Script (1A)

Young Won Lim
11/30/23

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Running a Python scripts

Python is a well-known high-level programming language.

The Python script is basically
a file containing code written in Python.

The file containing Python script has the extension '.py'
or can also have the extension '.pyw'
if it is being run on a Windows 10 machine.

To run a Python script,
we need a Python interpreter
that needs to be downloaded and installed.

https://www.geeksforgeeks.org/how-to-run-a-python-script/

Ways to run a Python scripts

Different ways to run Python Script

Here are the ways using which we can use to execute Python Programs.

- Interactive Mode
- Command Line
- Text Editor (VS Code)
- IDE (PyCharm)

https://www.geeksforgeeks.org/how-to-run-a-python-script/

Ways to run a Python scripts

Here is a simple Python script to print ‘Hello World!’.

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

To Execute this program first
we have to save it with ‘.py’ extension.

Then we can execute this file with the help of the terminal.

https://www.geeksforgeeks.org/how-to-run-a-python-script/

Execute Python Scripts

Execute Python scripts in the terminal or an IDE.

Python files have the .py extension.

Whenever you make a Python script, save it as name.py

A simple program (hello.py) is shown below.

The first line indicates that we want to use the Python interpreter.
The 3rd line outputs a line of text “hello wlrd” to the screen.

The text below can be copied into a text editor and save as hello.py.
Python works with files that end in .py.

```
#!/usr/bin/env python3

print('hello world')
```

You can use any text editor to create a Python program.

https://pythonbasics.org/execute-python-scripts/

Run Python

Run from terminal

You can start a Python program with the terminal or command line.
This works on all platforms (Mac OS, Windows, Linux).

Start program

To start the program, we have to open the command line and type:

```
python hello.py
```

For this to work you need to be in the correct directory.
That means, the directory where your python program is located.

https://pythonbasics.org/execute-python-scripts/

Run Python

Run from IDE

To run a Python script from an IDE, start a project first.

Once the project is created add your **.py** files
(or create them in the IDE) and press run.

In the **PyCharm** **IDE**:

- Start project
  - Welcome screen opens, click Create New Project.
  - On the main menu, choose File | New Project.
- Select Python interpreter
  - Choose Python version from the list. Use 3.x
- Click create
- Add new Python file (File new) and add **hello.py**
- Click the green triangle to start the program.

Another option is to click right mouse button on your Python file and selecting run.

Other IDEs have a similar process to run a Python program
(start project, add file, run button).

https://pythonbasics.org/execute-python-scripts/

Python Scripts and Interpreters

The interpreter processes the code in the following ways:

Processes the Python script in a sequence

Compiles the code into a byte code format
which is a lower-level language understood by the computers.

Finally, a Python Virtual Machine (PVM) comes into the picture.
The PVM is the runtime powerhouse of Python.
It is a process that iterates over the instructions of
your low-level bytecode code to run them one by one.

https://www.datacamp.com/tutorial/running-a-python-script

Python Scripts and Interpreters

Like scripts, you have something called Module, which is a Python script imported and used in another Python script.

The Python script is saved with a **.py** extension which informs the computer that it is a Python program script.

Unlike Windows, Unix-based operating systems such as Linux and Mac come with Python pre-installed.

Also, the way Python scripts are run in Windows and Unix operating systems differ.

https://www.datacamp.com/tutorial/running-a-python-script

Getting setting up

Command-line interpreter for Python can be accessed on the various operating systems in the following ways:

Like the Mac system, accessing the terminal on a Linux system is also very easy.

Right-click on the desktop and click Terminal in terminal type Python.

https://www.datacamp.com/tutorial/running-a-python-script

Writing Python scripts in the terminal

To accomplish this, first, you will type python3,
which means you will be using Python3 version.

After which, you can code typically
as you would in a text editor or an IDE,
though you will not be getting the functionalities
in the terminal as you would get with an IDE.

by just opening the terminal and typing Python3,
you can code in Python.

https://www.datacamp.com/tutorial/running-a-python-script

Writing Python scripts in the terminal

without even typing the **print** statement,
you were able to get the output.

```
python3
a = “Today”
b = “code”
a + “ “ + b
Today code
```

https://www.datacamp.com/tutorial/running-a-python-script

Writing Python scripts in the terminal

use the **NumPy** (Numerical Python) library
to create two arrays and
apply a few mathematical operations on it.

Numpy is a Python programming library
that has the capability of dealing with large,
multi-dimensional arrays and matrices,
along with an extensive collection of
high-level mathematical functions to operate on these arrays.

https://www.datacamp.com/tutorial/running-a-python-script

Writing Python scripts in the terminal

Let's complicate the code a bit
and lets you use the NumPy (Numerical Python) **library**
to create two arrays and apply a few mathematical operations on it.

```
python3
import numby as np
arr1 = np.array(([1,-2],[0,2],[10,4],[6,4]))
arr1.shape
(4, 2)
arr2 = np.array(([11,2],[10,-2],[1,1],[0,-4]))
arr2.shape
(4,2)
sum = np.add(arr1, arr2)
sum
array([12, 0],
       [10, 0],
       [11, 5],
       [  6, 0]])
```

https://www.datacamp.com/tutorial/running-a-python-script

Running Python scripts in the terminal

Running the Python script from the terminal is very simple,
instead of writing the Python script in the terminal
all you need to do is use a text editor
like vim, emacs or notepad++ and save it with a **.py** extension.

Then, open the terminal and go to the directory
where the code resides and
run the script with a keyword **python** followed by the script name.

```
python3   terminal.py
```

https://www.datacamp.com/tutorial/running-a-python-script

Running Python scripts in the terminal

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print("Output of adding x and y with a '+' operator:", x + y)
print("Output of adding x and y using 'numpy.add':", np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print("Output of subtracting x and y with a '-' operator:", x - y)
print("Output of subtracting x and y using 'numpy.subtract':", np.subtract(x, y))
```

https://www.datacamp.com/tutorial/running-a-python-script

Running Python scripts in the terminal

```
# Elementwise product; both produce the array
# [[ 5.0 12.0]
#  [21.0 32.0]]
print("Output of elementwise product of x and y with a '*' operator:", x * y)
print("Output of elementwise product of x and y using 'numpy.multiply':", np.multiply(x, y))

# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print("Output of elementwise division x and y with a '/' operator:", x / y)
print("Output of elementwise division x and y using 'numpy.divide':", np.divide(x, y))

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print("Output of elementwise square root x using 'numpy.sqrt':", np.sqrt(x))
```

https://www.datacamp.com/tutorial/running-a-python-script

Running Python scripts in the terminal

```
python3 terminal.py
```

```
Output of adding x and y with a `+` operator: [[6. 8.] [10. 12.]]
Output of adding x and y using a `numpy.add` : [[6. 8.] [10. 12.]]

Output of subtracting x and y with a `-` operator: [[-4. -4.] [-4. -4.]]
Output of subtracting x and y using a `numpy.subtract` : [[-4. -4.] [-4. -4.]]

Output of elementwise product of x and y with a `*` operator: [[5. 12.] [21. 32.]]
Output of elementwise product of x and y using a `numpy.multiply` : [[5. 12.] [21. 32.]]

Output of elementwise division of x and y with a `/` operator: [[0.2   0.33333333] [0.42857143 0.5]]
Output of elementwise division of x and y using a `numpy.divide` : [[0.2   0.33333333] [0.42857143 0.5]]

Output of elementwise square root of x using `numpy.sqrt` : [[1.   1.41421356] [1.73205081 2.]]
```

https://www.datacamp.com/tutorial/running-a-python-script

Passing Command Line Arguments

Within sys, you have **argv** which gives you
the list of command-line arguments passed to the Python program.

sys**.argv** reads the command line arguments as a list of items
where the first *item/element* in that list can be accessed as sys**.argv[1]**

while the first *argument*, i.e.,
sys**.argv[0]** is always the name of the program as it was invoked.

The command line argument is read as a string by Python,
so make sure to convert it as an integer in case you are dealing with numbers.

```
import sys
num = sys.argv[1]
for i in range(int(num)):
    print (i)
```

https://www.datacamp.com/tutorial/running-a-python-script

List index out of range error

You will get an **error** list index out of range,
which also reinforces that **sys.argv** reads as a list of items.

To avoid such errors, you need exceptional handling

```
$ python command_line.py
Traceback (most recent call last):
  File “command_line.py”, line 2, in <module>
    Num = sys.argv[1]
IndexError: list index out of range
$
```

https://www.datacamp.com/tutorial/running-a-python-script

List index out of range error

save the output of the Python script in a txt file using the > key.

You will first create a folder cli and
move the command_line.py code in the cli folder.

Then, you will type python3 command_line.py 10 > output.txt
and finally check the content of the cli folder.

```
$ mkdir cli
$ cd cli/
$ ../command_line.py
$ ls
command_line.py
$ python3 command_line.py 10 > output.txt
$ ls
command_line.py output.txt
$
```

https://www.datacamp.com/tutorial/running-a-python-script

List index out of range error

an interpreted programming or a script language.

Python is both an interpreted and a compiled language.

But calling Python a compiled language would be misleading.

People would assume that the compiler
translates the Python code into machine language.

Python code is translated into intermediate code,
which has to be executed by a virtual machine,
known as the PVM, the Python Virtual Machine.

This is a similar approach to the one taken by Java.

The question is, do I have to compile my Python scripts
to make them faster or how can I compile them?

The answer is easy: normally, you don't need to do anything
and you shouldn't bother,
because "Python" is already doing the thinking for you,
i.e. it takes the necessary steps automatically.

https://python-course.eu/python-tutorial/execute-a-script.php

List index out of range error

compile a python program manually

can be done with the module **py_compile**,
either using the interpreter shell

```
import py_compile
py_compile.compile('my_first_simple_program.py')
```

```
'__pycache__/my_first_simple_program.cpython-37.pyc'
```

or using the following command at the shell prompt

```
python -m py_compile my_first_simple_program.py
```

https://python-course.eu/python-tutorial/execute-a-script.php

List index out of range error

The compilation is hidden from the user for a good reason.

If Python has write-access for the directory where the Python program resides,
it will store the **compiled byte code** in a file that ends with a **.pyc** suffix.

If Python has no write access, the program will work anyway.

The byte code will be produced but discarded when the program exits.

Whenever a Python program is called, Python will check,
if a compiled version with the **.pyc** suffix exists.

This file has to be newer than the file with the **.py** suffix.

If such a file exists, Python will load the **byte code**,
which will speed up the start up time of the script.

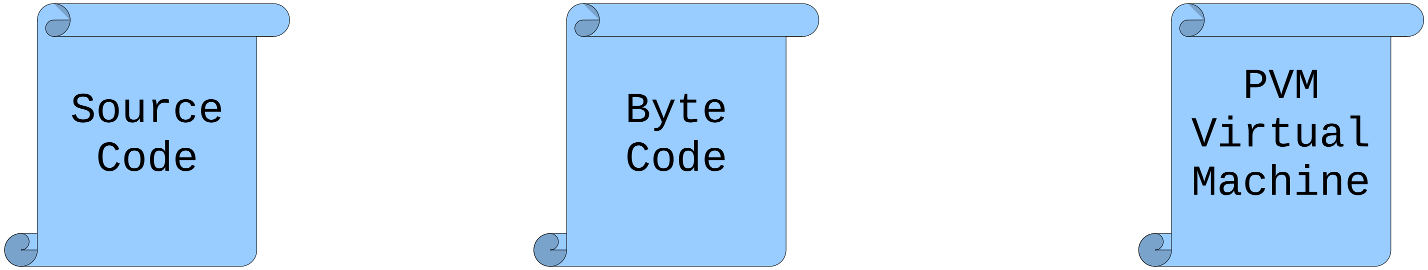
If there is no byte code version,
Python will create the byte code
before it starts the execution of the program.

Execution of a Python program means
execution of the byte code on the Python.

c

https://python-course.eu/python-tutorial/execute-a-script.php

List index out of range error



Source Code

Byte Code

PVM Virtual Machine

https://python-course.eu/python-tutorial/execute-a-script.php

PVM (Virtual Machine)

Compilation of a Python script

Every time a Python script is executed, a byte code is created.

If a Python script is imported as a module,
the byte code will be stored in the corresponding .pyc file.

So, the following will not create a byte code file:

```
$ python my_first_simple_program.py
My first simple Python script!
$
```

https://python-course.eu/python-tutorial/execute-a-script.php

PVM (Virtual Machine)

The import in the following Python2 session
will create a byte code file with the name
"my_first_simple_program.pyc":

```
$ ls
my_first_simple_program.py
$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import my_first_simple_script
My first simple Python script!
>>> exit()
$ ls
my_first_simple_program.py  my_first_simple_program.pyc
$
```

https://python-course.eu/python-tutorial/execute-a-script.php

Compilation of a Python script

Every time a Python script is executed, a byte code is created.

If a Python script is imported as a module,
the byte code will be stored in the corresponding .pyc file.

So, the following will not create a byte code file:

```
$ python my_first_simple_program.py
My first simple Python script!
$
```

https://python-course.eu/python-tutorial/execute-a-script.php

Compilation of a Python script

The import in the following Python2 session
will create a byte code file
with the name "my_first_simple_program.pyc":

```
$ ls
my_first_simple_program.py
$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import my_first_simple_script
My first simple Python script!
>>> exit()
$ ls
my_first_simple_program.py  my_first_simple_program.pyc
$
```

https://python-course.eu/python-tutorial/execute-a-script.php

On Linux (1)

python3 my_file.py

on the bash command line.

A Python script can also be started like any other script under Linux, e.g. Bash scripts.

Two steps are necessary for this purpose:

the shebang line #!/usr/bin/env python3 has to be added as the first line of your Python code file.

Alternatively, this line can be #!/usr/bin/python3, if this is the location of your Python interpreter.

Instead using env as in the first shebang line, the interpreter is searched for and located at the time the script is run.

https://python-course.eu/python-tutorial/execute-a-script.php

On Linux (2)

This makes the script more portable.

Yet, it also suffers from the same problem:

The path to env may also be different on a per-machine basis.

The file has to be made executable:

The command "chmod +x scriptname" has to be executed on a Linux shell,
e.g. bash. "chmod 755 scriptname" can also be used
to make your file executable. In our example:

```
$ chmod +x my_first_simple_program.py
```

https://python-course.eu/python-tutorial/execute-a-script.php

On Linux (3)

in a bash session:

```
$ more my_first_simple_script.py
#!/usr/bin/env python3
print("My first simple Python script!")

$ ls -ltr my_first_simple_script.py
-rw-r--r-- 1 bernd bernd 63 Nov  4 21:17 my_first_simple_script.py

$ chmod +x my_first_simple_script.py

$ ls -ltr my_first_simple_script.py
-rwxr-xr-x 1 bernd bernd 63 Nov  4 21:17 my_first_simple_script.py

$ ./my_first_simple_script.py
My first simple Python script!
```

https://python-course.eu/python-tutorial/execute-a-script.php

Compilers and Interpreters (1)

Compiler

Definition: a compiler is a computer program
that transforms (translates) source code of a programming language
into another computer language (the target language).

In most cases compilers are used
to transform source code into executable program,
i.e. they translate code from high-level programming languages
into low (or lower) level languages, mostly assembly or machine code.

https://python-course.eu/python-tutorial/execute-a-script.php

Compilers and Interpreters (2)

Interpreter

Definition: an interpreter is a computer program
that executes instructions written in a programming language.

It can either execute the source code directly or
translate the source code in a first step
into a more efficient representation
and execute this code.

https://python-course.eu/python-tutorial/execute-a-script.php