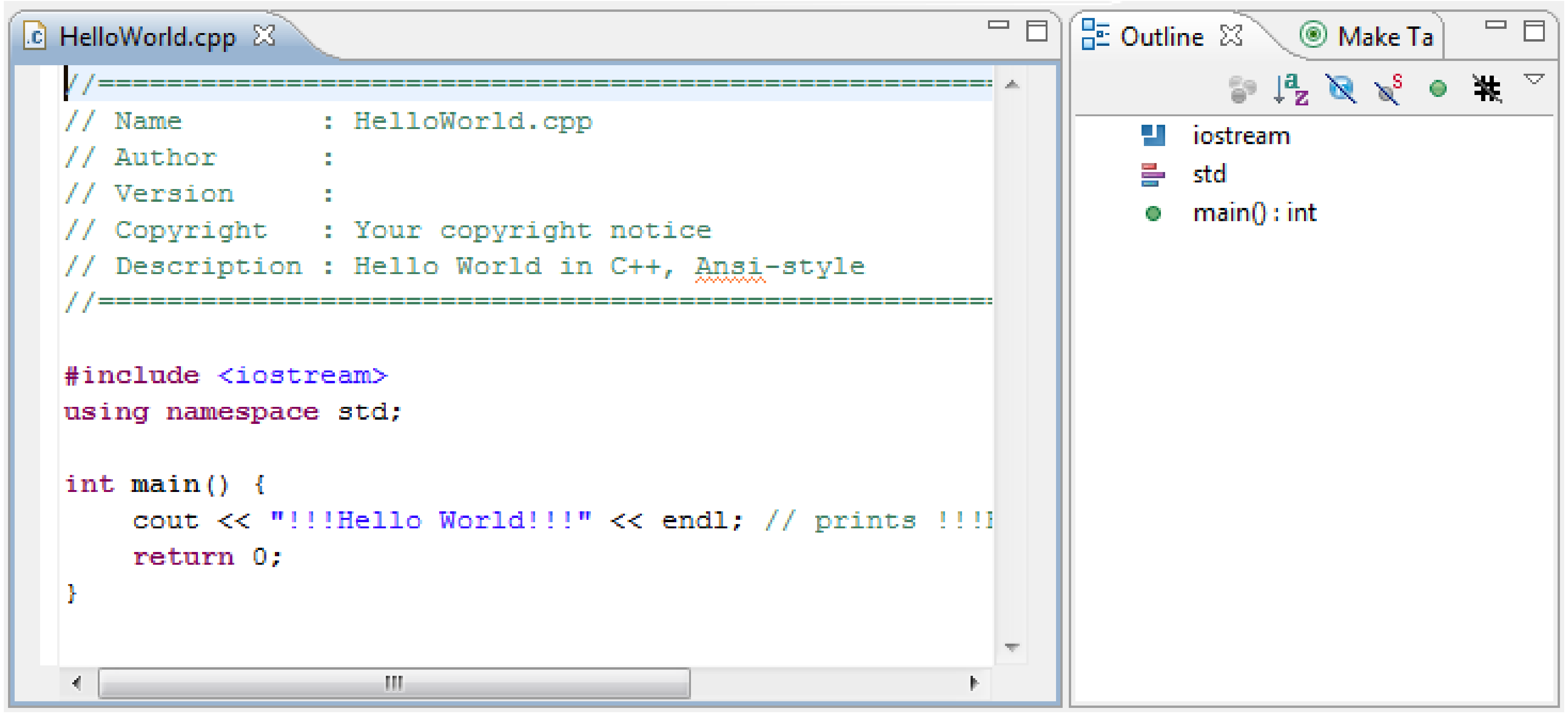
- ❑ Put your phone away during the class, unless it is ringing
- ❑ Most important: ask if you don't understand



A Bit Intro to Python

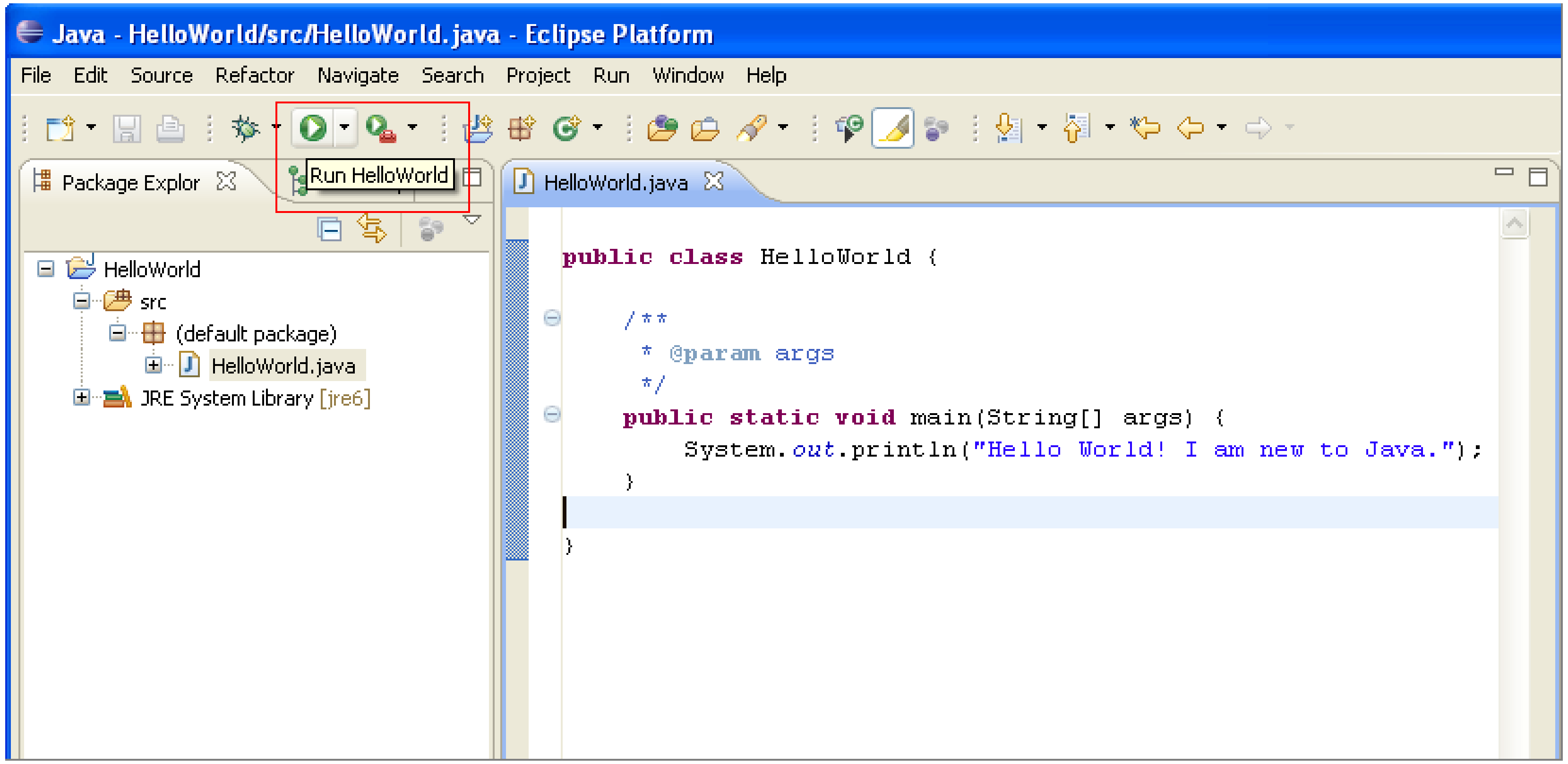
History and Features

Hello World in C++



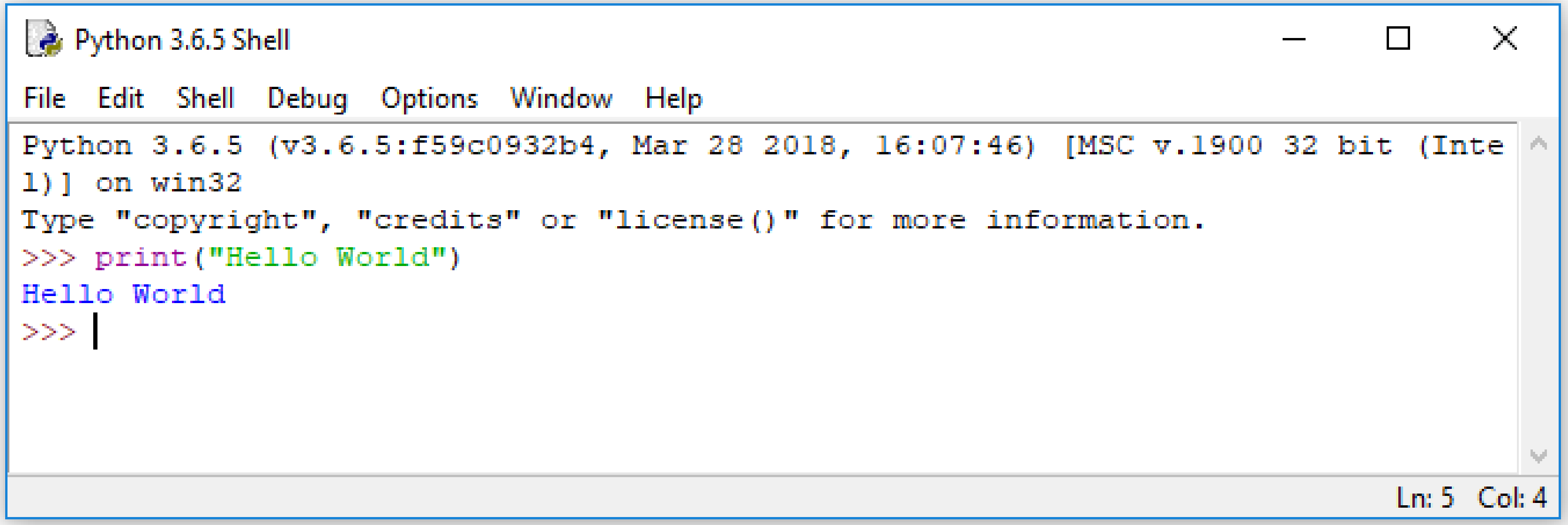
```
//============================================================
// Name        : HelloWorld.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!I
    return 0;
}
```

Hello World in Java


Java - HelloWorld/src/HelloWorld.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Run HelloWorld
Package Explor
HelloWorld.java
HelloWorld
src
(default package)
HelloWorld.java
JRE System Library [jre6]

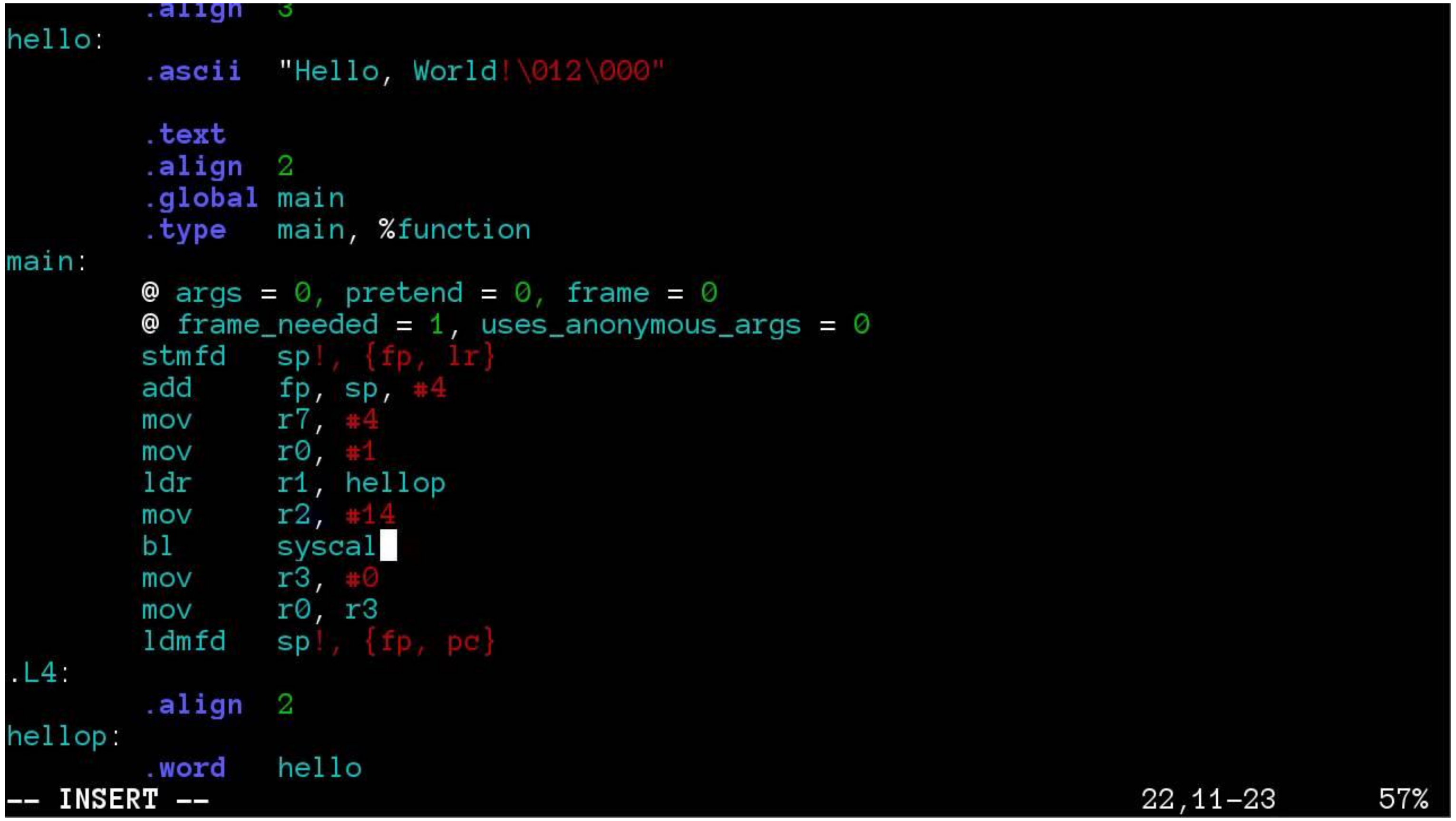
```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World! I am new to Java.");
    }

}
```

Hello World in Python



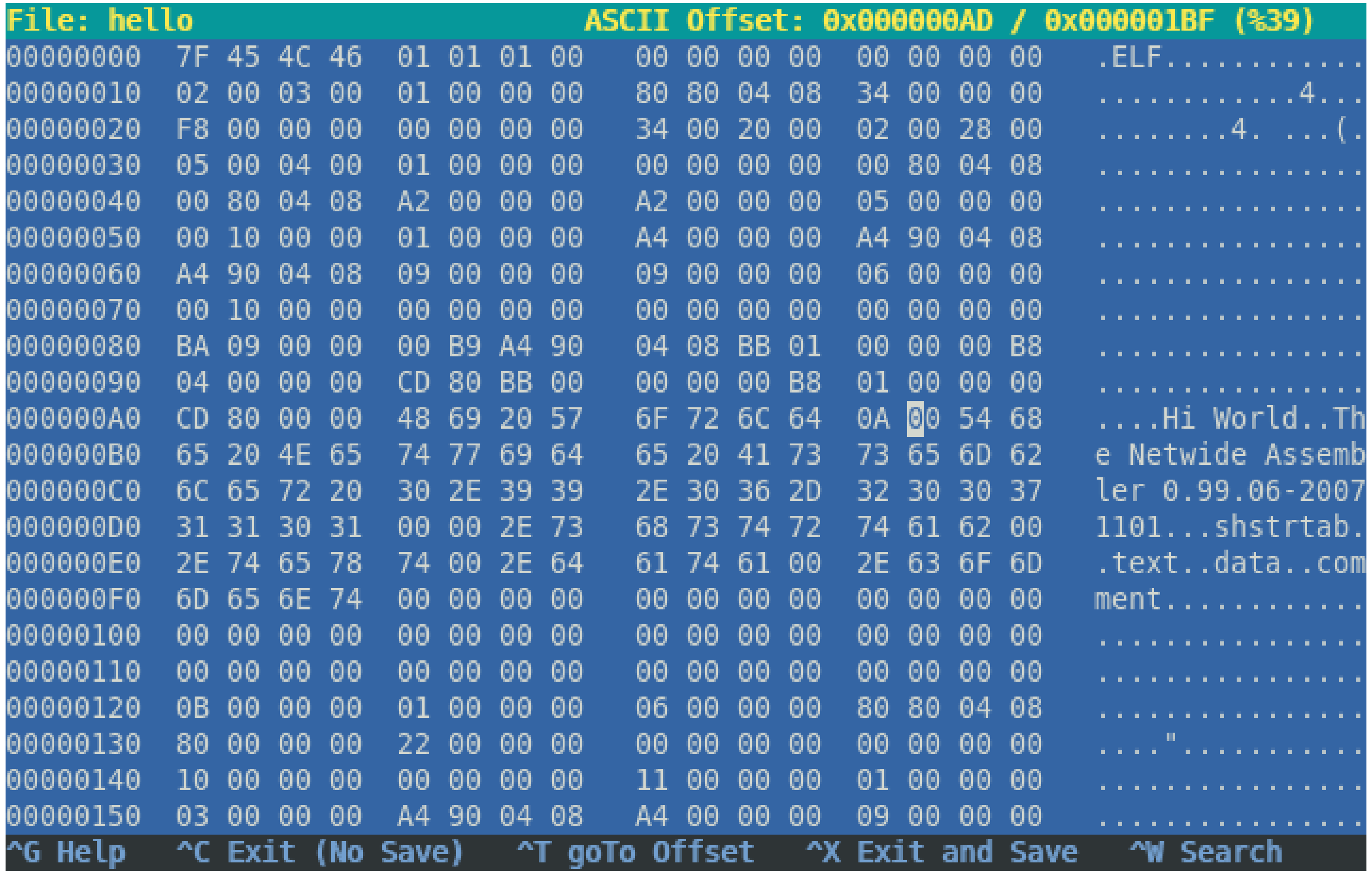
```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 
```

Hello World in ARM assembly



```
        .align  3
hello:
        .ascii  "Hello, World!\012\000"

        .text
        .align  2
        .global main
        .type   main, %function
main:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 1, uses_anonymous_args = 0
        stmfd   sp!, {fp, lr}
        add     fp, sp, #4
        mov     r7, #4
        mov     r0, #1
        ldr     r1, hellop
        mov     r2, #14
        bl      syscal
        mov     r3, #0
        mov     r0, r3
        ldmfd   sp!, {fp, pc}
.L4:
        .align  2
hellop:
        .word   hello
-- INSERT --                                                    22,11-23      57%
```

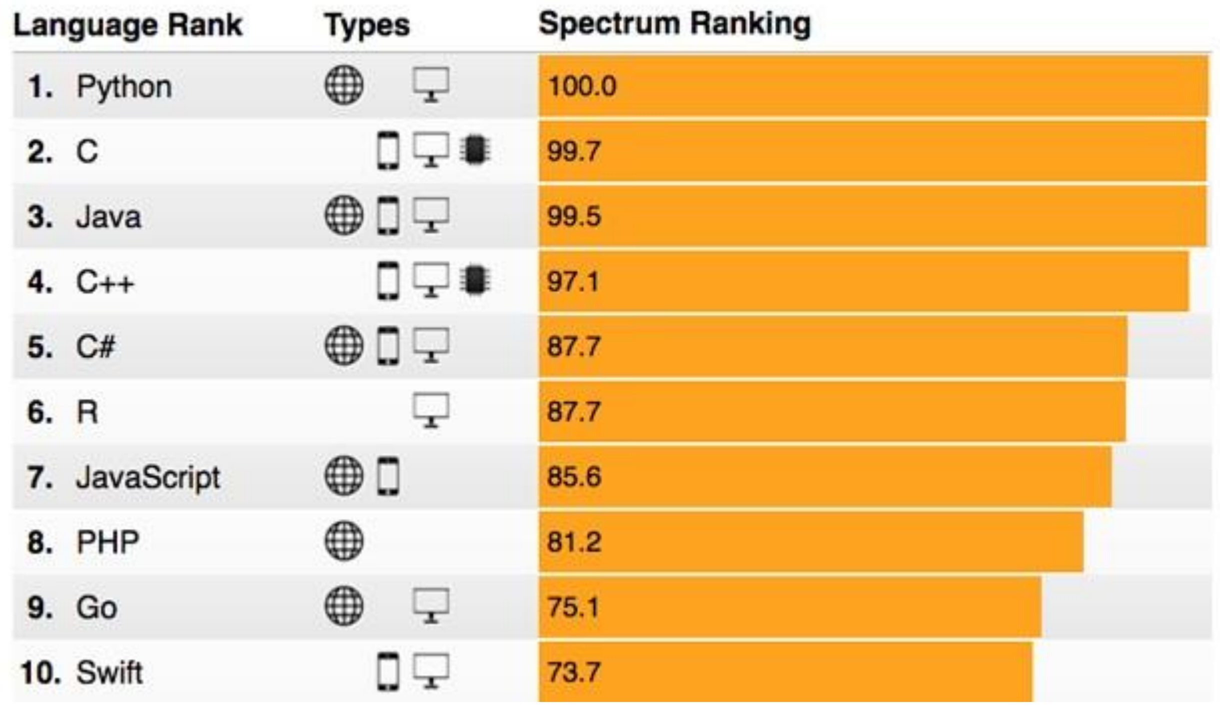
Hello World in x86 machine code



```
File: hello                     ASCII Offset: 0x000000AD / 0x000001BF (%39)
00000000  7F 45 4C 46  01 01 01 00   00 00 00 00  00 00 00 00   .ELF............
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00   ............4...
00000020  F8 00 00 00  00 00 00 00   34 00 20 00  02 00 28 00   ........4. ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08   ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00   ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08   ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  06 00 00 00   ................
00000070  00 10 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  BA 09 00 00  00 B9 A4 90   04 08 BB 01  00 00 00 B8   ................
00000090  04 00 00 00  CD 80 BB 00   00 00 00 B8  01 00 00 00   ................
000000A0  CD 80 00 00  48 69 20 57   6F 72 6C 64  0A 00 54 68   ....Hi World..Th
000000B0  65 20 4E 65  74 77 69 64   65 20 41 73  73 65 6D 62   e Netwide Assemb
000000C0  6C 65 72 20  30 2E 39 39   2E 30 36 2D  32 30 30 37   ler 0.99.06-2007
000000D0  31 31 30 31  00 00 2E 73   68 73 74 72  74 61 62 00   1101...shstrtab.
000000E0  2E 74 65 78  74 00 2E 64   61 74 61 00  2E 63 6F 6D   .text..data..com
000000F0  6D 65 6E 74  00 00 00 00   00 00 00 00  00 00 00 00   ment............
00000100  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000110  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000120  0B 00 00 00  01 00 00 00   06 00 00 00  80 80 04 08   ................
00000130  80 00 00 00  22 00 00 00   00 00 00 00  00 00 00 00   ....."..........
00000140  10 00 00 00  00 00 00 00   11 00 00 00  01 00 00 00   ................
00000150  03 00 00 00  A4 90 04 08   A4 00 00 00  09 00 00 00   ................
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Programming languages (1)

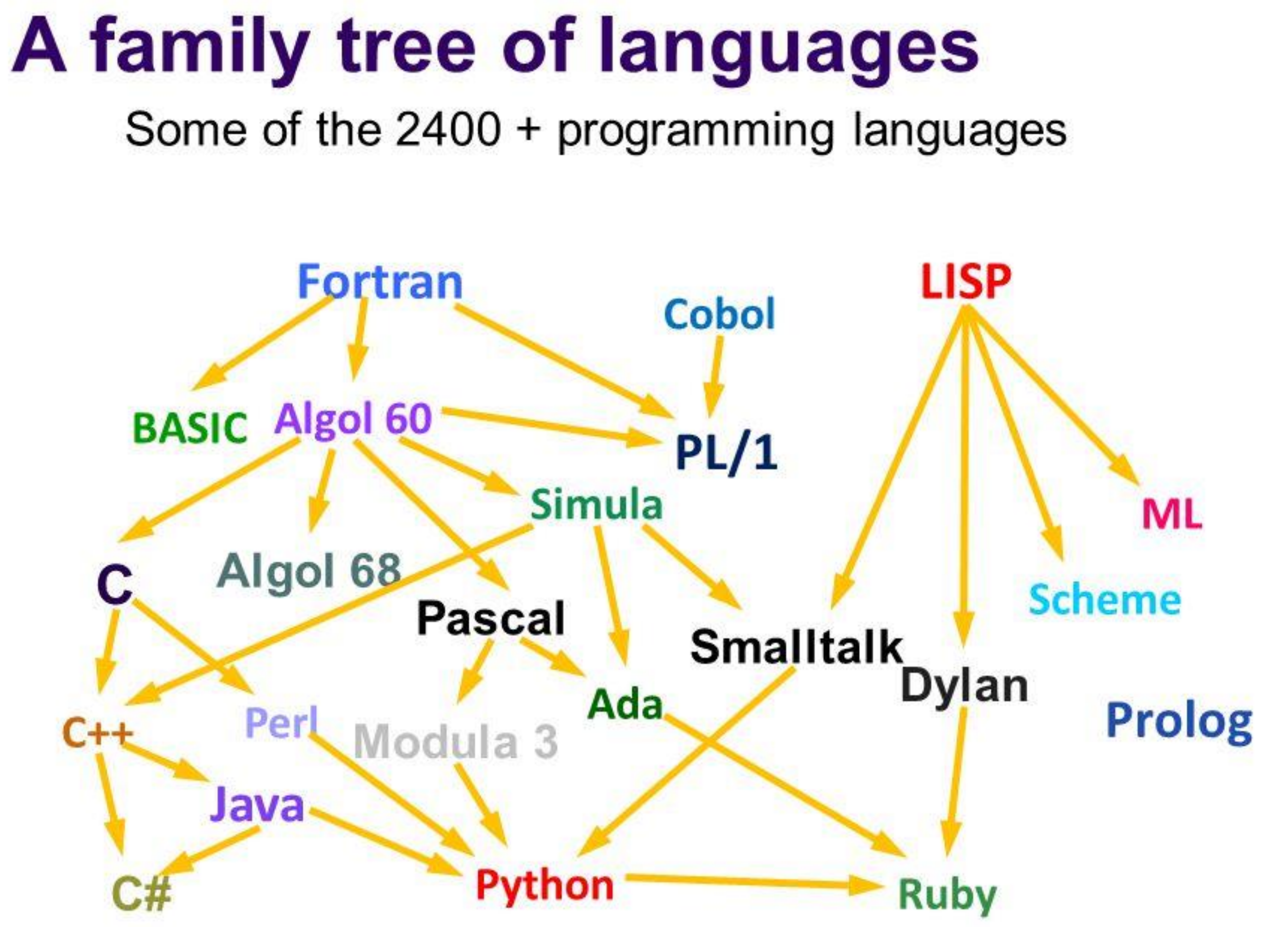
- A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.



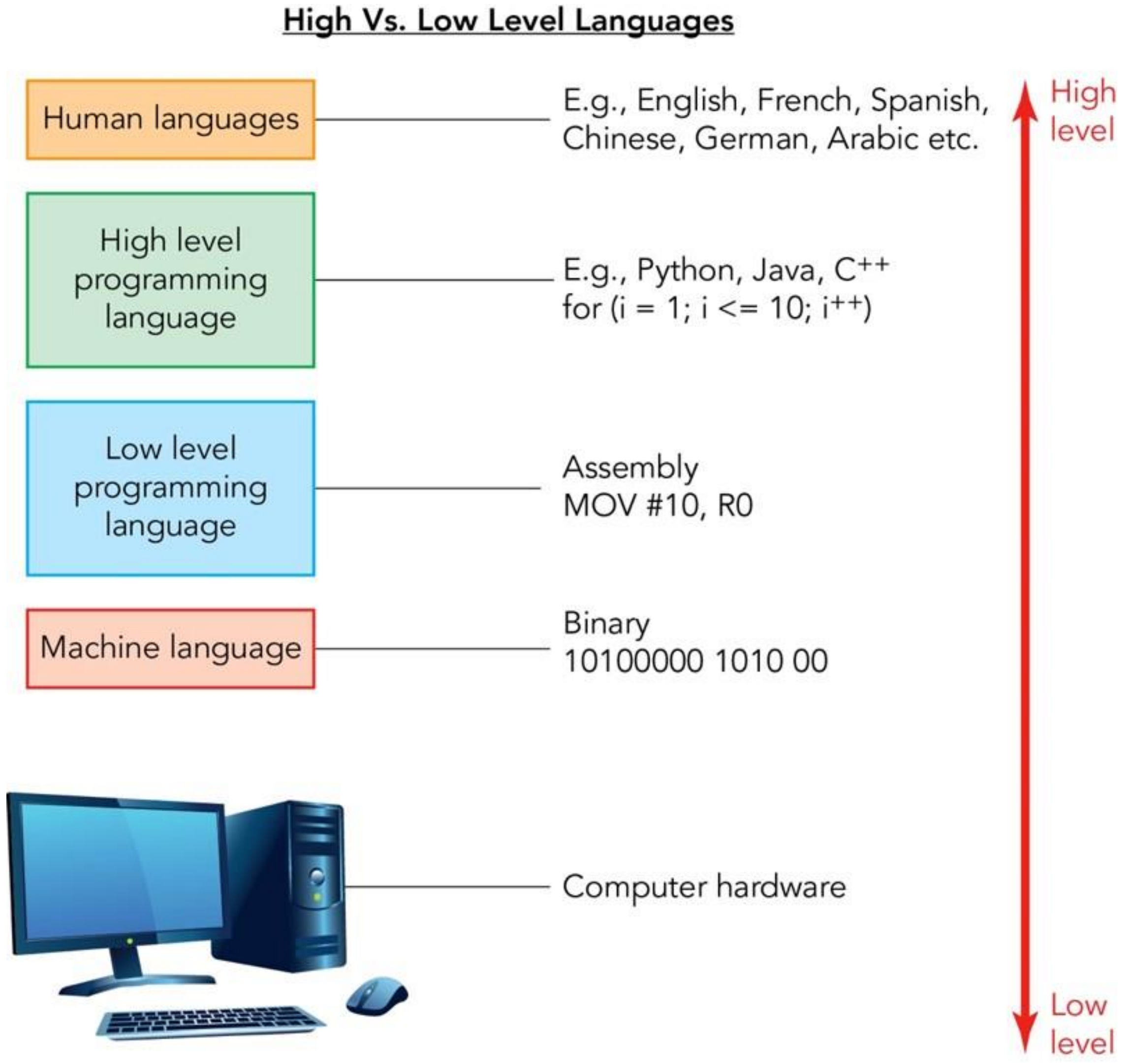
Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ▪	99.7
3. Java	🌐 📱 🖥	99.5
4. C++	📱 🖥 ▪	97.1
5. C#	🌐 📱 🖥	87.7
6. R	🖥	87.7
7. JavaScript	🌐 📱	85.6
8. PHP	🌐	81.2
9. Go	🌐 🖥	75.1
10. Swift	📱 🖥	73.7

IEEE Spectrum Interactive Ranking (2019)

Programming Languages (2)



High-level vs. Low-level programming language


High Vs. Low Level Languages
Human languages
E.g., English, French, Spanish, Chinese, German, Arabic etc.
High level programming language
E.g., Python, Java, C++
for (i = 1; i <= 10; i++)
Low level programming language
Assembly
MOV #10, R0
Machine language
Binary
10100000 1010 00
Computer hardware
High level
Low level

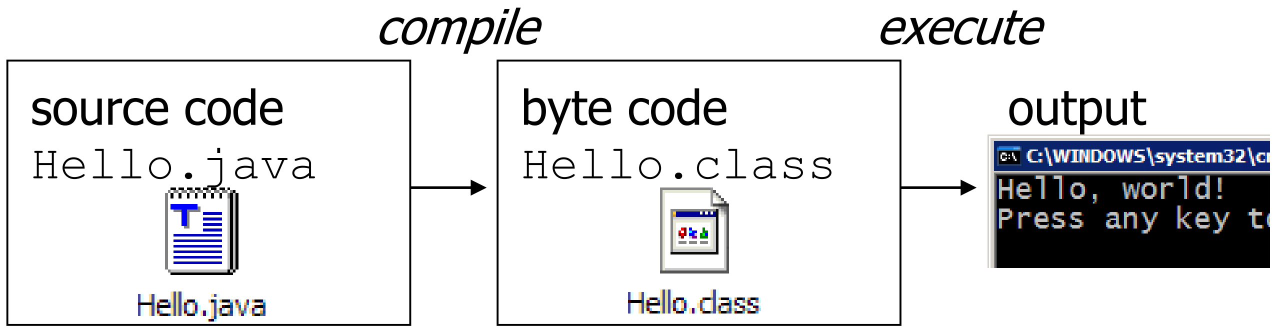
Computer program

- ❑A program is a set of instructions telling the computer what to do
  - ➢**Code** or **source code**: The sequence of instructions in a program

- ❑A program should usually follow strict syntax
  - ➢**Syntax**: the set of legal structures and commands that can be used
  - ➢If the compiler/interpreter does not recognize what you have typed, it will complain until you fix it

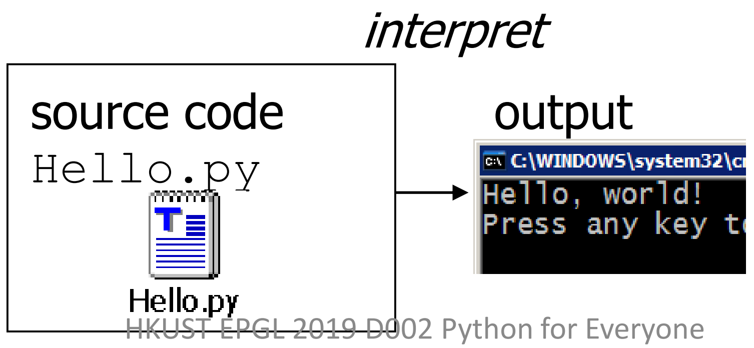
Compiling and interpreting

- Many languages require you to *compile* (translate) your program into machine code, so that the machine understands.



```
Hello, world!
Press any key t
```

- Python is a scripting language, it is instead directly *interpreted* into machine instructions.

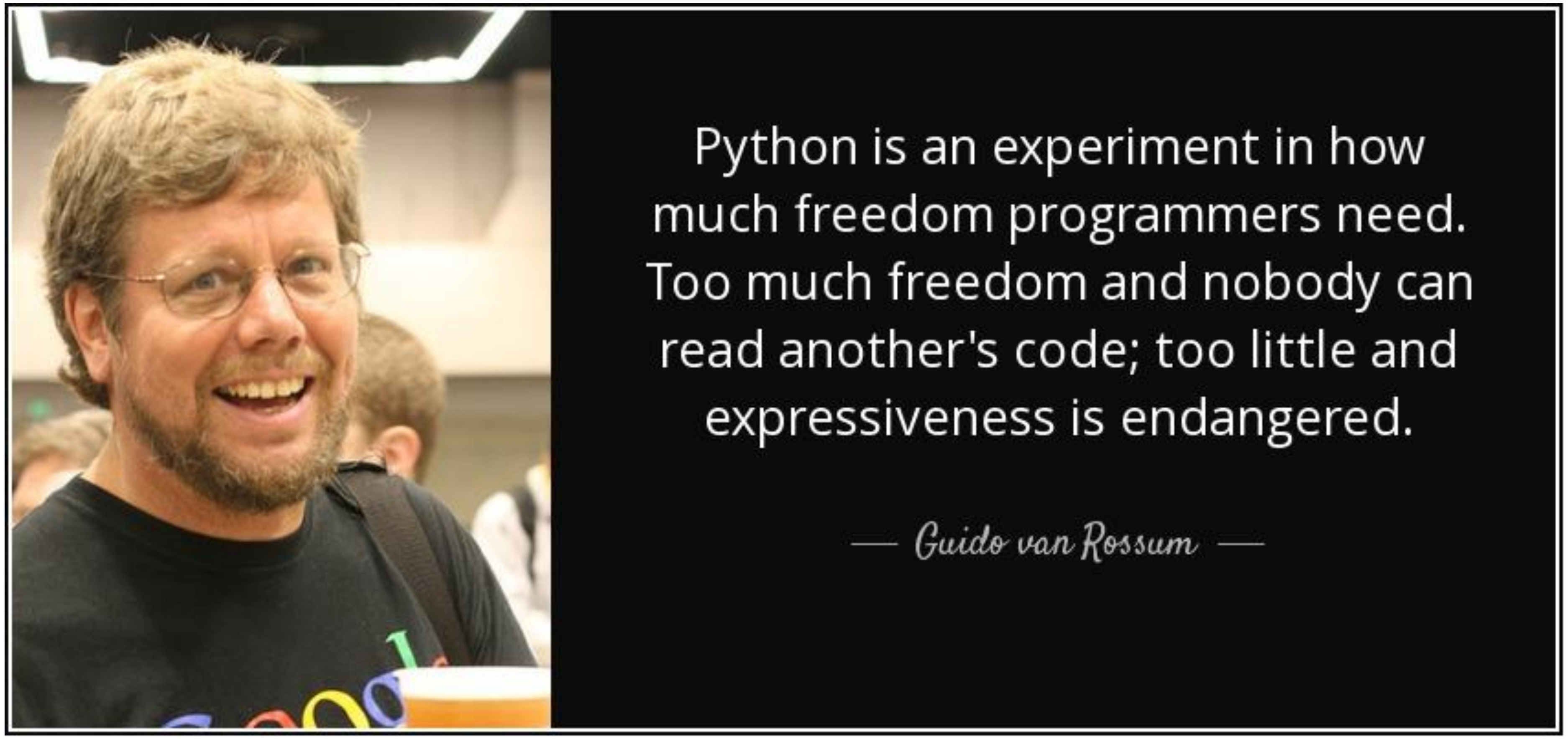


```
Hello, world!
Press any key t
```

Brief history of Python

- ❑Invented in the Netherlands, early 90s by Guido van Rossum
- ❑Named after Monty Python
- ❑Open sourced from the beginning, managed by Python Software Foundation
- ❑Considered a scripting language, but is much more
- ❑Scalable, object oriented and functional from the beginning
- ❑Used by Google from the beginning
- ❑Read more https://docs.python.org/3.7/faq/general.html

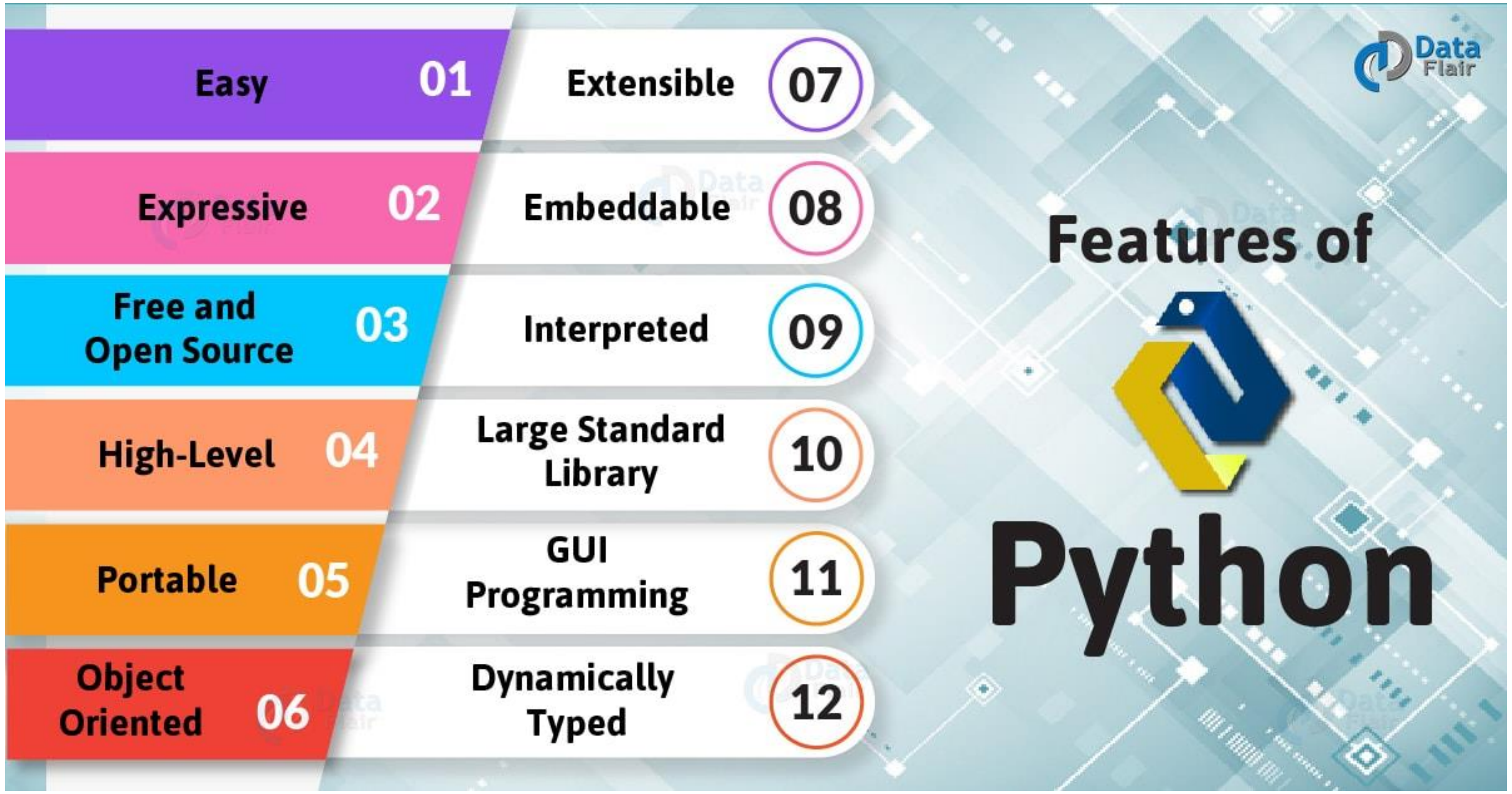
Quote from inventor



Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.

— *Guido van Rossum* —

Python features



Besides,
Python is extremely popular for

- **Data Science (Big Data)**
- **Machine Learning (AI)**

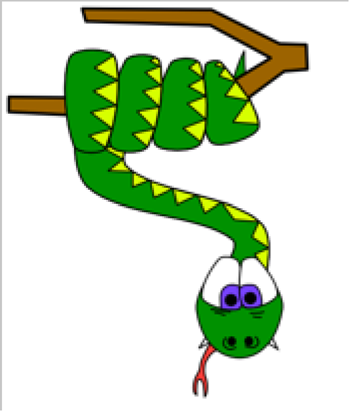
Python drawbacks

- ❑It is an interpreted language
- ❑Might take up more CPU time
- ❑Not suitable to build low level program like operating system (Windows/Android)
- ❑More difficult to manage when the project is really big

Python versions

There are many versions of Python started from 1994 to current date

Python version	Released date
Python 1.0	Jan 1994
Python 2.0	16 Oct 2000
Python 2.7	3 Jul 2010 (major version)
Python 3.0 (Py-3000)	3 Dec 2008 (successful version)
Python 3.7.4	8 Jul 2019



Let’s get into the world of Python

What will be covered in D002

- ❑Interactive "shell"
- ❑Basic types: numbers, strings
- ❑Container types: lists, dictionaries, tuples
- ❑Variables
- ❑Control structures
- ❑Functions & procedures
- ~~❑Classes & instances~~
- ❑Modules & packages
- ~~❑Exceptions~~
- ❑Files & standard library

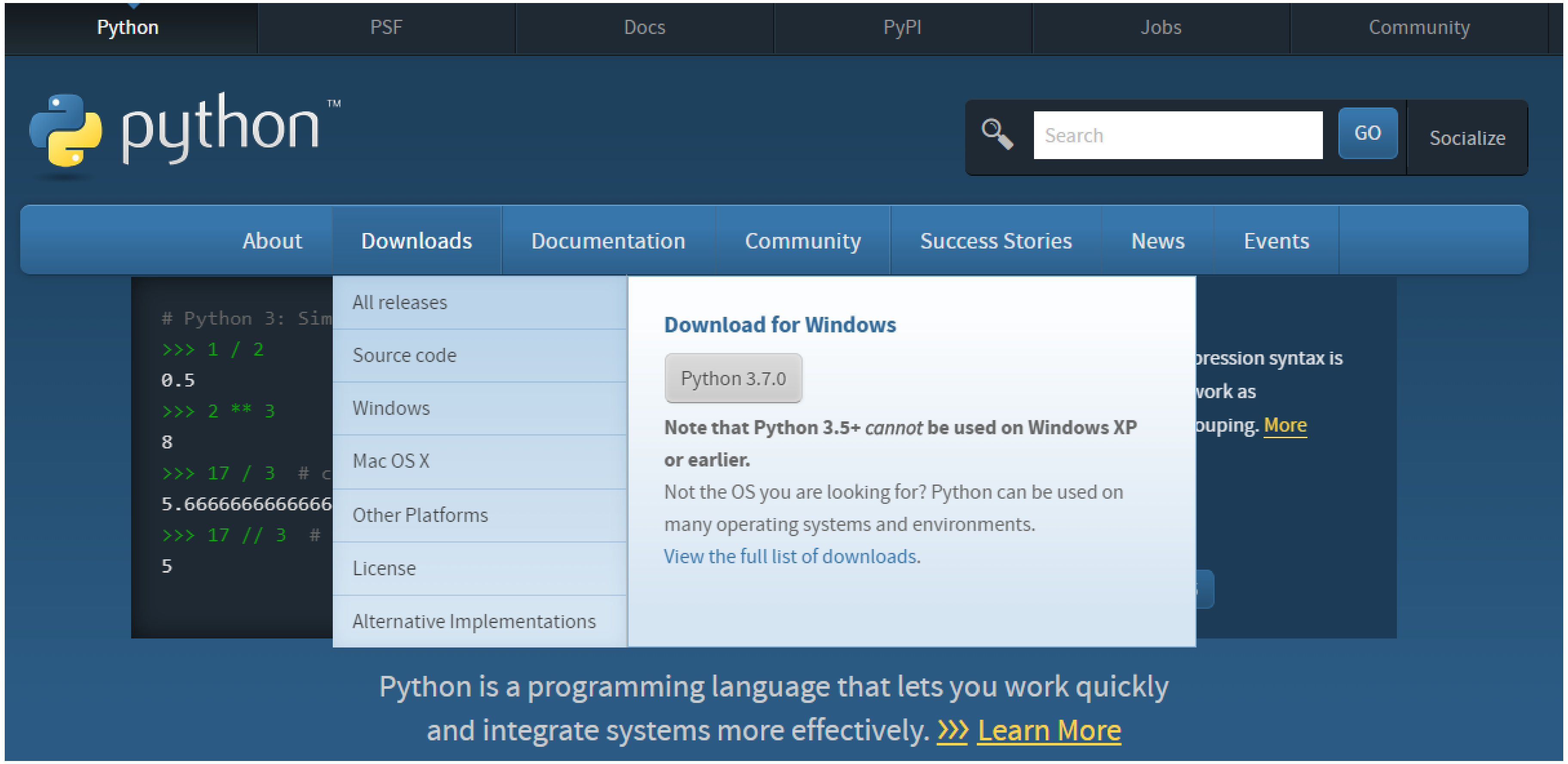
Github

- World leading file repository for programming/collaboration.
- Your programming CV
- It can be very complicate. We use the tiny bits of it.
  - Register an account at www.github.com (not necessary student account)
  - Fork my project at https://www.github.com/khwang0/D002-2019
  - Drag your code to ***YOUR forked repo***.
  - **Commit** it.

- Don't worry, I will show you how

Install Python … at home

- https://www.python.org/



```
# Python 3: Sim
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # c
5.666666666666
>>> 17 // 3  #
5
```

Download for Windows

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

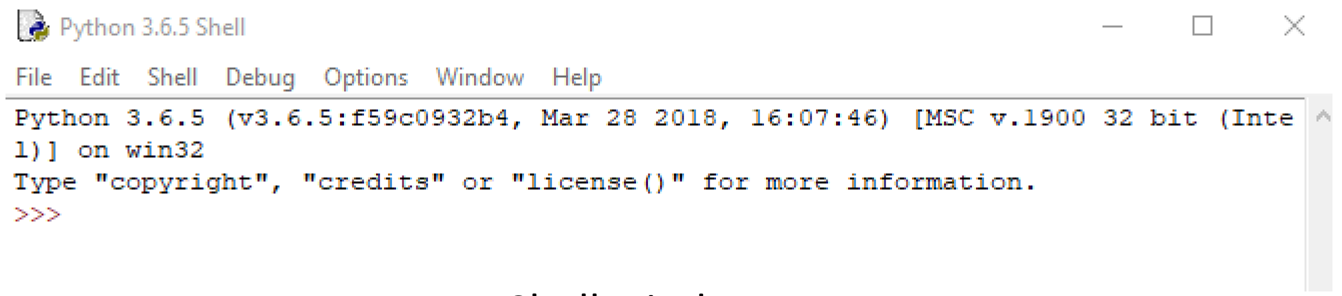
View the full list of downloads.

pression syntax is
work as
ouping. More

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

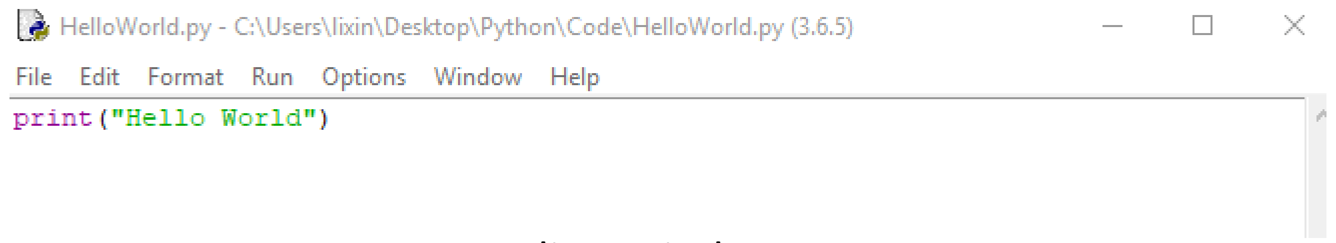
Python IDLE

- ❑Integrated Development and Learning Environment (IDLE)
- ❑two main window types, the Shell window and the Editor window.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

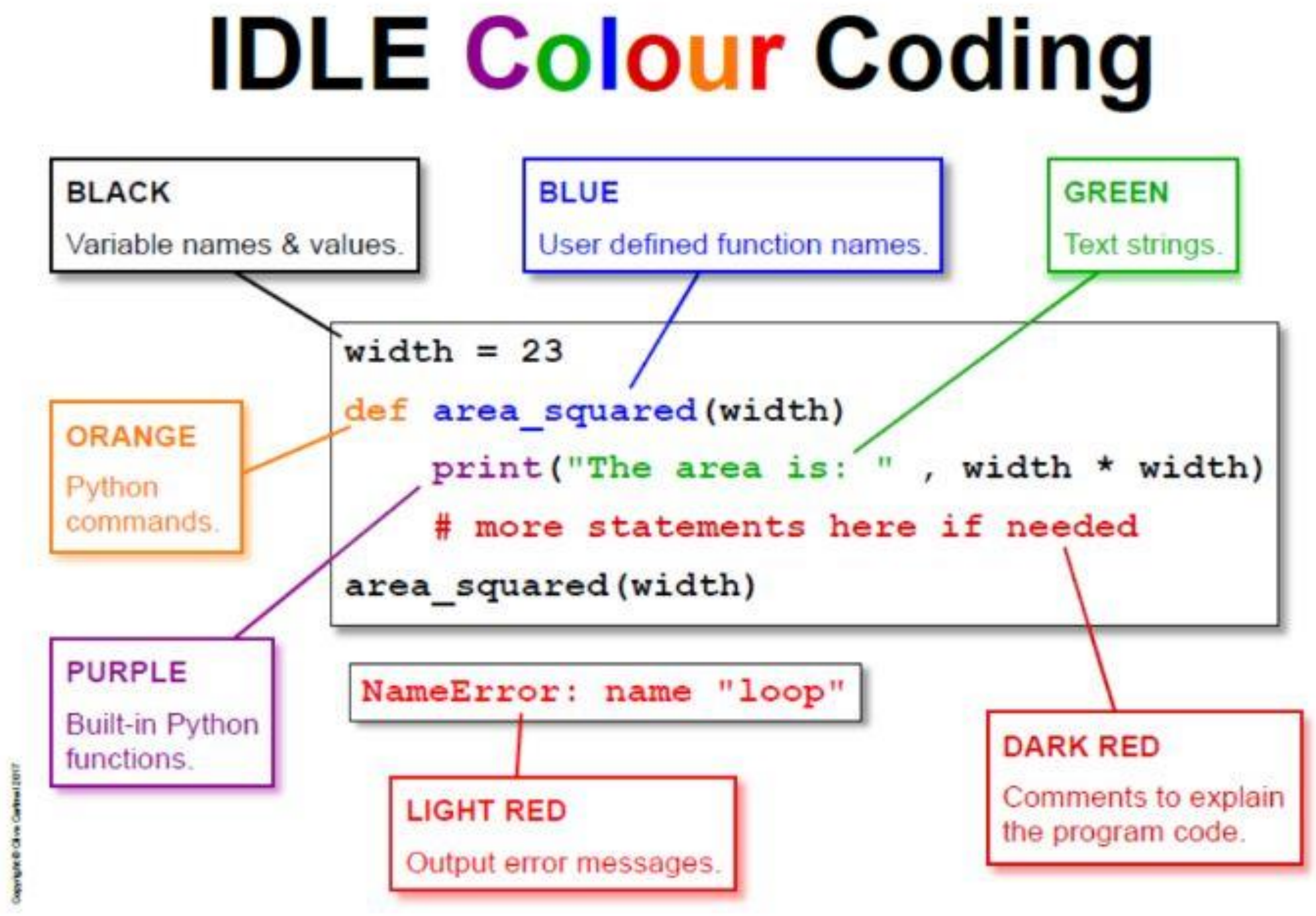
Shell window


HelloWorld.py - C:\Users\lixin\Desktop\Python\Code\HelloWorld.py (3.6.5)
File Edit Format Run Options Window Help

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

Editor window

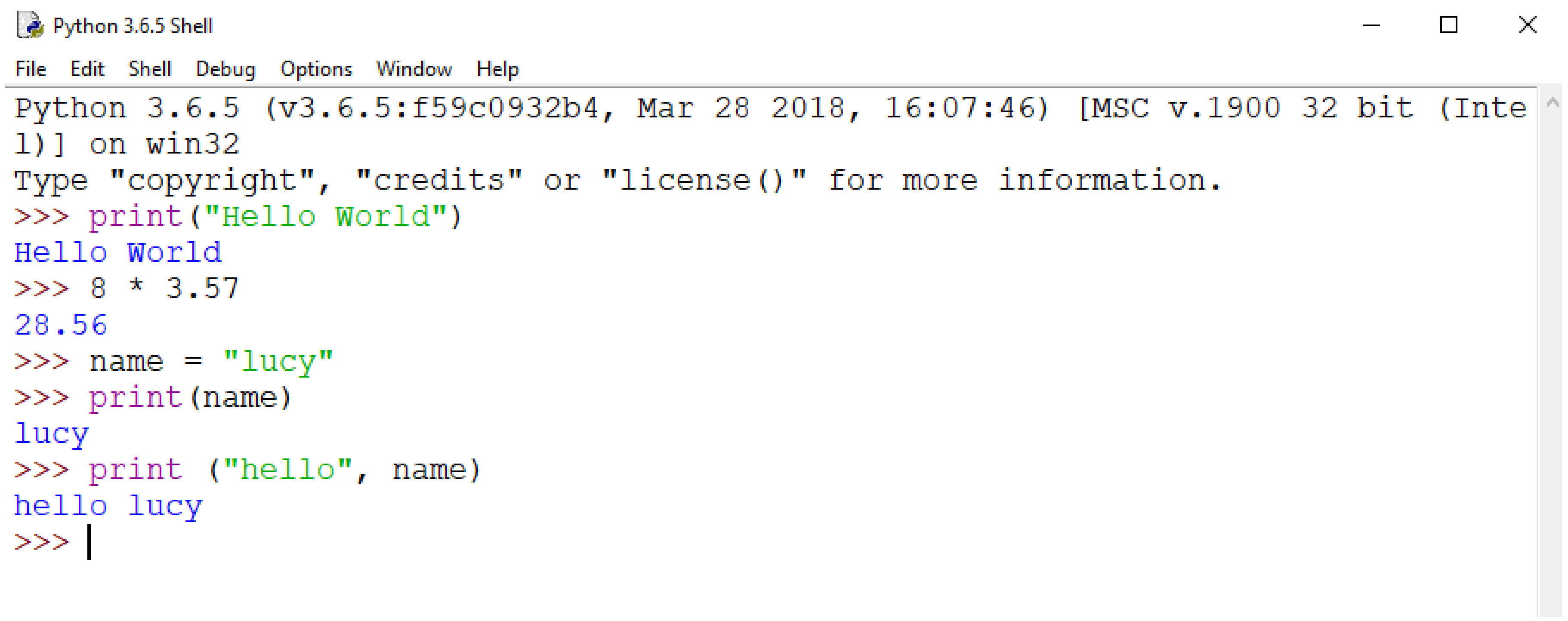
IDLE color coding



```
width = 23
def area_squared(width)
    print("The area is: " , width * width)
    # more statements here if needed
area_squared(width)
```

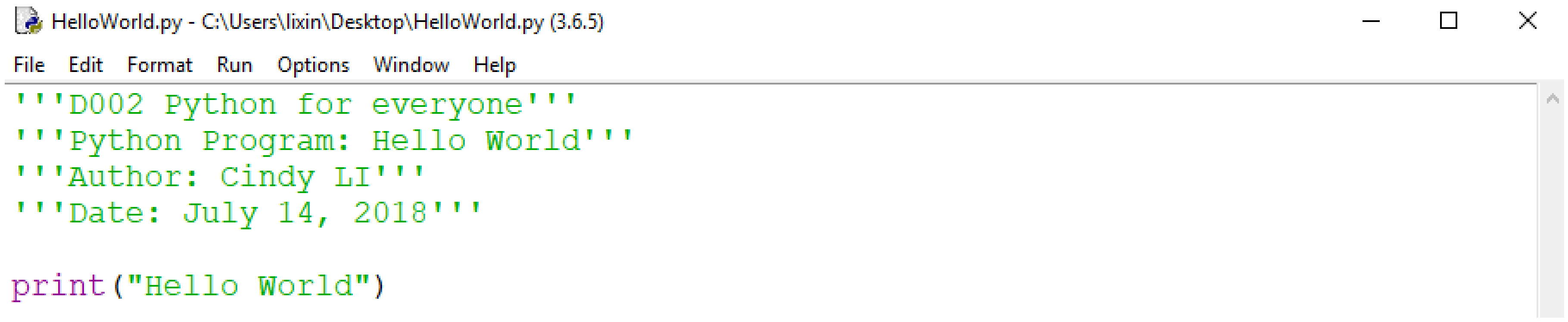
```
NameError: name "loop"
```

Python IDLE shell



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 8 * 3.57
28.56
>>> name = "lucy"
>>> print(name)
lucy
>>> print ("hello", name)
hello lucy
>>> 
```

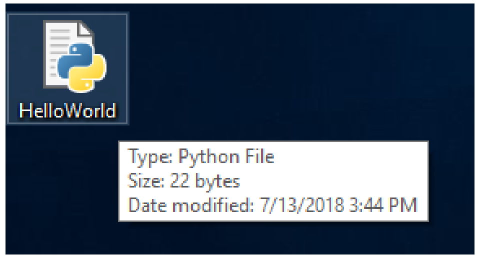
1st Python program – Q1



HelloWorld.py - C:\Users\lixin\Desktop\HelloWorld.py (3.6.5)

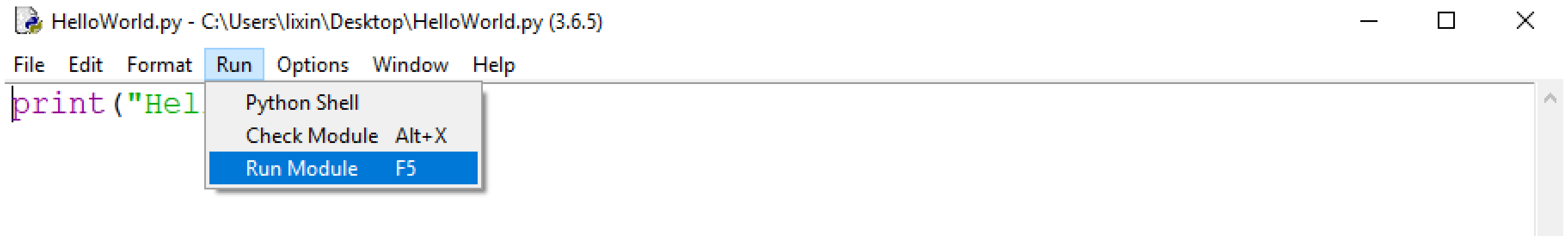
```python
'''D002 Python for everyone'''
'''Python Program: Hello World'''
'''Author: Cindy LI'''
'''Date: July 14, 2018'''

print("Hello World")
```

IDLE Editor



Type: Python File
Size: 22 bytes
Date modified: 7/13/2018 3:44 PM

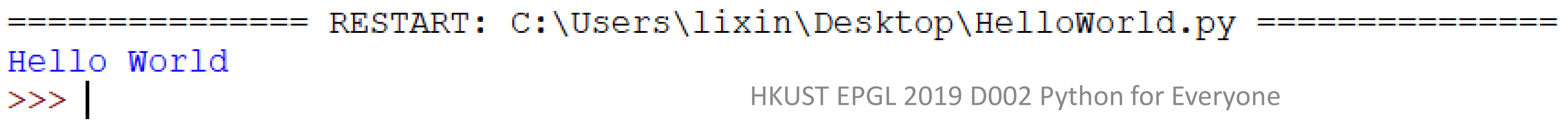
Python script created as a .py file



HelloWorld.py - C:\Users\lixin\Desktop\HelloWorld.py (3.6.5)

```python
print("Hel
```

Run the script



```
=============== RESTART: C:\Users\lixin\Desktop\HelloWorld.py ===============
Hello World
>>> 
```

Execution result

For more information?

http://python.org/

- documentation, tutorials, beginners guide, core distribution, ...

Python for kids

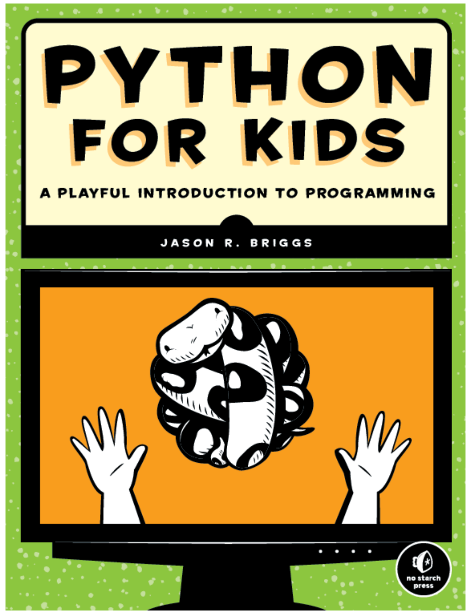
- Book used in D002

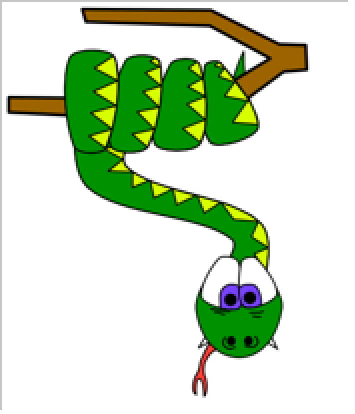
https://www.learnpython.org/

https://www.w3schools.com/python/python_getstarted.asp

https://www.codecademy.com/learn/learn-python

- Online learning materials





Python Basics

Arithmetic Operation

Variable

String

Input and Output

Branch

Arithmetic operation

- **expression**: A data value or set of operations to compute a value.

  Examples:`1 + 4 * 3`

- Arithmetic operators we will use:

`+ - * /`	addition, subtraction/negation, multiplication, division
`%`	modulus, a.k.a. remainder
`**`	exponentiation

- **precedence**: order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

    `1 + 3 * 4` is `13`

  - Parentheses can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

Integer division

- When we divide integers with / , the quotient is also an integer.

```
        3               52
  4 ) 14          27 ) 1425
        12               135
         2                 75
                           54
                           21
```

- More examples:
  - 35 / 5 is 7
  - 84 / 10 is 8
  - 156 / 100 is 1

- The % operator computes the remainder from a division of integers.

```
        3               43
  4 ) 14          5 ) 218
        12              20
         2               18
                         15
                          3
```

Real numbers

Real numbers:
Number with
decimal place

- Python can also manipulate real numbers.
  - Examples: `6.022`   `-15.9997`   `42.0`   `2.143e17`

- The operators `+ - * / % ** ( )` all work for real numbers.
  - The `/` produces an exact answer: `15.0/2.0` is `7.5`
  - The same rules of precedence also apply to real numbers:
    Evaluate `( )` before `* / %` before `+ -`

- When integers and reals are mixed, the result is a real number.
  - Example: `1/2.0` is `0.5`
  - The conversion occurs on a per-operator basis.

```
7 / 3 * 1.2 + 3 / 2
  2   * 1.2 + 3 / 2
    2.4     + 3 / 2
    2.4     +   1
         3.4
```

Math commands

❑Python has useful commands for performing calculations.

Command name	**Description**
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	**Description**
`e`	2.7182818...
`pi`	3.1415926...

❑To use many of these commands, you must write the following at the top of your Python program (will explain in detail in following lectures)

```
from math import *
```

Practice – Q2

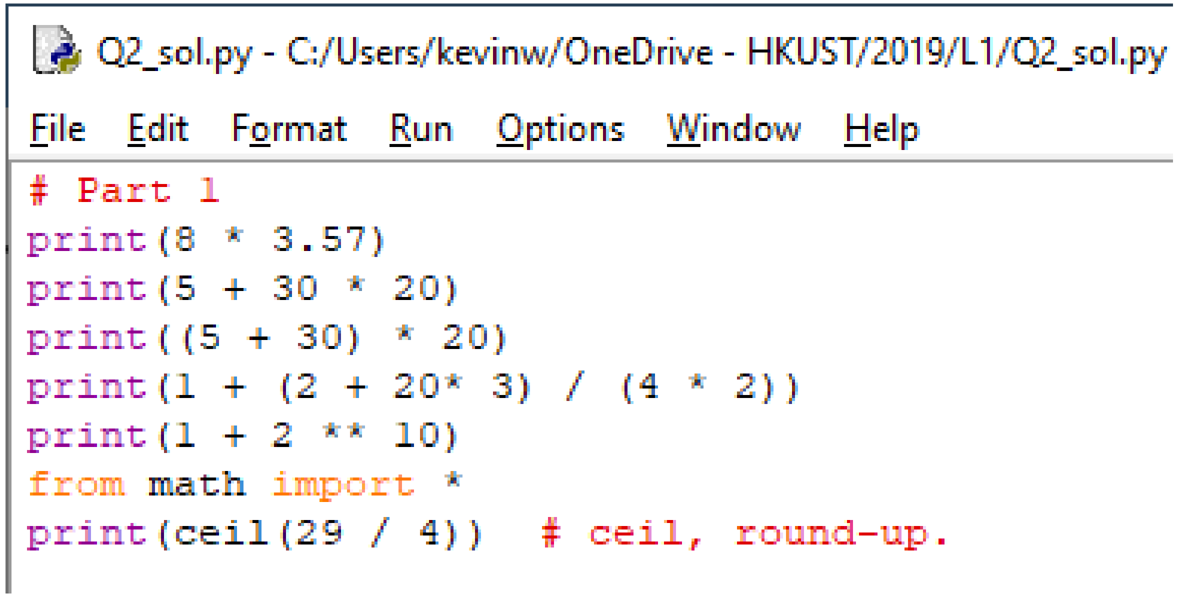
- Calculate the following with the Python shell

```
8 x 3.57

5 + 30 x 20

(5 + 30) x 20
```

$$1 + \frac{2 + 20 \times 3}{4 \times 2}$$

$1+2^{10}$

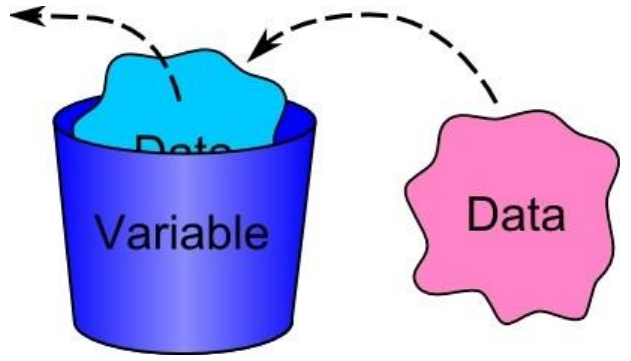
```
Find how many 4-seats taxi are needed to take all of us to a field trip.
```


Q2_sol.py - C:/Users/kevinw/OneDrive - HKUST/2019/L1/Q2_sol.py
File Edit Format Run Options Window Help

```
# Part 1
print(8 * 3.57)
print(5 + 30 * 20)
print((5 + 30) * 20)
print(1 + (2 + 20* 3) / (4 * 2))
print(1 + 2 ** 10)
from math import *
print(ceil(29 / 4))  # ceil, round-up.
```

Variable

- **variable**: A named piece of memory that can store a value.
  - Usage:
    - Compute an expression's result,
    - store that result into a variable,
    - and use that variable later in the program.



- **assignment statement**: stores a value into a variable.
  - Syntax:

    name = ***value***
  - A variable that has been given a value can be used in expressions.

Naming rules

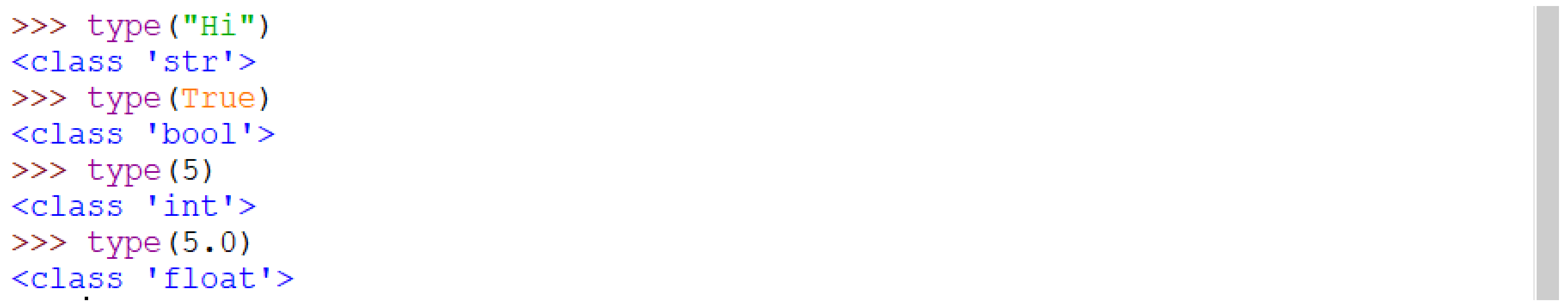
- Variables names must start with a letter or an underscore, such as:
  - `_underscore`
  - `underscore_`
- The remainder of the variable name may consist of letters, numbers and underscores
  - `password1`
  - `n00b`
  - `un_der_scores`
- Names are case sensitive
  - `case_sensitive, CASE_SENSITIVE, and Case_Sensitive are each a different variable`

Naming conventions

- ❑ Readability is very important. Which of the following is easiest to read?
  - ➢ python_puppet
  - ➢ pythonpuppet
  - ➢ pythonPuppet
- ❑ Descriptive names are very useful. If you are writing a program that adds up all of the bad puns made, which do you think is the better variable name?
  - ➢ total_bad_puns
  - ➢ super_bad

Python simple data type (1)

- In Python, all data has an associated data **“Type”**
- You can find the “Type” of any piece of data by using the `type()`



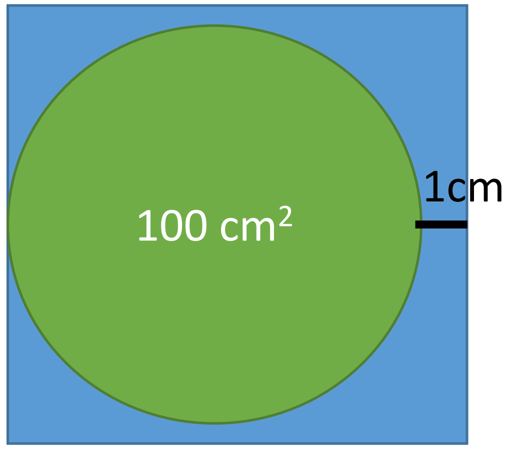
```
>>> type("Hi")
<class 'str'>
>>> type(True)
<class 'bool'>
>>> type(5)
<class 'int'>
>>> type(5.0)
<class 'float'>
```

Python simple data type (2)

- Numbers
  - int – Integer: -5, 10, 77
  - float – Floating Point numbers: 3.1457, 0.34 (with fractional part)
- bool – Booleans (`True` or `False`)
- Strings are a more complicated data type. They are made up of individual letters (strings of length 1). (You will see string soon)

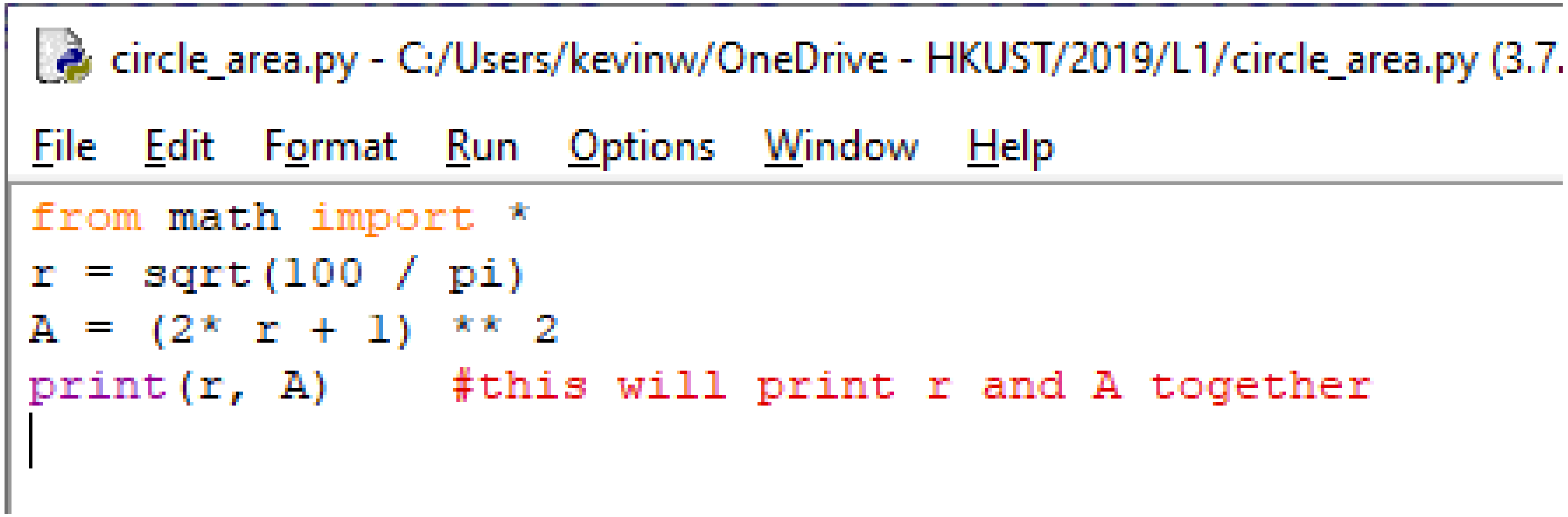
Using Python Variables – Q2 con’t

Try to calculate the area of the square, using python



$$r = \sqrt{100/\pi}$$

$$A = (2r + 1)^2$$

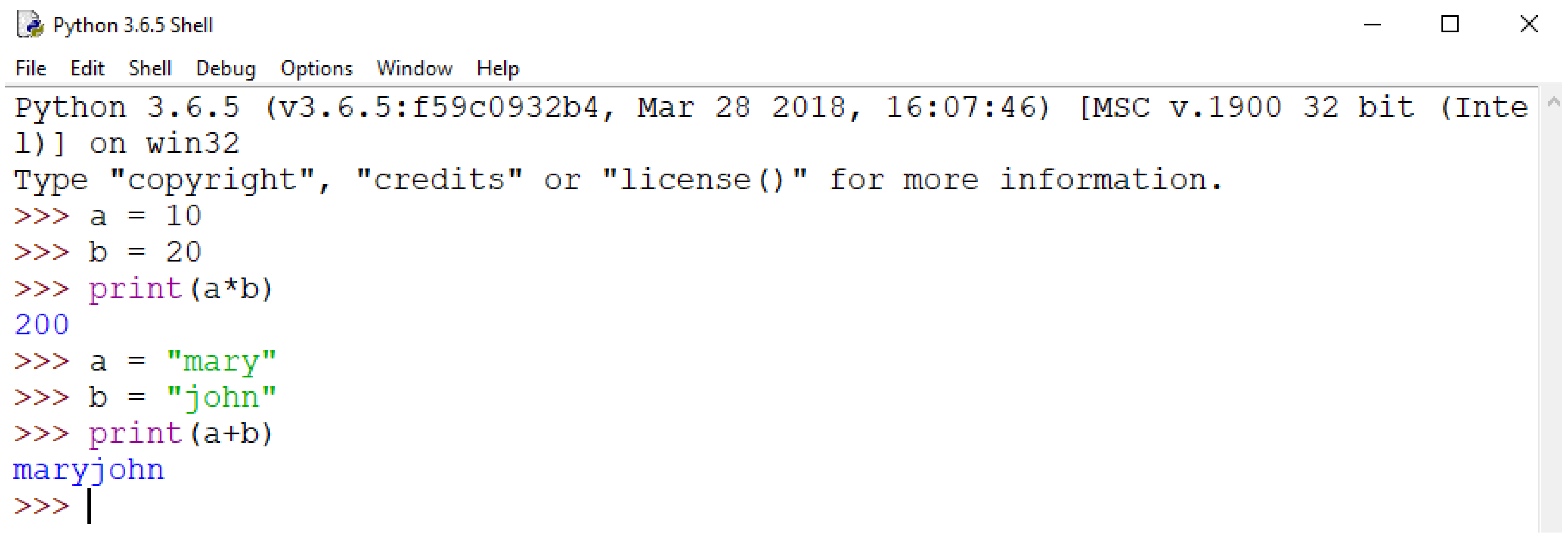

circle_area.py - C:/Users/kevinw/OneDrive - HKUST/2019/L1/circle_area.py (3.7.
File Edit Format Run Options Window Help

```
from math import *
r = sqrt(100 / pi)
A = (2* r + 1) ** 2
print(r, A)     #this will print r and A together
```

Would it work if we swap line 3 and 4?

Python variable

- Python variable is not "statically typed"
  - You can change the type of variables anytime.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = 10
>>> b = 20
>>> print(a*b)
200
>>> a = "mary"
>>> b = "john"
>>> print(a+b)
maryjohn
>>> 
```

Python string

❑A **string** is a sequence of letters (called **characters**)

❑Create string is done simply by enclosing characters in quotes

❑Python treats single quotes the same as double quotes

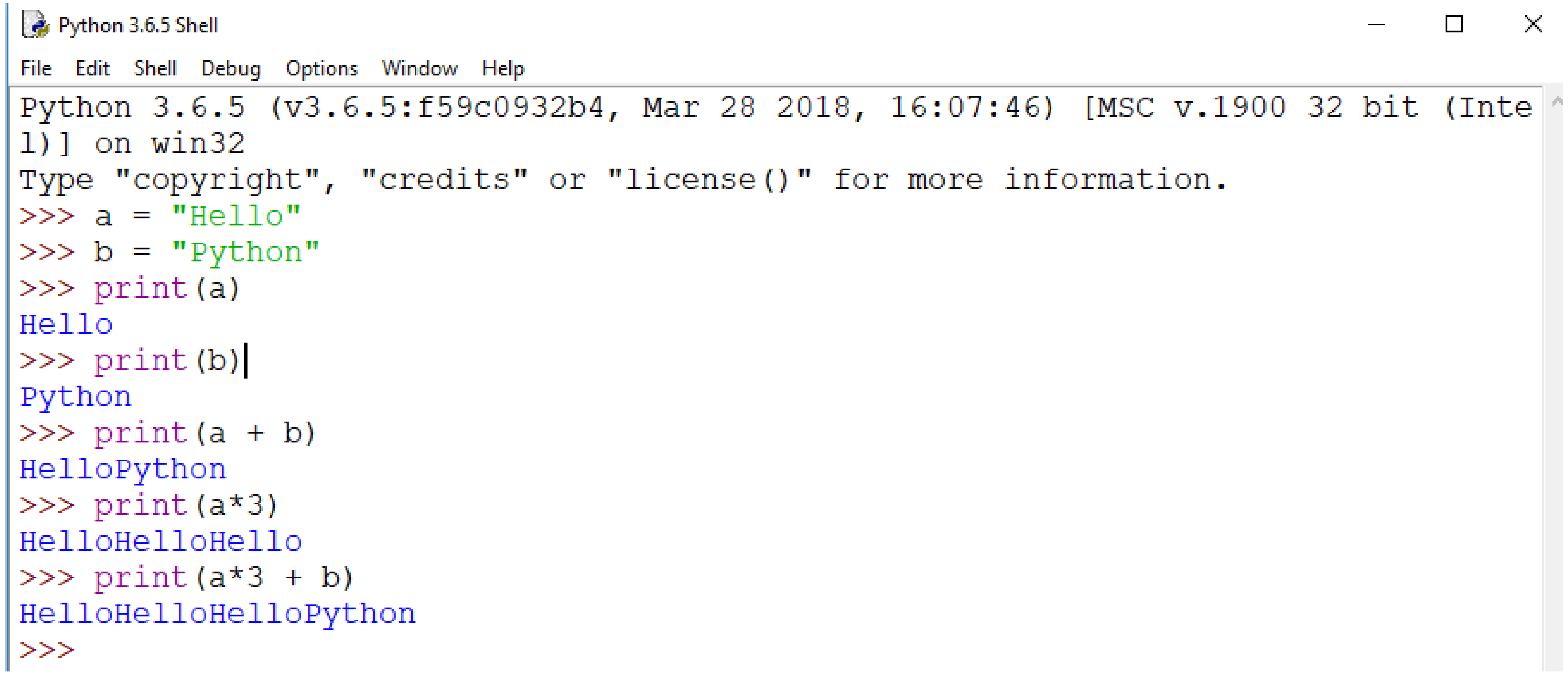
```
str1 = 'Hello World!'
str2 = "Python Programming"
```

❑Use triple quotes for multi-line string

```
str3 = '''Summer D002 teaches me Python.
         I enjoy the experience!'''
```

String operations

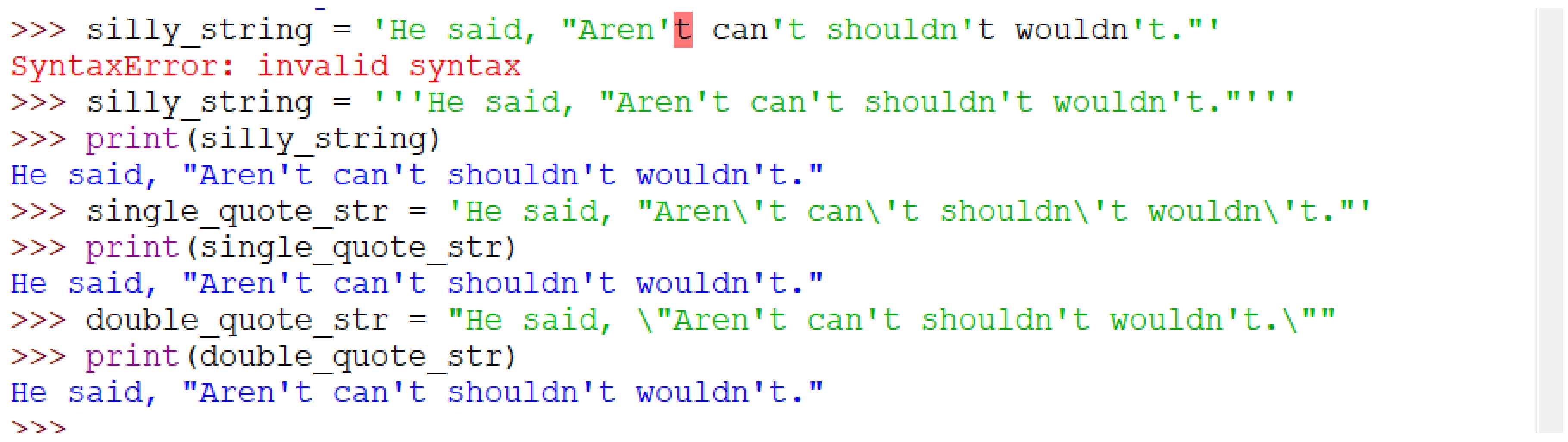
- + string concatenation
- * string repetition



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = "Hello"
>>> b = "Python"
>>> print(a)
Hello
>>> print(b)
Python
>>> print(a + b)
HelloPython
>>> print(a*3)
HelloHelloHello
>>> print(a*3 + b)
HelloHelloHelloPython
>>>
```

String with quotes

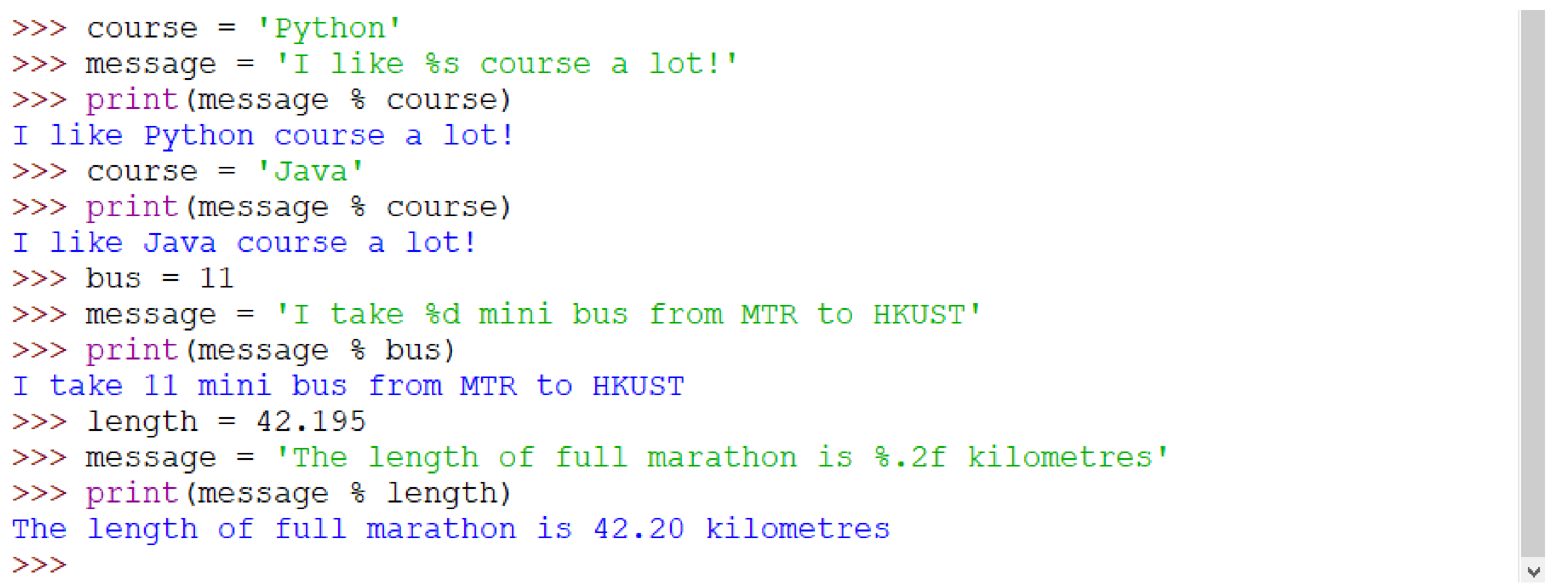
❑Solution: multi-line string, escape with \



```
>>> silly_string = 'He said, "Aren't can't shouldn't wouldn't."'
SyntaxError: invalid syntax
>>> silly_string = '''He said, "Aren't can't shouldn't wouldn't."'''
>>> print(silly_string)
He said, "Aren't can't shouldn't wouldn't."
>>> single_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'
>>> print(single_quote_str)
He said, "Aren't can't shouldn't wouldn't."
>>> double_quote_str = "He said, \"Aren't can't shouldn't wouldn't.\""
>>> print(double_quote_str)
He said, "Aren't can't shouldn't wouldn't."
>>>
```

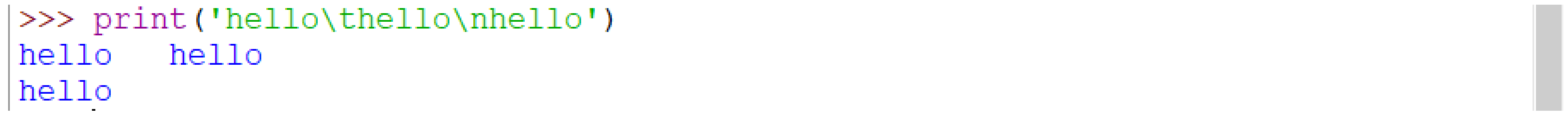
String formatting

- Values can be embedded to string using %
- %d integer, %s string, %f real number



```
>>> course = 'Python'
>>> message = 'I like %s course a lot!'
>>> print(message % course)
I like Python course a lot!
>>> course = 'Java'
>>> print(message % course)
I like Java course a lot!
>>> bus = 11
>>> message = 'I take %d mini bus from MTR to HKUST'
>>> print(message % bus)
I take 11 mini bus from MTR to HKUST
>>> length = 42.195
>>> message = 'The length of full marathon is %.2f kilometres'
>>> print(message % length)
The length of full marathon is 42.20 kilometres
>>>
```

- \t for tab, \n for Enter



```
>>> print('hello\thello\nhello')
hello	hello
hello
```

Input and Output (1)

❑ `print` : Produces text output on the shell

`print ("`***Message***`")`

`print (`***Expression***`)`

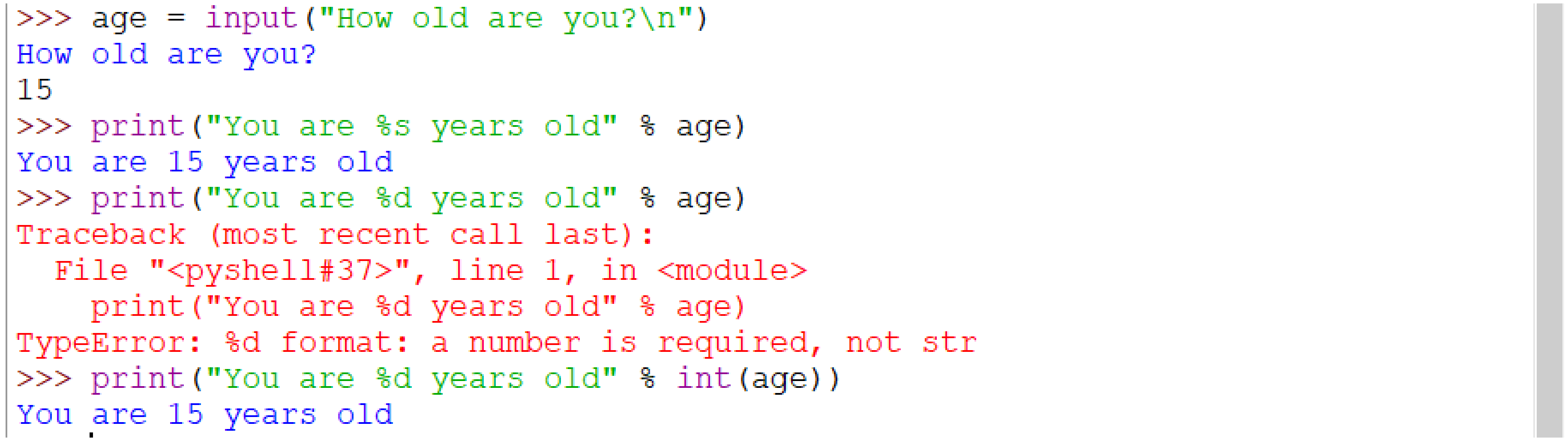
- Prints the given text message or expression value on the console, and moves the cursor down to the next line.

`print (`***Item1*** `,` ***Item2*** `,` … `,` ***ItemN***`)`

- Prints several messages and/or expressions on the same line.

Input and Output (2)

- `input` : Reads user input
  - You can assign (store) the result of `input` into a variable.



```
>>> age = input("How old are you?\n")
How old are you?
15
>>> print("You are %s years old" % age)
You are 15 years old
>>> print("You are %d years old" % age)
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    print("You are %d years old" % age)
TypeError: %d format: a number is required, not str
>>> print("You are %d years old" % int(age))
You are 15 years old
```

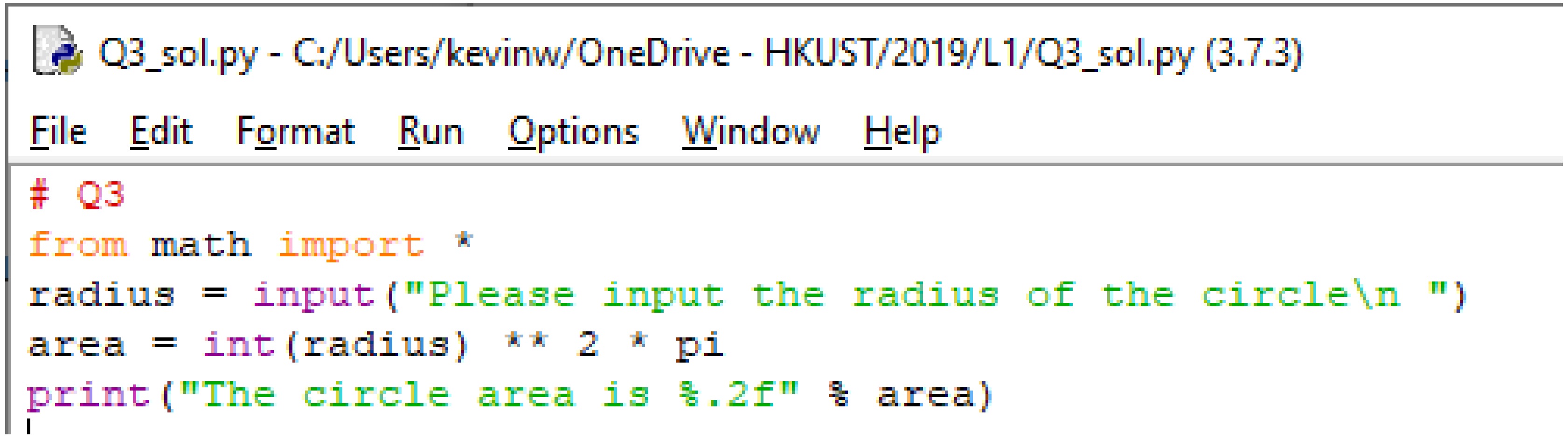
If I want to lie about my age, can I say …

`print(“I am %d-2 years old” % int(age))`

Should be
`print(“I am %d years old” % int(age) – 2)`

Practice: circle area calculator – Q3

- ❑Input: radius from user keyboard input
- ❑Output: circle area

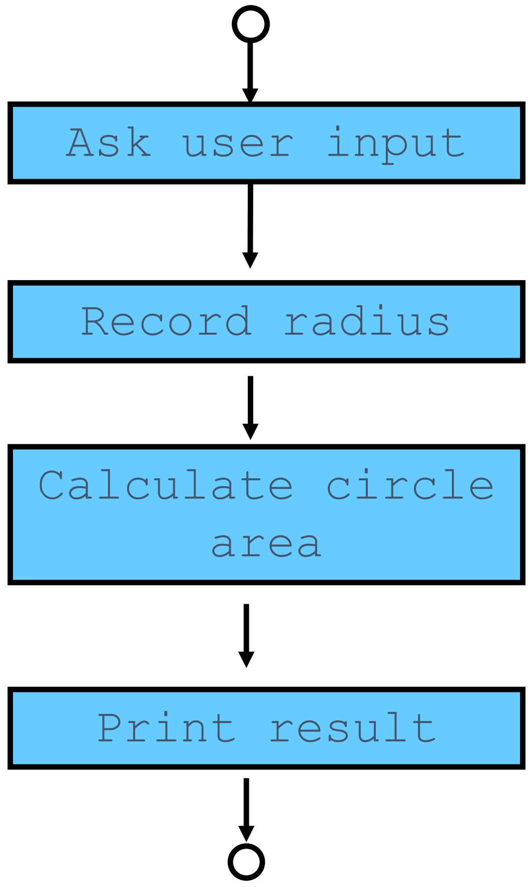

Q3_sol.py - C:/Users/kevinw/OneDrive - HKUST/2019/L1/Q3_sol.py (3.7.3)
File Edit Format Run Options Window Help

```
# Q3
from math import *
radius = input("Please input the radius of the circle\n ")
area = int(radius) ** 2 * pi
print("The circle area is %.2f" % area)
```

Keep 2 decimal places

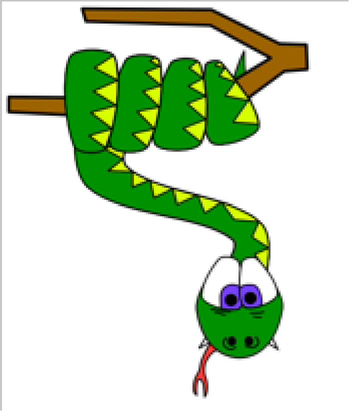
```
Please input the radius of the circle
20
The circle area is 1256.64
```

Sequence Control Structure



Control Structures

- 3 control structures
  - Sequential structure
    - Built into Python
  - Selection structure
    - The **`if`** statement
    - The **`if/else`** statement
    - The **`if/elif/else`** statement
  - Repetition structure
    - The **`while`** repetition structure
    - The **`for`** repetition structure



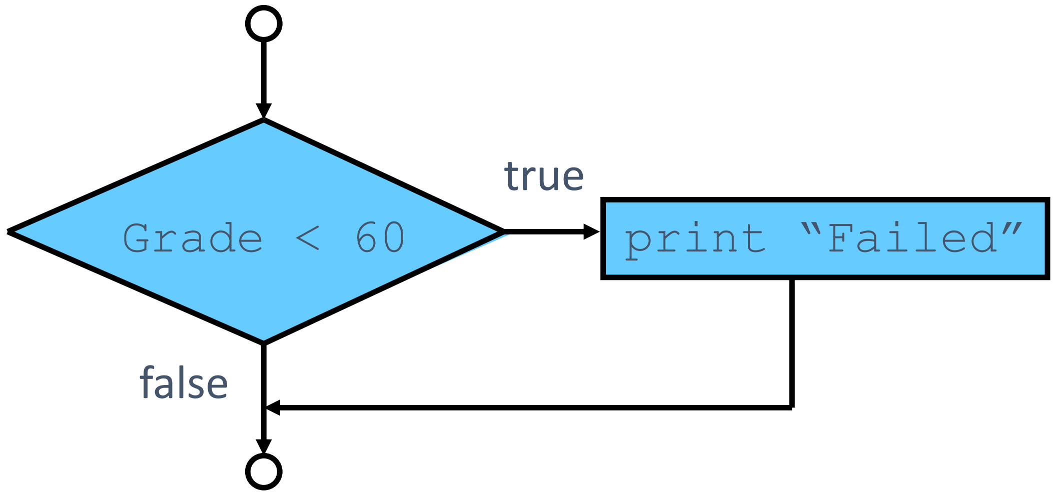
Branch to Make Decisions

Changing Output

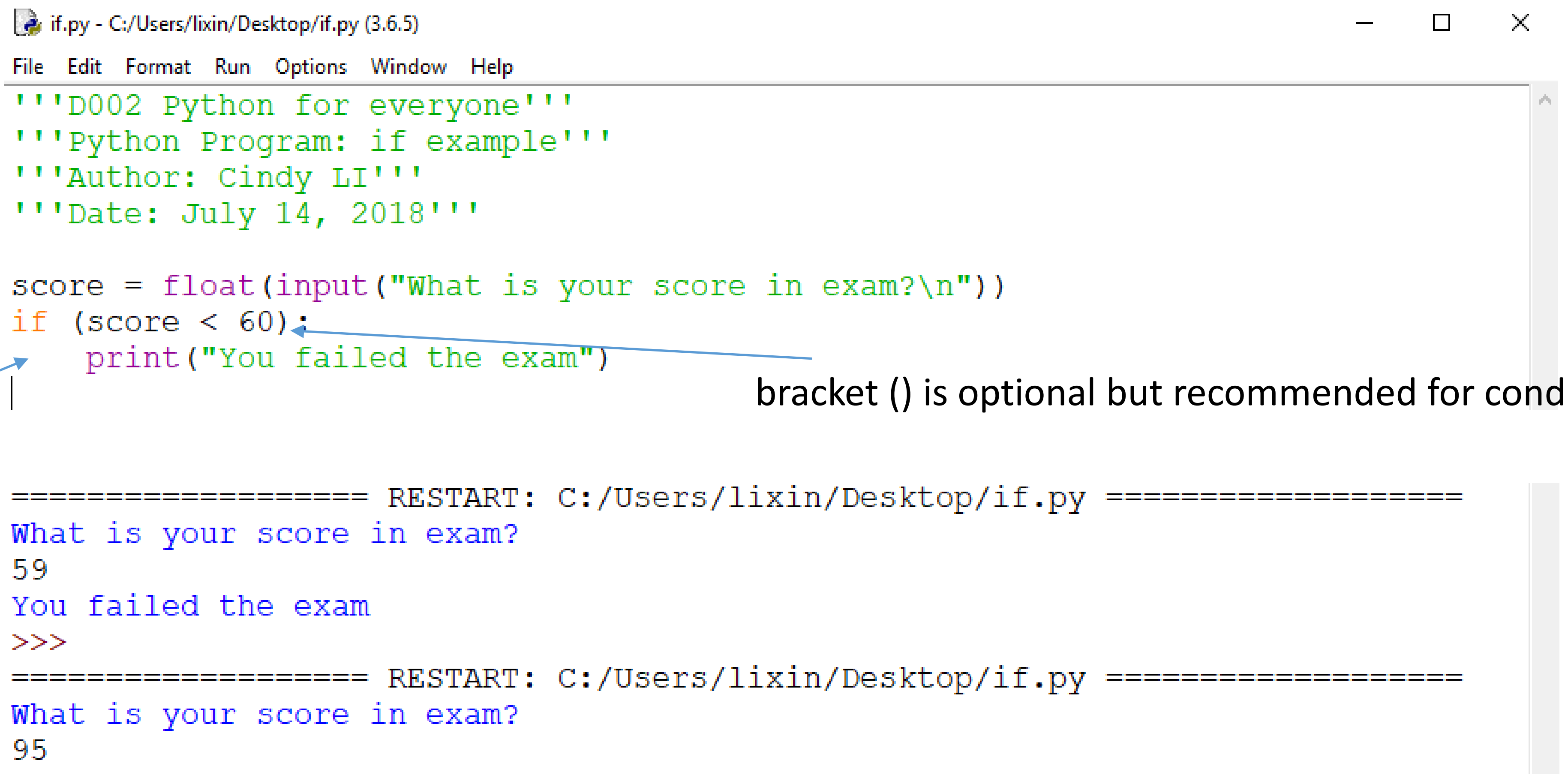
- The output of an algorithm often depends on conditions that occur when the program runs, so it may vary if the program runs more than once.

```
if (Typhoon Signal 8 is hoisted)
then
    sleep at home
else
    go to work/school
```

`if` Structure



if example



if.py - C:/Users/lixin/Desktop/if.py (3.6.5)
File Edit Format Run Options Window Help

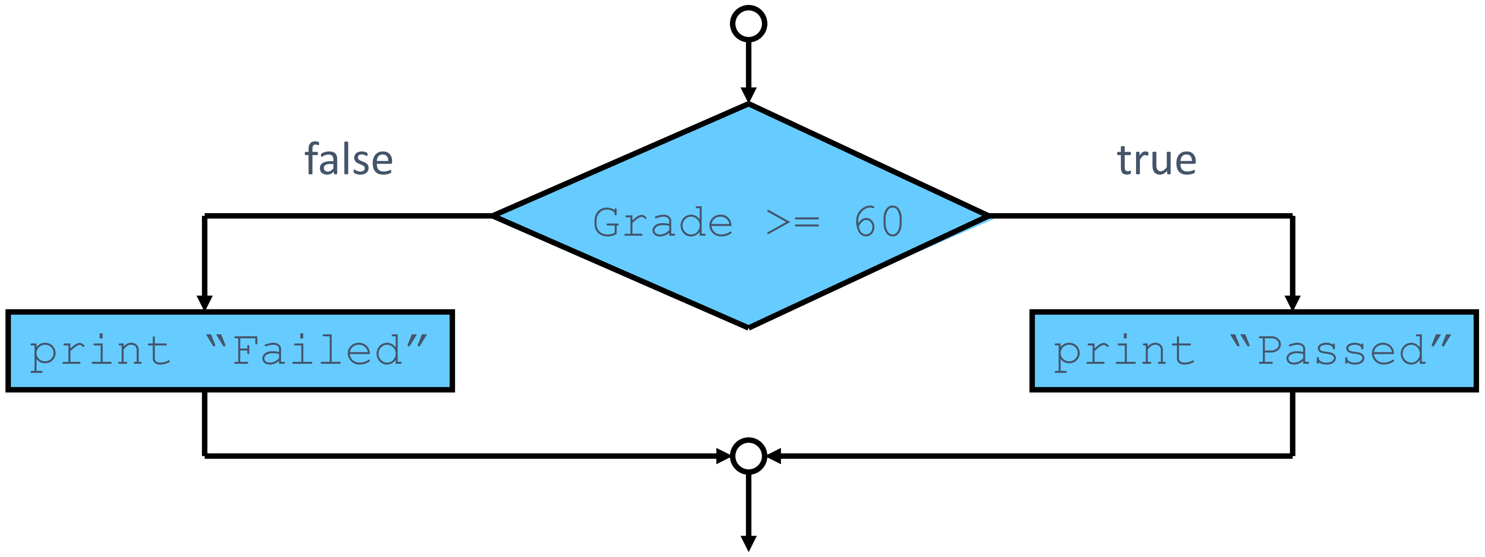
```
'''D002 Python for everyone'''
'''Python Program: if example'''
'''Author: Cindy LI'''
'''Date: July 14, 2018'''

score = float(input("What is your score in exam?\n"))
if (score < 60):
    print("You failed the exam")
```

Press a “tab” key here.

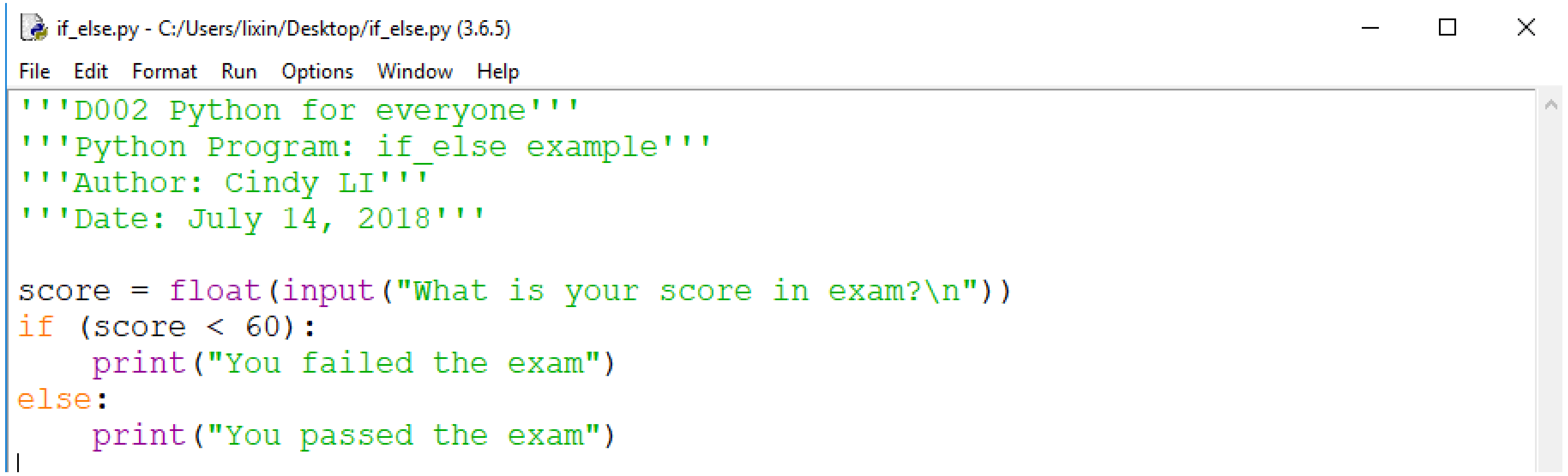
bracket () is optional but recommended for condition

```
=================== RESTART: C:/Users/lixin/Desktop/if.py ===================
What is your score in exam?
59
You failed the exam
>>>
=================== RESTART: C:/Users/lixin/Desktop/if.py ===================
What is your score in exam?
95
```

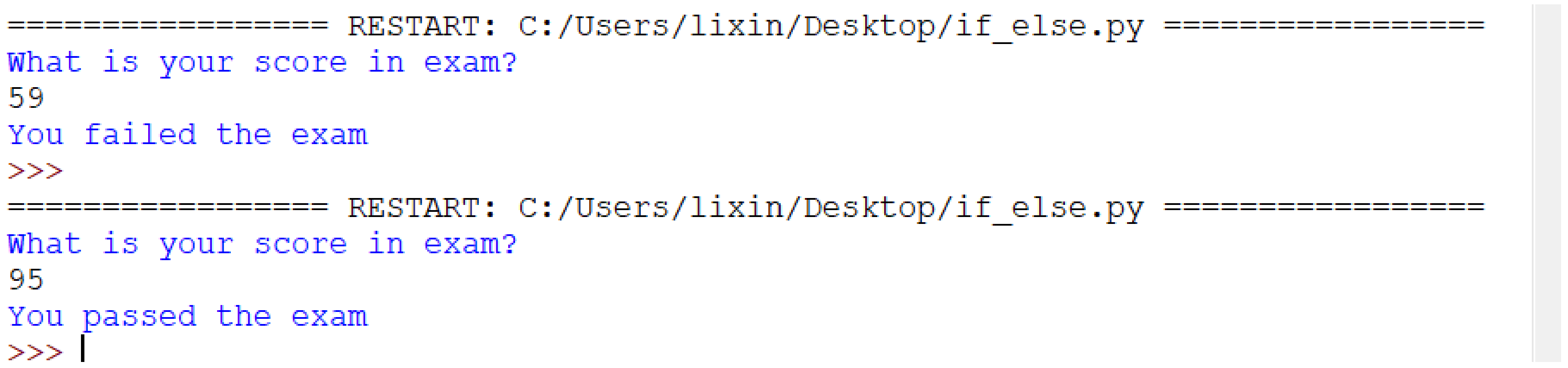
`if/else` structure



`if/else` example


if_else.py - C:/Users/lixin/Desktop/if_else.py (3.6.5)
File Edit Format Run Options Window Help

```python
'''D002 Python for everyone'''
'''Python Program: if_else example'''
'''Author: Cindy LI'''
'''Date: July 14, 2018'''

score = float(input("What is your score in exam?\n"))
if (score < 60):
    print("You failed the exam")
else:
    print("You passed the exam")
```



```
================= RESTART: C:/Users/lixin/Desktop/if_else.py =================
What is your score in exam?
59
You failed the exam
>>> 
================= RESTART: C:/Users/lixin/Desktop/if_else.py =================
What is your score in exam?
95
You passed the exam
>>> 
```

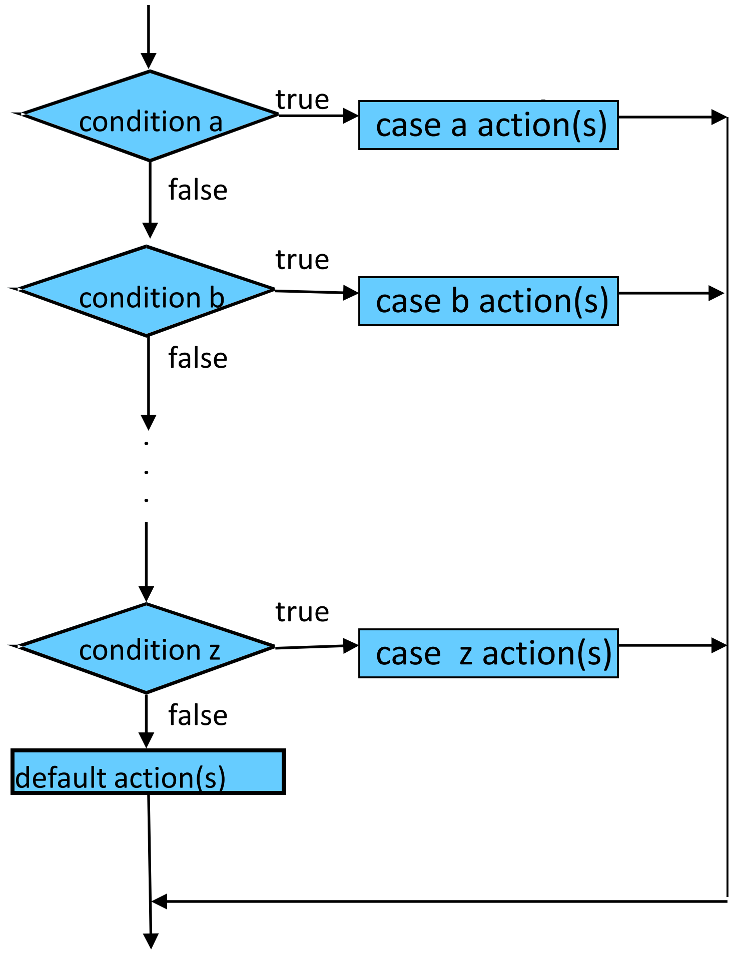
`if/elif/else` multiple selection

`if` statement

first **`elif`** statement

last **`elif`** statement

`else` statement



if/elif/else Example

```python
'''D002 Python for everyone'''
'''Python Program: if_elif example'''

score = float(input("What is your score in exam?\n"))
if (score >= 90):
    print("Your grade is A.")
elif (score >= 80):
    print("Your grade is B.")
elif (score >= 70):
    print("Your grade is C.")
elif (score >= 60):
    print("Your grade is D.")
else:
    print("Your grade is F.")
```

Why is it wrong?

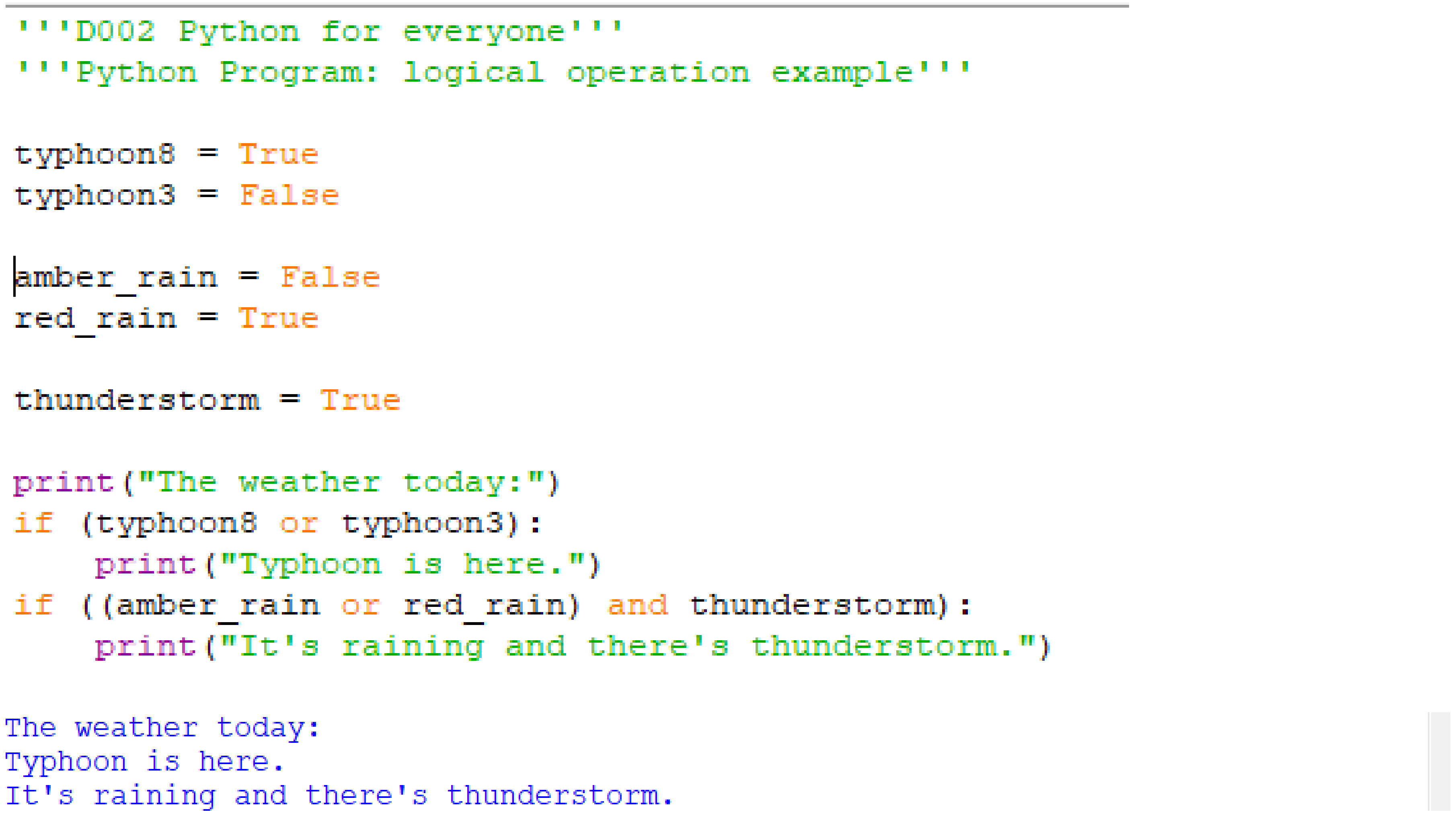
```python
score = float(input("Wrong example: enter your score?\n"))
if (score >= 90):
    print("Your grade is A.")
elif (score >= 60):
    print("Your grade is D.")
elif (score >= 70):
    print("Your grade is C.")
elif  (score >= 80):
    print("Your grade is B.")
else:
    print("Your grade is F.")
```

Logical Operation

- `and`
  - Binary. Evaluates to true if both expressions are true
- `or`
  - Binary. Evaluates to true if at least one expression is true
- `not`
  - Unary. Returns true if the expression is false

A: “She loves rich and handsome only. People like Kevin is not possible.”
B: “Really? I thought Kevin is not that bad.”

Logical operation example



```python
'''D002 Python for everyone'''
'''Python Program: logical operation example'''

typhoon8 = True
typhoon3 = False

amber_rain = False
red_rain = True

thunderstorm = True

print("The weather today:")
if (typhoon8 or typhoon3):
    print("Typhoon is here.")
if ((amber_rain or red_rain) and thunderstorm):
    print("It's raining and there's thunderstorm.")
```

```
The weather today:
Typhoon is here.
It's raining and there's thunderstorm.
```

Some more examples

- You want the user to enter a number between 1 to 10

```
n = input("enter a number")
if n >= 1 _____ n <= 10:    #should this be and/or?
      print("ok")
else:
      print("not ok")
```

Same?

```
n = input("enter a number")
if n < 1
       print("not ok")
elif n > 10:
       print("not ok")
else:
       print("ok")
```

- Only janitor or girls can use this toilet. Kevin cannot. This is because
  - Kevin is not a janitor _____ he is not a girl (# should this be and/or?)

```
janitor = False
girl = False
if not janitor:
    if not girl:
         print("forbidded")       # is this same logic as above?
```

Leap year – Q4 Homework

- ❑Input: year
- ❑Output: it’s leap year or not
- ❑Discuss with your neighbor student how should it be done
- ❑Test your program’s output with

1995, 2004, 2000, 1900

*Our definition of leap year: It is a leap year if the number is divisible by 4, but not by 100. But if it is divisible by 400, it is a leap year again.