WPI

ECE3311 – Principles of Communication Systems

Project 01

Due: 5:00 PM, Monday 2 November 2020

1 Project Objective & Learning Outcomes

The objective of this project is to provide a brief introduction to the Python programming language and illustrate via several examples how it can be used to implement some of the basic building blocks of a communication systems. In addition to the basic syntax of Python, several key libraries will be introduced to assist you with this course, such as NumPy and Matplotlib. Finally, techniques for running your code and saving the results using Jupyter Notebooks will be presented.

From this project, it is expected that the following learning outcomes are achieved:

- Increase familiarity of the Python programming language and establish the necessary foundation for implementing communication system models for this course.
- Understand available libraries and tools that can be leveraged to help implement communication system models in a computer simulation environment.
- Acquire expertise in presenting results of Python-based communication system simulation models and representation of project in Jupyter Notebook format.

2 Preparations

For this course, a virtual disk image (VDI) will be produced for and distributed to the class in order to minimize effort needed to implement the Python development environment for ECE3311. To use this image, the host computer will need to run a virtual machine (VM) software program called VirtualBox (www.virtualbox.org), which is supported across most operating systems. Instructions on how to install VirtualBox is available via a short video presentation located in the ECE3311 Canvas website in "*Files* → *Projects* → *VirtualBox Install* → *ece3311b20_install_virtualbox.mp4*". Once VirtualBox has been installed, you can use the distributed image to run the ECE3311 "ready-to-use" Ubuntu 20.04.1 Linux operating system with pre-installed Python3, libraries, tools, etc.

If you plan on using Python, associated libraries, and tools on your host computer, here is the minimum required version information for these items:

- Python 3.8.5
- Pip 20.2.3
- NumPy 1.19.2
- SciPy 1.5.3
- Matplotlib 3.3.2

- VS Code 1.50.1

The last item, VS Code, is Visual Studio Code, a free source code editor by Microsoft that possesses several convenient features for supporting programming in Python[1]. One feature that will be used extensively with each project in this course is the generation of a Jupyter Notebook[2], which presents both source code and software outputs in a human-accessible format.

Please note that if you decide to build your own development environment, we will be unable to assist you in the event there are issues regarding missing libraries and other environment-related problems.

For this project, you will need to download the Jupyter Notebook "*Project 1 ECE3311.ipynb*" from the ECE3311 Canvas website and open it in VSCode, which will provide you with all the example source code and outputs of subsequent topics discussed in this project[3].

3 Getting Started with Python Basics

Python is an extensive programming language that possesses numerous features and functions, making it very popular across a wide range of applications including communication systems prototyping, machine learning/artificial intelligence, finance, and robotics. Although there exists entire courses dedicated to learning Python, the focus of ECE3311 is communication systems. Consequently, we will learn just enough Python in order to study communication systems in this course. As a result, the purpose of this project is to provide a quick introduction to some of the basic syntax you will be using for the remaining five project, although it is expected you will need to learn additional Python commands, features, and techniques as we progress throughout the term and your competency in Python increases. In this section, we will look at several core features of Python that will help establish a solid foundation for programming in this language.

3.1 Import Command

Since Python does not automatically load modules when running a program, it is necessary to explicitly import them into your function using the `import` function. For example, to import NumPy, one would write `import numpy as np` at the top of your program. One important observation is that in addition to importing a module, you can also abbreviate that module to help simplify the programming process. For instance, writing `import numpy as np` means that `np` represents the module `numpy`.

3.2 Data Types and Structures

Similar to other programming languages, Python has several data types for variables, including: integer (`int`), float (`float`), string (`str`), and boolean (`bool`). For instance, the expression `x_1="Hello Class!"` is a variable of type `str` while the expression`x_2 = 3311` is a variable of type `int`. Another data type that we will use extensively in this course is complex, which can be expressed as the sum

[1]Details on how to setup Python environments in VS Code is available from *https://code.visualstudio.com/docs/python/environments*.

[2]Details on setting up VS Code to work with Jupyter Notebooks is available from *https://code.visualstudio.com/docs/python/jupyter-support*.

[3]All Jupyter Notebooks possess the file extension ".ipynb".

of a real-valued number and an imaginary-valued number, *e.g.*, `x_4=3311+1j` where `j` represents the square-root of -1.

In addition to these data types, it is important to be able to organize data into a structure to facilitate various applications, including operations such as convolution, filtering, Fourier transforms, and more. These structures are referred to as sequence types, and consist of the following: lists, tuples, and ranges. Lists and tuples are basically an ordered array of values. Although similar, lists are mutable, which means the individual elements in the list can be replaced. On the other hand, tuples are immutable such that values cannot be added, changed, or removed once the tuple has been defined. As for ranges, they produce an sequence incremental list of numbers, *e.g.*, `x_6 = range(6)` produces a sequence of integer numbers from 0 to 5.

Note that Python does not like it when you mix up variables of different types, and therefore you will need to *type cast* variables to the appropriate type before using it.

Question 1 (1 point): Suppose you are given the following snippet of code:

```
x=21
y="My kid is "
z="months old today!"
a=y + x + z   # String concatenation
print(a)
```

but an error occurs instead. Use type casting to resolve the error. Please provide source code as well as the output.

Question 2 (1 point): Describe the difference between a list and a tuple, and explain why we have these two different sequence types.

3.3 For Loops

For loops are a critical feature is all programming languages since it provides us with a framework to repeatedly perform a collection of actions using only a small amount of code. In Python, for loops are defined using an expression similar to `for i in range(10):`, where the `for` denotes this block of code is a for loop, and `i in range (10):` defines a index variable `i` that iterates from 0 to 9 (these values are produced using `range(10)`). Then, the lines of code following this expression are evaluated ten times via this iterative process. One very important point to make about for loops: Only the indented lines immediately following the for loop declaration get repeatedly evaluated.

Question 3 (2 points): Please use for loops to generate via two different code snippets:

```
Stranger Things Season 3 was totally awesome!
Stranger Things Season 3 was totally awesome!
Stranger Things Season 3 was totally awesome!
I cannot wait for Stranger Things Season 4.
```

and

```
Stranger Things Season 3 was totally awesome!
Stranger Things Season 3 was totally awesome!
Stranger Things Season 3 was totally awesome!
I cannot wait for Stranger Things Season 4.
I cannot wait for Stranger Things Season 4.
I cannot wait for Stranger Things Season 4.
```

Please provide your source code.

3.4 Functions

One important feature of Python is the definition of functions, which can be used to help simplify coding if there are multiple instances where repetitive operations are performed. The way to specify a function is to use the keyword `def` followed by the function details. For instance, if I wanted to create a function called `foo` that took as input an integer `x` and produced an output that is equal to the square of the input, we would have the following:

```
def foo(x):
   return x**2

print(foo(3))
```

Note that the same indentation rules apply for functions as they do with respect to for loops; the body of the function must all be indented immediately after the declaration of the function.

Question 4 (2 points): Using a for loop and creating your own function, write a snippet of code that would generate an integer sequence ranging from -4 to 4 based on loop index values ranging from 0 to 8.

3.5 Objects and Classes

Objects are a useful approach for representing variables and functions within the same entity. In Python, objects are defined using classes, which are defined at the top of the program. Here's an example of using classes to define an object `Student` that can then be used in an application:

```
# Defining the class "Student"
class Student:
```

```
    def __init__(self, name):
        self.name = name

    def change_name(self, new_name):
        self.name = new_name

# Running some sample code using the class "Student" defining an object "student"
student = Student("Bob")
print(student.name)

student.change_name("Jill")
print(student.name)
```

Let's go over some of the important features of this short code snippet:

- The name of the class is `Student` and we can instantiate any object defined by this class with the variable `name`.
- The definition of `__init__` is the initialization function for any object defined by this class. In this case, we initialize variables used by the object defined by this class, such as the variable `name`.
- The use of the keyword `self` is a placeholder for the actual name of the object. For example, if we specify `student = Student("Bob")`, what happens is that `self` refers to `student`. As a result, the object `student` contains a variable `student.name`.
- We can also define other functions in the object for a variety of purposes, such as changing the name stored in the object. For example, if we have a function `change_name`, it can be used to replace the existing name in the object with a new name.

Although creating objects using classes is an extremely powerful resource when programming in Python, it is also very important to understand how these classes and objects are defined in the first place in case you are using someone else's source code and they included their own classes and objects.

Question 5 (3 points): Write a class `Foo2` that defines an object `foo_py` which consists of the initialization function and two other functions, `subtract_one` and `multiply_three`, where `subtract_one` subtracts one from the integer value stored in the object and `multiply_three` multiplies the stored value by three. The initialization function initializes the variable `value`. Demonstrate this object by initializing with the value 10 and then subtracting 1 from it followed by multiplying the result by 3.

4 NumPy, SciPy, and Matplotlib

In the world of communication systems, Python is widely used for programming computer simulations of wireless networks, implementing algorithms for use on software-defined radios, processing wireless signal strength measurements collected by a wireless sensor, or running machine learning frameworks

to automate the operations of an intelligent radio transceivers. However, almost all of these implementations use pre-existing libraries and packages of well-known functions and operators, such as convolution, filtering, Fourier transforms, and many others. In this course, three libraries you will be using extensively throughout the term are:

- **NumPy**: Extensive array and matrix handling support in addition to a variety of advanced mathematical functions.
- **SciPy**: Supports scientific and technical programming via a collection of advanced functions designed for applications such as optimization, image/signal processing, linear algebra, and differential equations.
- **Matplotlib**: Plotting library capable of producing effective GUIs and other graphical presentations of data across a range of different formats.

In this course, we will be frequently using the NumPy array as the primary data structure for storing signal data from our communication systems. NumPy arrays efficiently store information, and enable a wide variety of scientific computation. These array consist of pointers to memory, along with metadata used to interpret the data stored there, notably 'data type', 'shape' and 'strides'. The data type describes the nature of elements stored in an array. An array has a single data type, and each element of an array occupies the same number of bytes in memory. Examples of data types include real and complex numbers (of lower and higher precision) as well as strings.

The shape of a NumPy array determines the number of elements along each axis, and the number of axes is the dimension of the array. For example, a vector of numbers can be stored as a one-dimensional array of shape N, whereas colour images are three-dimensional arrays of shape $(M, N, 3)$.

Question 6 (2 points): Write a code snippet that concatenates three NumPy arrays, where the first array is equal to seven 0 values, the second array is equal to five 1 values, and the third array is equal to seven 0 values. Consider using the `numpy.concatenate` function. Print out the result of the concatenation.

Question 7 (2 points): Using the result from Question 6, convolve the result array with itself using the definition of the convolution. Print out the result of the convolution.

Question 8 (2 points): Repeat Question 7 but use the SciPy `signal.convolve` function.

5 Time and Frequency Representations of Sinusoidal Signals

Being capable of representing signals in both the time domain and frequency domain is critical with respect to studying the behavior of different communication systems. Consequently, there are several functions that will prove to be useful throughout this course when working on all the projects, namely:

- `numpy.cos` and `numpy.sin` are NumPy arrays of cosine and sine waveforms whose frequency can be specified as an input parameter.

- `matplotlib.plot`, `matplotlib.xlabel`, and `matplotlib.ylabel` are plotting functions that can be employed to produce useful time domain and frequency domain graphs of signals.
- `numpy.fft` is the Fast Fourier Transform (FFT) function available from NumPy. This function can be used to transform a time domain waveform into its frequency domain representation.

Question 9 (3 points): Generate a signal that is the sum of two sine waves with frequencies 50 Hz and 700 Hz, respectively. Plot the result in time domain. Assume a sampling frequency of 44.1 kHz.

Question 10 (3 points): Using your response in Question 9, plot the frequency response.

6 Writing a Jupyter Notebook

Sharing software source code with other individuals is always a challenge since the information is not always presented in a format that enhances readability. Consequently, Project Jupyter produced a web-based interactive interface call Jupyter Notebook that allows individuals to share Python source code and results in a format that is more human readable.

To create a Jupyter Notebook, select " File –¿ New File" in VSCode such that a new file is created. Then, save that file and provide it a filename that has the extension `.ipynb`. Once saved, VSCode will automatically recognize the file format as a Jupyter Notebook and configure the environment such that you can start using it within editor. The result of this process should look like the window shown in Figure 1.

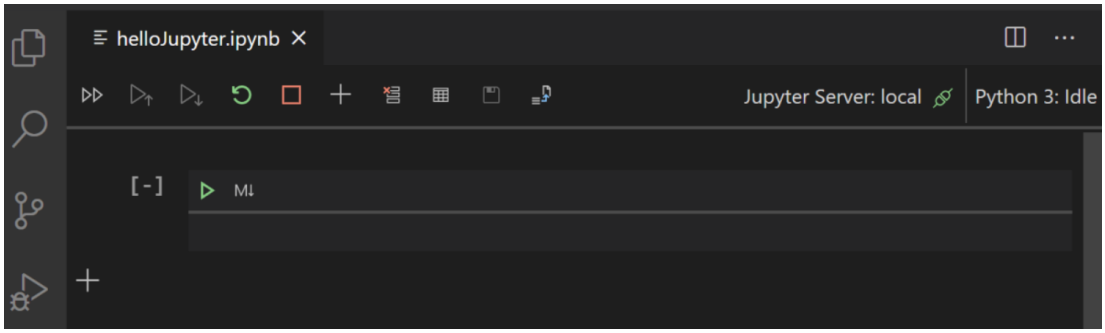

Figure 1: Screen capture of a Jupyter Notebook in the VScode development environment.

Once the Jupyter Notebook has been created and the environment configured, you can start using it to store your Python code snippets. For this project, the code snippets from each question should be written to its own "cell", which allows for the logical separation of the source code. Additionally, once the code has been written to a cell, it can also produce any outputs accompanying it, whether it is a terminal output or a plot generated by Matplotlib.

One critical part of a successful Jupyter Notebook is the commenting needed throughout the code to ensure proper documentation. All comments will appear in the Jupyter Notebook, so having enough text to accompany your code will be important to describe what your software is doing, defining all variables, and highlighting important stages in the program execution.

7 Report Submission

For each project in this course, the report submission to be uploaded to the ECE3311 Canvas website will consist of a single comprehensive Jupyter Notebook (.ipynb) file and nothing else unless it is requested. For this project, only the Jupyter Notebook is to be submitted electronically by the due date. Failure to submit the Jupyter Notebook by the specified due date and time will result in a grade of "0%" for the project.

Several important items to keep in mind when preparing your submission:

- Include the course number, project team number, names of team members, and submission date at the top of the Jupyter Notebook.
- Make sure your source code is thoroughly documented such that it is made clear what exact the program is doing line-by-line. Adequate commenting is worth **5 points** of the total score for this project.
- Responses to all questions indicated in the project handout. Please make sure that the responses are of sufficient detail.