Laboratory Manual
for
CE/CZ1003
Introduction to Computational Thinking

Practical Exercise #4:
Basic Python Programming -
Conditional Executions and Iterations

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

Ex. #4 – Basic Python Programming

Learning Objectives

The manual provides information and practical exercises to let you learn how to apply the relational operations, selections and repetition to implement robust programs through hands-on coding exercises on the Raspberry Pi.

Intended Learning Outcomes

At the end of this exercise, you should be able to

- apply the concept of relational operations with selection to form conditional expressions.
- make use of iteration in your program to perform repetitive tasks.

Equipment and accessories required

i) Raspberry Pi 3 Model B (RPi3) board with Sense HAT add-on display module/board.
ii) A USB power source to power the RPi3 board (E.g. Power Bank, Adaptor or USB port of a desktop computer).
iii) A computer (desktop PC or notebook) with Ethernet port and cable for remote access of RPi3. Software (open source) to be installed on the computer – PuTTY, VNC Viewer and WinSCP

---

1. Conditional Execution

For a program to be useful and practical, it is essential that the program has the ability to check conditions and alter its behaviour accordingly. The simplest way to perform this conditional execution is to make use of the `if` statement, which has the general form of

```
if <boolean expression>:
    statement
^^^^    :
    statement
```

Note that the body of the compound statement that get executed when the condition (i.e. the boolean expression) is true must be indented by the same amount, and it is recommended to use 4 spaces for indenting in Python.

In practice, it is more common to use the `if-else` and `if-elif-else` statements. (Refers to LAMS module 3.2 for detail and examples on how they can be used.)

2. Iteration

Repeated execution of a set of statements is known as iteration. Iteration is used extensively in practical programs to repeat similar tasks, either for a specified number of times or until a condition is met.

There are two methods to perform iterations in Python: using the `while` statement and using the `for` statement. (Refers to LAMS module 3.3 for detail and examples on how they can be used.) For example, the following is the general form for the while statement:

```
while <boolean expression>:
    statement
^^^^    :
    statement
```

Obviously, you can combine the `if/else` statement within the `while` statement to perform more sophisticated tasks in your program.

Ex. #4 – Basic Python Programming

3. Coding Exercises

<u>Exercise 4a</u>

Turn on your RPi board. Start the Cloud9 IDE. Make sure that the Cloud9 IDE is configured for Python 3 (See Appendix for detail)

- Code a Python 3 program to display a message (e.g. "This is fun!") on the Sense Hat board with the following specifications (similar to the one you developed in Exercise 3b earlier).
    - The program will prompt the user to choose the colour of the message, and the scroll speed of the message.

<u>Exercise 4b</u>

Ask your classmate sitting beside/nearby you to run your program (on your RPi) to verify whether your program is working as expected, while you also test your classmate's program on his/her RPi (e.g. swap your seat).

- Your task is to see whether you can crash each other program during execution (i.e. make the program terminates without printing the message.)

- If your program crashed during execution, ask your classmate what he/she has done to cause it to malfunction. In computing, this is known as having "**bug(s)**" in your program.

- Study your code carefully, and think about how you will resolve the problem, i.e. how to debug your program.

  Tips: See section 1 and 2 above, and google for potential function(s) that may be useful to solve the problem.

- Sketch/Draw a flowchart of your refined design

- Implement the new design in your program.

- Once completed, ask your classmate to test your program again (and vice versa) until you are satisfied that your program can perform properly <u>under all conditions</u>.

- Compare each other approach used to solve the problem, and discuss which one you think is the 'better' solution.

<u>Exercise 4c (Optional)</u>

Add the following features to your program

- The program will repeat (i.e. ask for different colour) until the user issues a command to quit.

- The program will allow the user to choose the colour of the background.
    - Consider the scenario that the user can choose the same colour for both the message and the background. How would you resolve this in your design?

Appendix – Configure Cloud9 IDE for Python 3

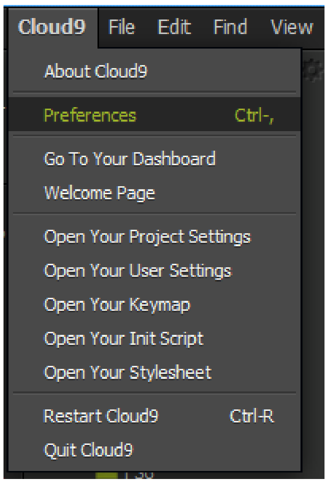
Python programming language was conceived in the late 1980s, and has since gone through several revisions, with version 3 being the latest (specifically, version 3.7 that was released on June 27, 2018). (https://en.wikipedia.org/wiki/History_of_Python). But Python 2 (2.7) can still be found in many existing/legacy applications.

There are plenty of differences between version 2 and version 3 of the Python language. Two of the differences you will notice when you code programs in this course are the `print()` and `input()` functions.

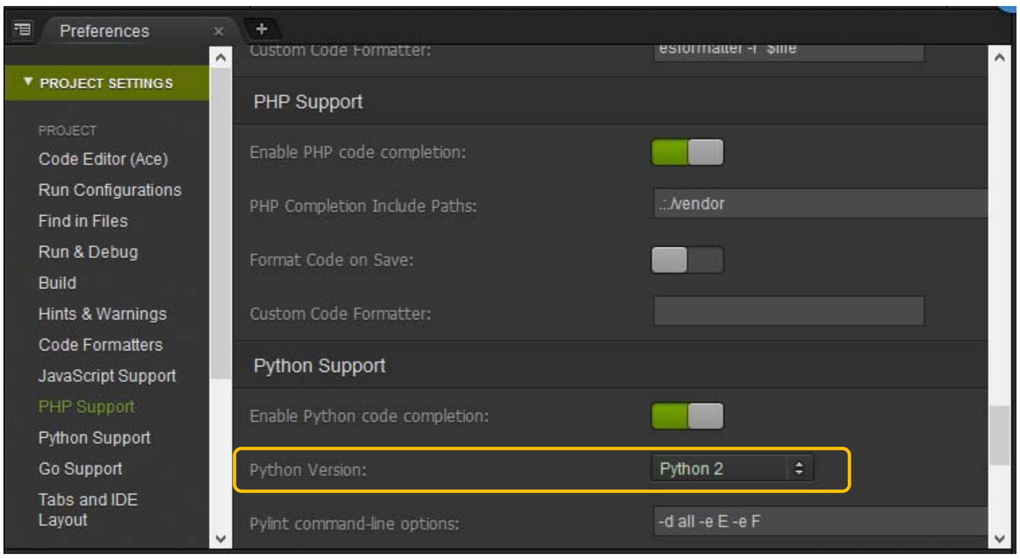
- In Python 2, the print statement is written as `print "Hello world"`, while in Python 3, print statement is replaced as a function and is written as `print ("Hello world")`.
- In Python 2, the `input()`function evaluates the data it receives and returns the data type based on what it 'thinks' it should be. In Python 3, `input()` function always returns the data as str (i.e. string) data type object.

Cloud9 IDE supports both Python 2 and Python 3, but it is obvious that we should learn the latest, which is Python 3. As such, you will configure Cloud9 IDE for Python 3 in this course, using the steps as described below.

i) Select the **`Preferences`** option under the Cloud9 menu



ii) The **`Preferences`** window pane will be shown. Look for the **`Python Version`** setting.



Custom Code Formatter: esformatter -i $file

PHP Support

Enable PHP code completion:

PHP Completion Include Paths: .:./vendor

Format Code on Save:

Custom Code Formatter:

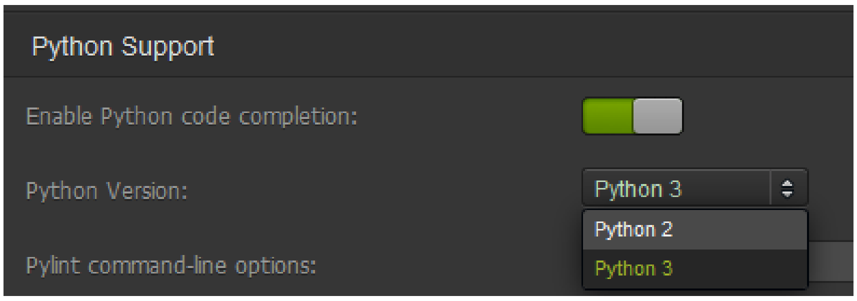
Python Support

Enable Python code completion:

Python Version: Python 2

Pylint command-line options: -d all -e E -e F

iii) If the **Python Version** is shown as **Python 2**, change it to **Python 3**.

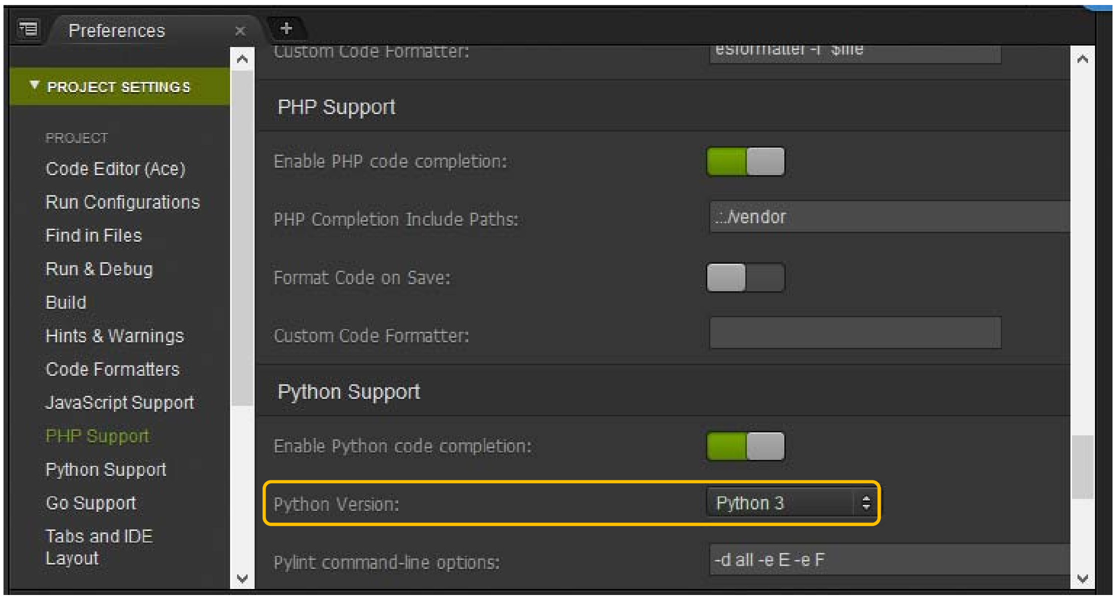


Python Support

Enable Python code completion:

Python Version: Python 3

Pylint command-line options:



Custom Code Formatter: esformatter -i $file

PHP Support

Enable PHP code completion:

PHP Completion Include Paths: ./vendor

Format Code on Save:

Custom Code Formatter:

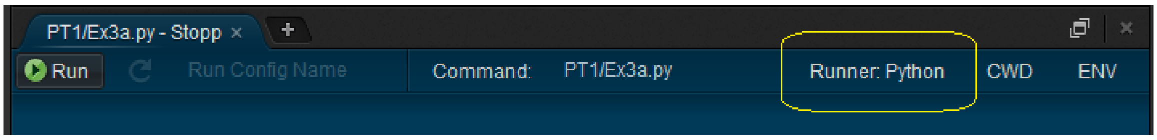
Python Support

Enable Python code completion:

Python Version: Python 3

Pylint command-line options: -d all -e E -e F

iv) In addition, when you later run your program, check that the setting for **Runner** is set to **Python** (and NOT Python 2 Runner: Python 2).



Command: PT1/Ex3a.py

Runner: Python