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

• Section I: Answer questions about Code
  – What is printed out?
  – What is the value of a variable after code executes?

• Section II: Find and correct errors in code
  – Identify errors
  – State how to correct the errors

• Section III: Write functions that solve simple problems involving: user input, printing, calculating values, etc.
What you should know

• Topics: *algorithm, program, programming language, function, operator, input, output, side effect, variable, scope, data types, simple for loops, 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`
String Components

• Two identical delimiters: ““, ’, ""

• The Characters between delimiters including:
  – digits (0-9)
  – letters(a,b,c,..z,A,B,C,...,Z, space)
  – escape characters: \n, \t, \\

• Sample Strings:
  – '!@##$^&*()'
  – “This is an apostrophe: ' “
  – 'This is a double quote: “ '
  – ""Triple quotes can include ' or “ or between them""
More on Strings

• When working with strings they are represented so it is easy to see all their components:
  – 'The quote \' and the newline \n are useful'

• When printed, a string will be displayed in a way that interprets these components.
  – The quote ' and the newline are useful
  – The delimiters are eliminated and escape characters are interpreted.
Numbers

- **Integers:**
  - No Decimal Place
- **Float**
  - Limited in Length
  - Used for numbers with decimals
  - Approximations using Scientific Notation
- **Normal Division with Integer Input**
  - Output is a float
- **Integer Division (**//**)**
  - Input/Output are integers (output is floor of answer)
- **Import Math library for many special functions/variables**
Type Conversion Functions (Numbers)

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

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

- Converted Strings can participate in math operations
  - $5 \times \text{int('5')}$
  - $20 / \text{float('5.5')}$
Converting Non-Strings to Strings

• \( \text{str}(5.55) \)
  – '5.55'

• Makes a string out of any type of object (using definition of that object)

• Once converted, non-strings can be combined with strings through concatenation
  – 'The number is '+\( \text{str}(5) \)
  – output = 5+100
  – 'The sum of 5 and 100 is '+ output
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?
for Loops

• The first line of a for loop:
  – Is of the form
    • for VARIABLE in SEQUENCE:
  – VARIABLE refers to some variable name (item, character, number, etc.)
  – SEQUENCE refers to a sequence (range(5), 'hello', etc.)
• The body of the loop is indented directly beneath the first line
  – The body repeats one time for each element in the sequence
  – On each iteration, VARIABLE is set to the next item in the sequence.
Sample for loop

• def diagonal_print(word):
  
  number = 0

  for letter in word:
    print(number*' ',letter,sep='')
    number=number+1

• Try it in IDLE
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 10000
  – 1 is 1/10000th of the area of view and 10000 is the entire field of vision. The object can be larger than the field of vision.
• The program outputs the new relative size.
Reminder: A Test is a Game

• Unfortunately, tests are imperfect because some people know how to play better than others.

• How to win the test game
  – Study sample test instructions
  – Time is a crucial factor (you have just over 1 hour)
  – Do easy problems before hard ones
  – Do not spend a lot of time on a low-point problem
  – Do not get stuck on details that you don't need
    • Example: On practice test, it is not always necessary to understand the program to locate bugs. It takes less time
  – Go for partial credit on program questions (most points)
    • If you cannot program some detail – write pseudo code
    • Basic solution strategy is more important than perfect syntax
The Midterm is in 2 (or more) 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!