ECE40862: Software for Embedded Systems

Fall 2019

Lab 0 – Introduction to Python

To be done individually; Due by 11:59pm, Monday, August 26, 2019.

1. Overview

This assignment will familiarize you with Python programming language and the **Thonny Python IDE editor**, which you will use throughout the semester. Python is an easy-to-learn but powerful programming language and it's the fastest growing language for embedded computing. In this course, you will be using **MicroPython** to program **ESP32 board**. MicroPython is a lean and efficient implementation of Python3 programming language. Before working with MicroPython on your board, acquaint yourself with Python basics.

2. Getting started with Python and MicroPython

The following sections describe various software installation procedures. You should perform all the necessary installations apart from section 2.6 before you receive your board.

2.1. Installing Python on your own PC

This webpage outlines common ways to install python on your own laptop/PC. https://realpython.com/installing-python/. You will find installation instructions for different OS: **Windows, Linux, Mac OS**, etc. Follow the instructions relevant to the OS on your PC and install the LATEST PYTHON3 VERSION.

2.2. Installing Thonny IDE on your own PC

Thonny is a Python IDE meant for learning programming. You will find more details about this IDE on https://thonny.org and https://github.com/thonny/thonny/wiki. Installation instructions for **Windows, Linux, Mac OS** are provided in this webpage: https://randomnerdtutorials.com/getting-started-thonny-micropython-python-ide-esp32-esp8266/. You might get a *Windows Smart Screen Filter* warning when trying to install Thonny IDE on Windows, you can safely ignore it and go ahead with installation. More details can be found at https://github.com/thonny/thonny/wiki/Windows. Install LATEST VERSION 3.2

2.3. Install USB Driver on your own PC (Windows and Mac only)

Download and install **CP210x USB to UART Bridge VCP Drivers** so that your machine identifies when you connect ESP32 board to your PC. You can download files here: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers. Linux machines usually do not need this driver, however, if your Linux machine is unable to detect the board, you might want to install Linux version of this driver.

2.4. Installing rshell, esptool on your own PC

2.4.1. Rshell

Remote MicroPython shell (rshell) is a simple shell which runs on the host (in your case, ESP32 board) and uses MicroPython's raw-REPL (Read-Evaluate-Print-Loop) to send python snippets to the microcontroller board in order to get filesystem information, and to copy files to and from MicroPython's filesystem. It also can invoke the regular REPL, so can be used as a terminal emulator. You will find more information on rshell on this webpage: https://github.com/dhylands/rshell.

```
>> pip3 install rshell
```

2.4.2. Esptool

Esptool is a Python-based, open source, platform independent, utility to communicate with the ROM bootloader in ESP32 chips. **Esptool** can be used to load MicroPython on your ESP32 board. You will find further details here: https://github.com/espressif/esptool. Install esptool on your machines using any of the following commands.

```
>> pip3 install esptool
                                    OR
>> python3 -m pip install esptool
```

NOTE: You can either use *rshell* or *Thonny IDE* on your own PC for programming board. In EE217 machines, you will be using *rshell* only for code demonstration.

2.5. Installing rshell, esptool in LAB (EE217) machine

Python 3.6.8 is installed by default in the lab machines in EE217. You need to create a virtual environment to install any packages in these machines. **venv** (for Python 3) allows you to manage separate package installations for different projects. They essentially allow you to create a "virtual" isolated Python installation and install packages into that virtual installation. Login to your career account on the lab machine, go to your local project's directory and run venv as shown below. Before you can start installing or using packages in your virtual environment, you'll need to activate it. Activating a virtual environment will put the virtual environment-specific **python** and **pip** executables into your shell's **PATH.** More information on **venv** is available here: https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/. Once you activate your virtual environment, you can easily install *rshell* and *esptool* using pip inside your environment using commands shown here.

```
>> python3 -m venv my_py_env          # create virtual environment
>> source my_py_env/bin/activate      # activate environment
(my_py_env)>> pip3 install rshell     # install rshell
(my_py_env)>> pip3 install esptool    # install esptool
```

2.6. Flashing MicroPython on ESP32 Board

Download latest **ESP32 firmware** from http://micropython.org/download. As of today, the latest available firmware is '**esp32-20190821-v1.11-230-gfe3c064d4.bin**'. Program your board using the **esptool.py** program. If you are putting MicroPython on your board (identified as ***/dev/ttyUSBx**,* x=0/1/2) for the first time, then erase the entire flash using:

```
>> esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```

Then program the firmware starting at address **0x1000** using:

```
>> esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z
0x1000 esp32-20190821-v1.11-230-gfe3c064d4.bin
```

2.6.1. Flashing MicroPython (2nd Method)

If you are not able to install ***rshell*** and ***esptool*** on your machine (Windows and MAC), you can install **uPyCraft IDE** following the instructions on https://randomnerdtutorials.com/install-upycraft-ide-windows-pc-instructions/. Subsequently, follow the instructions given on https://randomnerdtutorials.com/flash-upload-micropython-firmware-esp32-esp8266/ to flash the downloaded MicroPython firmware on the ESP32 board using **uPyCraft IDE.**

3. Python programming exercises

3.1. Character Input (Upload as program1.py)

Write a program that asks the user to enter name and age. Prints out a message that tells the year the user will turn 100 years old.

Hint: Use **input** command to get user input

```
>>> python program1.py
What is your name? Xavier
How old are you? 25
Xavier will be 100 years old in the year 2094
```

3.2. Lists and Conditional Statements (Upload as program2.py)

Write a program to initiate a list of numbers, print the list, and ask the user for a number and return a list that contains only elements from the original list that are smaller than the number given by the user.

```
>>> python program2.py
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
Enter number: 25
The new list is [1, 1, 2, 3, 5, 8, 13, 21]
```

3.3. Loops

3.3.1. While loop: (Upload as program3a.py)

Write a program to get Fibonacci series between 0 to a **user input number** using **while** loop

```
>>> python program3a.py
How many Fibonacci numbers would you like to generate? 10
The Fibonacci Sequence is: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

3.3.2. For loop: (Upload as program3b.py)

Write a program that generates a random number (0-10) and asks the user to guess it within three chances. If user guesses correctly, print 'You win!', otherwise print 'You lose!'

```
>>> python program3b.py
Enter your guess:3
Enter your guess:2
Enter your guess:10
You win!
```

3.4. Dictionary (Upload as program4.py)

Write a program to create a dictionary of names and birthdays. Upon execution, it should ask the user to enter a name, and return the birthday of that person back to them.

```
>>> python program4.py
Welcome to the birthday dictionary. We know the birthdays of:
Albert Einstein
Benjamin Franklin
Ada Lovelace
Who's birthday do you want to look up?
Benjamin Franklin
Benjamin Franklin's birthday is 01/17/1706.
```

3.5. Class and Functions (Upload as program5.py)

Write a program to create a Python class to find a pair of elements (indices of the two numbers) from a given list of numbers whose sum equals a specific target number.

Use this list in your program: [10,20,10,40,50,60,70]

Hint: You might create dictionaries of array index as *keys* and array item as *values* as you scan through the array. Use conditional statements, loops, etc.

```
>>> python program5.py
What is your target number? 60
index1=1, index2=3
```

REFERENCES

[1] https://docs.python.org/3/tutorial/

[2] https://realpython.com