Introduction to:
Computers & Programming:
Review prior to 1st Midterm

Adam Meyers
New York University
Summary

• Some Procedural Matters
• Summary of what you need to Know
  – For the Test and To Go Further in the Class
• One Practice in-class writing of program
  – I will give everyone 15 minutes to write the same basic program independently
Procedural Matters Regarding the Midterm

• The class will be proctored by 1 or 2 people (depending on class size).
• If possible, there will be at least one empty seat between students.
• I will take attendance – please bring your School ID.
• The test will be graded on a curve.
• Each midterm counts for 20% of the final grade.
• If you are auditing and want to take the midterm, please let me know in advance and I will try to figure something out.
Type of Questions that Could Be on the Test

• Given a Python Expression
  – Determine what value will be returned
  – Determine what side effects will occur, e.g., what will be printed out

• Given some lines of code
  – Find the errors
  – State how to correct the errors

• Write a function to implement a problem that is about the same complexity as the Biorhythm program from last week
What you should know

• Understand basic concepts: *algorithm*, *program*, *programming language*, *function*, *operator*, *input*, *output*, *side effect*, *variable*, *scope*, *data types*, etc.

• Know how to:
  – Define functions
  – Assign values to variables
  – Return values from functions
  – Use print statements, input statements, operators
  – Import modules
  – Use data types and coerce one data type to another
  – Write a simple function that works and is easy to understand, due to comments and variable/function names
Algorithms and Programs

• An Algorithm
  – step by step plan for solving a problem

• Program
  – Executable implementation of algorithm, written in a computer language

• Programming Language
  – Formal language for writing computer programs

• Python
  – High level computer language
  – Popular for teaching and for writing not-too-large programs
More on Algorithms

– Pseudo Code: a series of ordered statements
  • Structured using line numbers, indents, bullets, etc.
  • Connected by logical and temporal connectors
    – if, else, unless, not, until, when

– Flow Chart: Connected Series of Boxes
  • circles/ovals = start/end
  • Rectangles = steps in processing
  • Diamonds = Decisions
  • Arrows = Sequence of Steps
Functions

• Programming language Functions have 3 optional features:
  – Input
  – Output
  – Side effects

• `print` versus `return`
  – `print` is significant for its side effect—printing to the computer screen
  – `return`
    • Exits block (function)
    • Provides a value to a function (maps to the domain)
Data Types

- Floats and Integers
- Strings
- Boolean (True or False)
- Nonetype (Output of void functions, like *print*)
Strings

• Components
  – Two identical delimiters: “, ', ""
  – The Characters in between the delimiters
  – Examples:
    • '!@##$^&*()'
    • “This is an apostrophe: ' “
    • 'This is a double quote: “ ' 
    • ""Triple quotes can include ' or “ or between them"

• Know how they look printed out, i.e., experiment printing different strings
Numbers

- **Integers:**
  - No Decimal Place
  - Not Too Big

- **Float**
  - Limited in Length
  - Used for numbers with decimals
  - Approximations using Scientific Notation

- **Normal Division with Integer Input**
  - Output can be integer or float

- **Integer Division (\//)**
  - Input/Output are integers (output is floor of answer)

- **Import Math library for many special functions/variables**
Type Conversion Functions

- **Float**
  - Converts Integers and compatible strings to floats

- **Str**
  - Makes a string out of any type of object (using definition of that object)
  - The *print* function prints strings to the screen without delimiters

- **Int**
  - Converts floats (by truncation) to integer
  - Converts compatible string to integer
Operators

• Know all the mathematical operators and what they do: +, -, *, **, /, //, %

• Be familiar with the two equal signs
  – The assignment operator =
  – The test for equality operator ==

• Understand how most of the operators can be restated as functions

• Note that the assignment operator = cannot be simulated as a function
Making Code Readable

• Comments
  – ## Know How to use comments
  – ## Know Why to use comments

• Naming Variables and Functions
  – Choosing names that are self explanatory
Identifiers

• Functions
  – How to define functions
  – Legal names for functions
  – Using colon, parentheses and indents

• Variables
  – Legal names of variables
  – Scope
    • global versus local
    • Python defaults regarding 2 variables with the same name, but different scope properties
  – When a variable is passed as an argument of a function, does the function use the actual variable or its value?
Practice: In-class Exercise

• Everybody implement the following function independently.
• You are writing a function for use with a pair of binoculars with a zoom lens. The Lens allows for magnification from 10 times to 100 times.
• The user enters the magnification M to be used
  – To multiply the apparent width and height of any object in view by a factor of M
• The user enters the current relative size of the object on a scale from 1 to 1000
  – 1 is $1/1000^{th}$ of the area of view and 1000 is the entire field of vision. The object can be larger than the field of vision.
• The program outputs the new relative size.
The Midterm is in 2 Classes

• That means that you have a whole week
  – To prepare as you see fit
  – To ask questions to lab tutors, the email tutor and myself

• Next Class
  – You will have opportunities to discuss problems
  – You have a few days to figure out what those problems are

• Good luck!