Introduction to Electrical and Computer Engineering Design ECEG 201

Session 11 Lecture Notes

Announcements

1. Reading:
   (a) The Python Tutorial
      i. Section 3, *An Informal Introduction to Python*
      ii. Section 4, *More Control Flow Tools*
2. Second quiz on 2020-03-02, covers homework through Homework 6
3. Final exam is 2020-05-01, 3:30
4. Homework 6 is due on 2020-02-28
5. Supplemental materials:
   (a) Think Python 2e by Allen Downey
   (b) Learn to Program with Python 3 by Irv Kalb
   (c) The Quick Python Book by Naomi R. Ceder
6. Grades have been uploaded to Moodle, let me know if there are errors

$$CourseGrade = ((0.2 \times Homework) + (0.3 \times Labs) + (0.3 \times Quizzes))/0.8$$

Purpose of this course

From **Criteria for Accrediting Engineering Programs**, ABET Engineering Accreditation Commission, November 2, 2018:

> "2. an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors"
>
> "3. an ability to communicate effectively with a range of audiences"
>
> "6. an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions"
>
> "Student performance must be evaluated."

Doing your own work

From the Lab 4 Handout:

Each student will work individually *on this activity. You may* ***discuss*** *the activity with other students but you must* ***do your own work*** *and the report you submit must be entirely your own work.*

From the course syllabus:

As a general rule, anything that you submit for grading must be your own original work. You must properly cite the source of ***any*** *material that you copy, quote, or paraphrase from* ***any*** *other source. This includes, but is not limited to, text, data,* ***code****, or graphic material from* ***another student****, the instructor's lecture notes, a textbook, or the internet.*

Your work ***must not*** *be publicly accessible either during or after a laboratory session. Your programming for this class should be created and maintained in your private netspace folder.* ***Making your code available to another student, either directly or indirectly, will be considered academic misconduct.***

What is Python?

Python is an **interpreted** programming language.

Python is defined as an **open standard** by the Python Software Foundation.

We will use **Python 3**.

Development support for Python 2 ended earlier this year. When you start reading about Python, make sure you are reading about Python 3. One big clue is how the print statement is used:

```
print "I don't like spam!"                # Python 2
print("It doesn't have much spam in it!") # Python 3
```

Running Python

Ways to run Python programs:

- In a cell in a Jupyter notebook
- In Spyder IDE
- On the Feather board via Mu editor
- At the command line in a shell

  You can use the PowerShell in Windows to run Python programs. For the computers in the ECE labs, right-click on the Windows "start" icon in the lower left corner of the screen, then select `PowerShell`.

  ```
  PS U:\> cd private
  PS U:\> C:\Anaconda3\python.exe mypythonprogram.py
  ```

Ways to run Python interactively (REPL):

- In Spyder IDE
- On the Feather board via Mu editor
- At the command line in a shell

  You can use the PowerShell in Windows to run Python programs. For the computers in the ECE labs, right-click on the Windows "start" icon in the lower left corner of the screen, then select `PowerShell`.

  ```
  PS U:\> cd private
  PS U:\> C:\Anaconda3\python.exe
  ```

Objects in Python

All data in a Python program is in the form of **objects**.

Every object has a

- name or identity

  The name of an object never changes after it is created. Once an object can no longer be used by a program (for example, after a function ends, any object created inside that function) the Python **garbage collector** may destroy it. "Destroying" an object just means that the memory space that was used for it is now made available for some other purpose; this memory space is not necessarily cleared or erased.

- type

  The type of an object defines what kind of values it may have, and what kind of operations can be done on it.

- value

Python numerical data types

In Python, the **size** of an integer is limited by the amount of memory available.

For practical purposes, integers in Python will never **overflow** or **roll over**.

The actual size of an integer value changes at run time.

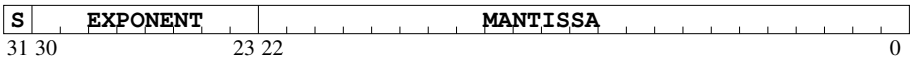
Some versions of CircuitPython do not support arbitrarily long integers. CircuitPython on Express boards **does**.

By default, Python floating-point values are **double precision**.

The IEEE Double-Precision Floating Point number…

- Has a range of roughly 10^{-308} to 10^{308}
- Has a resolution of roughly 16 decimal digits
- Includes $+0$ and -0
- Includes $+\infty$ and $-\infty$
- Allows **denormalized** numbers (very small but lower resolution)
- Defines a **NAN** (Not A Number)
- Is defined in IEEE Std 1754

The IEEE Single-Precision Floating-Point Format:



- Uses a 32-bit word
- One **sign** bit, 1 means the mantissa is negative
- An 8-bit **exponent**
  - In excess-127 format
  - `0x00 = -127,  0x7F = 0,  0xFF = 128`
- A 24-bit **mantissa**
  - In **unsigned** binary
  - **Normalized** with implied MSB

$$Value = -1^{S} \times \left(1 + \frac{\texttt{MANTISSA}}{2^{23}}\right) \times 2^{\texttt{EXP}-127}$$

The IEEE Single-Precision Floating-Point number…

- Has a range of roughly 10^{-38} to 10^{38}
- Has a resolution of roughly 7 decimal digits

CircuitPython floating-point values:

- Have a 22-bit mantissa and an 8-bit exponent
- Has a range of roughly 1.7×10^{-38} to 3.4×10^{38}
- Has a resolution of roughly 5 or 6 decimal digits

Python Numerical Operators

Addition, subtraction, and multiplication

- If either operand is floating, the other operand is converted to floating.
- If both operands are integers, an integer operation is performed.

Division (`/`) and floor division (`//`)

- Simple division always returns a floating-point quotient.
- Floor division of integers returns an integer.

  Conceptually, the floor division operation performs a floating-point division and then applies the **floor** operator to the quotient. The floor operator returns the largest integer value that is not greater than its operand.

Other data types

The `None` data type is used to indicate the lack of a value.

If you try to use a return value from a function that doesn't return anything you will get a `None`. Since `None` is a data type and not a value, you should use `is None` or `is not None` rather than `== None` or `!= None`.

The `bool` data type is a subtype of integers.

There are only two objects of this type, `True` and `False`. They behave like the integer values 1 and 0, respectively. The `None` data type is considered to be false.