National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2018/2019

Mission 0
Setting Up Python

Release date: 14 August 2018
Due: 21 August 2018, 23:59

Required Files

- mission00-template.py

The objective of this mission is to guide you in the installation and setting up of the programming tool, called IDLE, that you will be using for the rest of the semester. Also, we include a simple exercise to help you familiarize yourself with the basics of Python, and also learn how to submit homework through Coursemology system.

This mission consists of only **one** task.

Part 1: Installing Python 3.7.0 and required packages

Before you start on your quest to master the Python programming language, you have to install the necessary tools. Please follow the following instructions to set up your programming environment for the class.

PIL is required to render images that will be used in the later missions. We will be using PILLOW, which is a modern replacement for PIL. Scipy/Numpy packages are also required in the later missions.

Note: The highest priority package to install is PILLOW, as we will be using it over the next few weeks.

Please follow the following instructions carefully.

Troubleshooting

You can contact the CS1010S staff or check the Coursemology forum if you face any difficulty during the setup/installation process.

Please provide the following information to help us quickly identify your problem:

1. The operating system and version that you are using. MacOS, Windows 10, 32-bit or 64-bit, etc.

2. Which step in the instructions given below did it fail or produce an error?

3. Take a screen shot of the failure or error.

Windows Users:

You may download the appropriate .zip.exe file from the links below. Please install the correct version (32 or 64 bit) depending on your OS.

- 32-bit: `http://www.comp.nus.edu.sg/~cs1010s/python-installer-32-bit.zip.exe`

- 64-bit: `http://www.comp.nus.edu.sg/~cs1010s/python-installer-64-bit.zip.exe`

You may check if your OS is 32 or 64 bit by either:

- Start menu > right-click "My Computer" or "This PC" > Properties, or

- Press WIN + Pause/Break

You should see your 32- or 64-bit version under "System Type".

Before continuing, check that you are connected to the Internet as some packages will be downloaded by the installer.
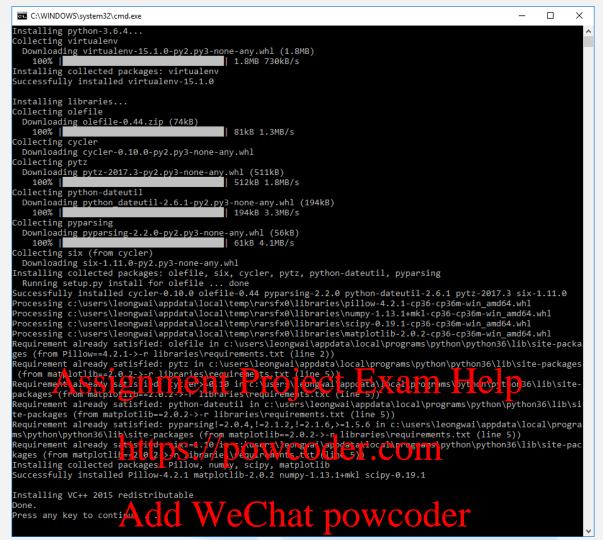
Run the self-executing zip file and extract the contents to an easily accessible folder like your Desktop.

A new folder `python-installer-xx-bit` will be created at the extracted location and the Python installer **should automatically run**. In case it does not, open this folder and run the `install-python` file. Do **NOT** run the other files.

You will need an administrator account for a system update of the C Runtime Library. Also, the Visual C++ 2015 redistribution will be installed for some of the Python libraries to work.

If successful, you should not get any error along the way and see an output similar to this:

```
C:\WINDOWS\system32\cmd.exe                                          —    □    ×

Installing python-3.6.4...
Collecting virtualenv
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
    100% |████████████████████████████████| 1.8MB 730kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0

Installing libraries...
Collecting olefile
  Downloading olefile-0.44.zip (74kB)
    100% |████████████████████████████████| 81kB 1.3MB/s
Collecting cycler
  Downloading cycler-0.10.0-py2.py3-none-any.whl
Collecting pytz
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |████████████████████████████████| 512kB 1.8MB/s
Collecting python-dateutil
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
    100% |████████████████████████████████| 194kB 3.3MB/s
Collecting pyparsing
  Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 4.1MB/s
Collecting six (from cycler)
  Downloading six-1.11.0-py2.py3-none-any.whl
Installing collected packages: olefile, six, cycler, pytz, python-dateutil, pyparsing
  Running setup.py install for olefile ... done
Successfully installed cycler-0.10.0 olefile-0.44 pyparsing-2.2.0 python-dateutil-2.6.1 pytz-2017.3 six-1.11.0
Processing c:\users\leongwai\appdata\local\temp\rarsfx0\libraries\pillow-4.2.1-cp36-cp36m-win_amd64.whl
Processing c:\users\leongwai\appdata\local\temp\rarsfx0\libraries\numpy-1.13.1+mkl-cp36-cp36m-win_amd64.whl
Processing c:\users\leongwai\appdata\local\temp\rarsfx0\libraries\scipy-0.19.1-cp36-cp36m-win_amd64.whl
Processing c:\users\leongwai\appdata\local\temp\rarsfx0\libraries\matplotlib-2.0.2-cp36-cp36m-win_amd64.whl
Requirement already satisfied: olefile in c:\users\leongwai\appdata\local\programs\python\python36\lib\site-packa
ges (from Pillow==4.2.1-.>-r libraries\requirements.txt (line 2))
Requirement already satisfied: pytz in c:\users\leongwai\appdata\local\programs\python\python36\lib\site-packages
 (from matplotlib==2.0.2-.>-r libraries\requirements.txt (line 5))
Requirement already satisfied: ... in c:\users\leongwai\appdata\local\programs\python\python36\lib\site-
packages (from matplotlib==2.0.2-.>-r libraries\requirements.txt (line 5))
Requirement already satisfied: python-dateutil in c:\users\leongwai\appdata\local\programs\python\python36\lib\si
te-packages (from matplotlib==2.0.2-.>-r libraries\requirements.txt (line 5))
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=1.5.6 in c:\users\leongwai\appdata\local\progra
ms\python\python36\lib\site-packages (from matplotlib==2.0.2-.>-r libraries\requirements.txt (line 5))
Requirement already satisfied: ...=1.0/i... c:\users\leongwai\appdata\local\programs\python\python36\lib\site-pac
kages (from matplotlib==2.0.2-.>-r libraries\requirements.txt (line 5))
Installing collected packages: Pillow, numpy, scipy, matplotlib
Successfully installed Pillow-4.2.1 matplotlib-2.0.2 numpy-1.13.1+mkl scipy-0.19.1

Installing VC++ 2015 redistributable
Done.
Press any key to continue . . .
```

Note the version numbers will be different but the process is similar.

If you see any red text, it means something have gone wrong somewhere. Please check the forums for troubleshooting.

IMPORTANT: Please do **NOT** install both versions (32 and 64 bit) of Python in your computer at the same time. If you need to install the other version, please **uninstall** your existing Python installation first before installing the other version.

Mac Users:

You may download Python 3.7.0 from `https://www.python.org/downloads`. Download and install the appropriate dmg installer for the version of your Operating System. Be sure to install Python for "All Users", not just the current user. Once installation is completed, you should see that IDLE is available from your finder.

You may need to install Tcl/Tk to run IDLE, more instructions can be found at `http://www.python.org/download/mac/tcltk/`. Download the newest version from the recommended Tcl/Tk column according to your OS version.

Setting up Command Line Tools for Mac

Install the Command Line Tools (required for PILLOW installation). Follow the instructions depending on your OS X version. You can determine the version by clicking on the Apple Icon (in the menu bar) > About This Mac.

- **Mavericks (10.9) or later**

  Run the following commands in your terminal. (You can find the terminal by clicking on Finder on the dock, Go > Utilities > Terminal.)

  `xcode-select --install`

  A popup will appear, asking if you wish to install the command line developer tools. Click on Install to begin the installation.

- **Mountain Lion (10.8) and earlier**

  Install Xcode from the App Store. Open Xcode and go to Preferences. Click on the Downloads tab, and you'll see Command Line Tools. Click the Install button to install the Command Line Tools.

Setting up Homebrew and dependencies

After installing the Command Line tools, Run the following commands in your terminal. (You can find the terminal by clicking on Finder on the dock, Go > Utilities > Terminal.) You will **need to be connected to the Internet** as the installer will download the required files.

First, we will install Homebrew, a package manager for MacOS.

Visit `https://brew.sh/` and follow the instruction on that page. Basically you should copy and paste the given command in your Terminal prompt.

After which, enter the next two commands into the Terminal as separate lines, i.e., press Enter after entering each line.

```
brew install freetype
brew install pkg-config
```

Setting up PILLOW, Scipy/Numpy, Matplotlib

In the same terminal, run the following commands. This will install pip — a package manager for Python, with the following command.

```
curl https://bootstrap.pypa.io/get-pip.py | sudo python3
```

Follow up with the next 3 commands:

```
sudo pip3.7 install PILLOW
sudo pip3.7 install scipy numpy
sudo pip3.7 install matplotlib
```

Note: You may be prompted to enter your Mac password when you run the above commands. **It is normal for nothing (not even *) to appear in the Terminal as you key in your password. Just type it in and press Enter.**

Linux Users:

If you are *really* using Linux, then you are 1337 and do not need any help installing Python. :)

Just kidding. You would probably have to find the TA for help as the tutors might not be 1337 enough.

Editing Python Files

The default behaviour of double clicking on the Python file **executes** the content of the Python file. You should see the a command line window briefly opens, and close when Python has finish executing the file.

In order to make changes to the Python file, you will need to **edit** the file using the IDLE program.

- Windows User: Right click on the Python file > **Edit with IDLE**

- Mac Users: Right click on the Python file > **Open With** > **IDLE**

The content of the Python file should now appear in the IDLE program. You can then make changes to the file, and execute it.

To execute the Python file, go to **Run** > **Run Module**. The output of your Python file should then appear in the Shell Window.

Part 2: The Task

Consider the following Python expressions. Your job is to predict the output when each expression is evaluated in IDLE.

Before checking your answers with IDLE, write down `briefly` your guess of what the interpreter would **display** when the expressions are evaluated **sequentially** as listed. If you do not expect any output, you may write "no output" and if you expect an error, you may write "error".

Now, run the code by removing the # in the front of the respective lines in the template file. You should comment out error-causing definitions by adding # to the front of such definitions to allow IDLE to skip processing them. (Or by choosing the area that you would like to comment out and then pressing IDLE's hot-key `alt+3` for windows, `control+3` for macOS)

After you have checked your answers, if any expression has evaluated differently from your expected answer, write down what it evaluated to **below** your expected answer. **Please note that you will be graded only on your <u>final</u> answer (If you have no corrections to make, your initial answer will be your final answer).**

However, if any expression would generate error message after running, please **specify the type of error** (such as `TypeError`) together with the **error message** in your final answer. You should not include the full error output. The wanted answer is usually the *last line* of the error output only.

Please use the template file provided in **mission00-template.py** instead of copying from this PDF file.

Reminder: Lines that begin with a # are comments (text that do not affect the Python execution). To execute a particular expression, ensure it does not begin with a #.

```python
print(42)
# expected answer:
# final answer:

print(0000)
# expected answer:
# final answer:

print("the force!")
# expected answer:
# final answer:

print("Hello World")
# expected answer:
# final answer:

print "Hello World"
# expected answer:
# final answer:

print(6 * 9)
```

```
# expected answer:
# final answer:

print(2 + 3)
# expected answer:
# final answer:

print(2 ** 4)
# expected answer:
# final answer:

print(2.1**2.0)
# expected answer:
# final answer:

print(15 > 9.7)
# expected answer:
# final answer:

print((5 + 3) ** (5 - 3))
# expected answer:
# final answer:

print(--4)
# expected answer:
# final answer:

print(1 / 2)
# expected answer:
# final answer:

print(1 / 3)
# expected answer:
# final answer:

print(1 / 0)
# expected answer:
# final answer:

print(7 / 3 == 7 / 3.0)
# expected answer:
# final answer:

print(3 * 6 == 6.0 * 3.0)
# expected answer:
# final answer:

print(11 % 3)
# expected answer:
# final answer:
```

```python
print(2 > 5 or (1 < 2 and 9 >= 11))
# expected answer:
# final answer:

print(3 > 4 or (2 < 3 and 9 > 10))
# expected answer:
# final answer:

print("2" + "3")
# expected answer:
# final answer:

print("2" + "3" == "5")
# expected answer:
# final answer:

print("2" <= "5")
# expected answer:
# final answer:

print("2" + "3")
# expected answer:
# final answer:

print("May the force" + "be" + "with you")
# expected answer:
# final answer:

print("force"*3)
# expected answer:
# final answer:

print('daw' in 'padawan')
# expected answer:
# final answer:

a, b = 3, 4

print(a)
# expected answer:
# final answer:

print(b)
# expected answer:
# final answer:

a, b = b, a

print(a)
```

```
# expected answer:
# final answer:

print(b)
# expected answer:
# final answer:

print(red == 44)
# expected answer:
# final answer:

red, green = 44, 43

print(red == 44)
# expected answer:
# final answer:

print(red = 44)
# expected answer:
# final answer:

print("red is 1") if red == 1 else print("red is not ")
# expected answer:
# final answer:

print(red - green)
# expected answer:
# final answer:

purple = red + green

print("purple")
# expected answer:
# final answer:

print(red + green != purple + purple / purple - red % green)
# expected answer:
# final answer:

print(green > red)
# expected answer:
# final answer:

print("green bigger") if green > red else print("red equal or bigger")
# expected answer:
# final answer:

print(green + 5)
# expected answer:
# final answer:
```

```
print(round(1.8))
# expected answer:
# final answer:


print(int(1.8))
# expected answer:
# final answer:


# The following questions are to ensure that you have installed
# PILLOW, matplotlib, scipy, and numpy correctly.
# Do not worry about the syntax - just run the line and observe the output
# (if any)

from PIL import *
# expected answer:
# final answer:


# Please check with your tutor if you still encounter errors for the
# following statements. This is only required for Side Quest 8.1

from matplotlib import *
# expected answer:
# final answer:

from numpy import *
# expected answer:
# final answer:

from scipy import integrate
# expected answer:
# final answer:
```

To submit your work to Coursemology, complete **mission00-template.py** and copy the contents from the file into the box that says "Your answer" on the mission page, and click "Save." You can continue to make changes to your submission if you wish.

Once you are satisfied with your solution, click "Finalize Submission." **Note that once your submission is finalized, it is considered to be submitted and cannot be changed.** If you need to undo this action, you will have to email your tutor or the lecturer. You will not be allowed to undo your *finalized* submission, or a penalty might be imposed. Please do not finalize your submission until you are sure that you want to submit your solutions for grading.