Introduction to: Computers & Programming: Review for Summer 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
- Two Practice in-class writing of programs
  - I will give everyone 15-25 minutes to write the same programd independently

Procedural Matters Regarding the Midterm

- The test is on Monday, June 13, 2011
- There should 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
- The midterm counts for 33% of the final grade
- If you are auditing you can take the midterm if you want, but you do not have to.

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
  - Note: This type of question will not be on the final.
- 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
  - If/elif/else statements and basic decision trees
  - Create graphics with the turtle module
  - Basic *for* loops

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 call – Example:
      - If: function1(a) returns 5 and function2(b) returns 10
      - Then: function(a) + function(b) = 15

Data Types

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

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 * int('5')
  - 20 / float('5.5')

Converting Non-Strings to Strings

- 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 '+str(5)
  - output = 5+100
  - 'The sum of 5 and 100 is '+ output

Arithmetic Operators & the Assignment Operator

- 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

Boolean Operators

- **and, or, not**
  - True and True → True, True and False → False, False and True → False, False and False → False
  - not(True) → False, not(False) → True
  - False or False → False, True or False → True, False or True –> True, True or True → True
- **<, >, <=, >=** – Expected meanings from math
- **==, !=** – 'is equal to', 'is not equal to'
- **in** – currently, we only see this in *for* loops, but it is a Boolean operator that tests for membership, e.g.,
  - 'a' in 'abcdefg' → True, 5 in range(4) → False

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?

If/elif/Else Statements

- Syntax (elif and else parts optional)

```
If + boolean-expression + :
    body
elif + boolean-expression + :
    body
else:
    body
```

- Example:

```
def classify_integer(integer):
    if integer==0:
        return('zero')
    elif (integer%2) == 0:
        return('even')
    else:
        return('odd')
```

Basics of Turtle Graphics

- Loading module, creating a screen and a turtle

```
import turtle
my_screen = turtle.Screen()
my_turtle = turtle.Turtle()
```

- Basic Turtle commands
  - my_turtle.fd(distance)
  - my_turtle.left(degrees)
  - my_turtle.pd()
  - my_turtle.pu()

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

Basics of Planning a Program

- Figure out the big steps first and “pretend” that you have already written most of the functions.
- Then write the functions that you assume that you need. Writing these functions may involve pretending that more functions exist, but that is OK.
- Repeat process until you don't have to pretend anymore because the functions are simple enough to just write and be done.

In-class Exercise 1

- 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.

In-class Exercise 2

- Write a function that takes two parameters: a string and a target character
- Use a for loop to check each character in the string and see if it matches the target character. Count the number of characters that match.
- Return the total number of matches, i.e., the number of times that the target character occurs in the input string.

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 1 hour and thirty five minutes)
  - 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 Next Week

- That means that you have time
  - To prepare as you see fit
  - To ask questions
  - To try out the practice midterm
  - To try examples from previous midterms
- This Week
  - You will have opportunities to discuss problems
  - You have time to figure out what those problems are
  - We will go over the practice midterm later in the week
- Good luck!