CMSC201
Computer Science I for Majors

Lecture 19 – Modules and “Random” Numbers
Last Class We Covered

• What makes “good code” good
  – Commenting guidelines

• Top down design

• Code implementation
  – Bottom up
  – Top down
  – Incremental development
Any Questions from Last Time?
Today’s Objectives

• To learn about Python’s Standard Library

• To understand modules and importing
  – Syntax
  – Purpose

• To learn about “random” numbers
  – Pseudo randomness
Python’s Standard Library

• The “standard library” is made up of two parts

• The “core” of the Python language
  – Built-in types and data structures (int, list, etc.)
  – Built-in functions (`min()`, `max()`, etc.)

• Optional *modules* the programmer can import
  – Math things like `fractions` and `random`
  – Useful pieces like `datetime` and `calendar`
Modules
Modules

• A *module* is a Python file that contains function definitions and other statements
  – Named just like a regular Python file:
    ```python
    myModule.py
    ```

• Python provides many useful modules for us
• We can also create our own if we want
Importing Modules

• To use a module, we must first import it

• Where does Python look for module files?
  • In the current directory
  • In a list of pre-defined directories
    – These directories are where libraries like random and calendar are stored
Importing
Importing Modules

- To import modules, use this command:

  ```python
  import moduleName
  ```

- This imports the entire module of that name
  - Every single thing in the file is now available
  - This includes functions, data types, constants, etc.
import

• To use the things we’ve imported this way, we need to append the filename and a period to the front of its name ("moduleName.")

• To access a function called function:

  moduleName.function()
Calendar Module Example

```python
import calendar
exCal = calendar.TextCalendar()
printCal = exCal.formatmonth(2016, 11)
print(printCal)

    November 2016
 Mo Tu We Th Fr Sa Su
  1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30
```
“Random” Numbers
Random Numbers

• Random numbers are useful for many things
  – Like what?
  – Cryptography
  – Games of chance
  – Procedural generation
    • Minecraft levels, snowflakes in Frozen

• Random numbers generated by computers can only be *pseudo* random
Pseudo Randomness

• “Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.” – John von Neumann

• Pseudorandom appears to be random, but isn’t
  – Mathematically generated, so it can’t be
  – Called a Random Number Generator (RNG)
Seeding for Randomness

• The RNG isn’t truly random
  – The computer uses a “seed” in an attempt to be as random as possible

• By default, the seed is the system time
  – Changes every time the program is run

• We can set our own seed
  – Use the `random.seed()` function
Seeding for Randomness

- Same seed means same “random” numbers
  - Good for testing, allow identical runs

```python
random.seed(7)
random.seed("hello")
```

- 7 always gives 0.32, 0.15, 0.65, 0.07
- “hello” always gives 0.35, 0.66, 0.54, 0.13
Seeding with User Input

• Can allow the user to choose the seed
  – Gives user more control over how program runs
    `random.seed(userSeedChoice)`

• Can also explicitly seed the system time
  – Give the `seed()` function `None` or nothing
    `random.seed(None)`
    `random.seed()`
Generating Random Integers

- `random.randrange()`
- Works the same as normal `range()`
  - Start, stop, and step

```
>>> random.seed("dog")
>>> random.randrange(2, 21, 4) 14
>>> random.randrange(2, 21, 4) 6
>>> random.randrange(2, 21, 4) 10
>>> random.randrange(2, 21, 4) 10
>>> random.randrange(6) 5
>>> random.randrange(6) 4
```
Generating Random Floats

- `random.random()`
- Returns a random float from 0.0 up to (but not including) 1.0

```python
>>> random.seed(201)
>>> random.random() 0.06710225875940379
>>> random.random() 0.3255995543326774
>>> random.random() 0.0036753697681032316
>>> random.random() 0.28279809896785435
```
Generating Random Options

- `random.choice()`
- Takes in a list, returns one of the options at random

```python
>>> dogs = ['Yorkie', 'Xolo', 'Westie', 'Vizsla']
>>> random.seed(11.2016)
>>> random.choice(dogs)
'Xolo'
>>> random.choice(dogs)
'Westie'
>>> random.choice(dogs)
'Vizsla'
>>> random.choice(dogs)
'Westie'
```
How Seeds Work

• “Resets” the random number generator each time it is seeded

• Should only seed once per program

• Seeding and calling gives the same number

```python
>>> random.seed(3)
>>> random.random() 0.23796462709189137
>>> random.seed(3)
>>> random.random() 0.23796462709189137
```
Time for LIVECODING!!!
Generating PINs

• Write a program that stores usernames and their PINs in a dictionary

• Ask the user for their username
  – If it exists, tell them their pin code
  – If it doesn’t exist, create one using random
    • Tell the user what their new temporary pin is

• Pin should be between 0000 and 9999
Announcements

• Project 1 is due Wednesday
  – It is much harder than the homeworks
  – No collaboration allowed
  – Start early
  – Think before you code
  – Come to office hours