Resources for those Skipping CSC/CPE 101

Last updated: November 22, 2016

1 Overview

This document and the accompanying website are meant to provide, for those students able to skip CSC/CPE 101, some context and resources for preparation for CSC/CPE 202. Though the goal of CSC/CPE 101 is to introduce students to the thought-process and mechanics of introductory programming, it is quite common for students (and some faculty) to define the course by the language used (i.e., Python). CSC/CPE 101 is not, by any meaningful measure, a course on the Python programming language. That said, students completing CSC/CPE 101 will have gained some valuable knowledge of and experience with Python and CSC/CPE 202 will expect that students are familiar with programming and programming in Python.

More significantly, and of more general importance, CSC/CPE 101 should introduce the conceptual tools for problem solving in a computing context. This includes problem decomposition (iteratively breaking a problem into smaller subproblems until the individual problems can be solved and then composed into a solution to the whole), the expression of problem solutions as function, testing to determine if a proposed solution satisfies stated requirements, recognizing and applying common programming patterns, and reasoning about boolean algebra (and, or, not). That said, it is assumed that you are comfortable with such topics and able to demonstrate such skills based on your prior experience. As such, the remainder of this document and the provided resources focus on the Python programming language (though there is explicitly no focus on being "Pythonic").

2 Course Outline

The language related material of CSC/CPE 101 is split into multiple subsections. In each subsection, the general topic is discussed and sample code is provided, with further details found elsewhere.

CSC/CPE 101 has, thus far, used Python 2.7 (technically 2.6 is installed on the lab machines) but the course will move to Python 3. Even so, for the language features covered in CSC/CPE 101, the language version is mostly irrelevant.

2.1 Formatting

Python treats leading whitespace characters (spaces and tabs) as significant. In particular, the leading whitespace is used to indicate when one statement is nested inside of another (examples will follow). As such, you must indent each level of code (e.g., statements within a function and statements within the "then" branch of an `it`). You are encouraged to use only spaces (no tabs) to avoid any accidental misalignments.

2.2 Data

Python directly supports many types of data values including integers, floating-points, and boolean values. The standard arithmetic and relational operators are supported for both integers and floating-points. The standard boolean operators (discussed below) are supported.

Variables in Python are introduced through assignment.

```
   a = 27
   b = a + 3   # b has value 30
```

2.2.1 Structured Data

CSC/CPE 101 does not focus on object-oriented programming, but objects are introduced to group data into an aggregate. This allows one to work at a higher conceptual level by developing problem-specific abstractions. The following example illustrates defining a class with an `__init__` function taking two arguments, `x` and `y`. Note that the first parameter `self` is explicitly listed, but implicitly passed (see the creation of a `Point` object in the last line). This first parameter references the object itself (analogous to `this` in Java), it must be listed, and it must be explicitly used in accessing and assigning to attributes (fields) within the object. Unlike in Java, there is no declaration of the attributes within an object (assignment creates the attribute).

```
# define a class to store x- and y-coordinates for a 2d point
class Point:
   def __init__(self, x, y):
      self.x = x
      self.y = y

# create an instance of the Point class
pt = Point(1, 2)
```

Aside from `__init__`, `__eq__` and `__ne__` for testing, and, possibly, `__str__` or `__repr__`, CSC/CPE 101 does not introduce the general definition of methods.

2.3 Functions

Functions are defined outside of classes. In CSC/CPE 101, a function takes a finite set of arguments and returns a single value (that single value may be a tuple). Note that a function that does not explicitly return will result in the `None` value being returned.

```
# define a new function taking a single argument
def square(x):
   return x * x

# function call
square(7)
```

Arguments are passed by value. Variables may references objects and these references are passed by value; as such, the contents (attributes) of an object may be modified within a function (the function parameter becomes an alias for the object), but be sure this is the desired behavior.

```
# change the position of the given point
def translate_mutate(pt, dx, dy):
   pt.x = pt.x + dx
   pt.y += dy        # similar to above using a compound assignment


# return a new object at the new position
def translate_clone(pt, dx, dy):
   return Point(pt.x + dx, pt.y + dy)
```

Write many little functions. They are easier to reason about, easier to test, and easier to reuse.

2.4 Logic and Conditionals

Python supports the `True` and `False` values as well as the `and`, `or`, and `not` operators to build interesting boolean expressions. You should be familiar with the meanings of these operators and DeMorgan's Law. Python also supports, in addition to `True` and `False`, the notion of "truthy" and "falsy" values; these are not emphasized in CSC/CPE 101, but may be of interest when reading existing Python code.

Due to Python's indentation requirements, multiple `if` statements can be chained used `elif` instead of nesting.

```
def example(a, b, c, d):
   if a > b:
      return a
   elif b > d:
      return b
   elif c > d:
      return c
   else:
      return d
```

2.5 Lists

Python provides a list data type to hold sequential data. This data type is very similar to an array in Java (and is implemented using an array). The majority of the work in CSC/CPE 101 focuses on the manipulation of lists.

```
   nums = [1, 2, 3, 4]
   nums[0]              # first element, value 1
   len(nums)            # count of elements, 4
   nums.append(77)      # adds to end
   nums                 # [1, 2, 3, 4, 77]
```

In addition, basic list comprehensions are useful for the very common operations of applying a computation to every element of a list (creating a new list) and filtering a list to create a new sublist.

```
nums = [1, 2, 3, 4]
squared = [x * x for x in nums]     # square has value [1, 4, 9, 16]

mixed = [5, 2, 9, 3, 7, 6, 4]
lesser = [x for x in mixed if x < mixed[0]]  # lesser has [2, 3, 4]
                                             # values in mixed less than first (5)
```

2.6 Iteration

Python supports a `while` statement and a foreach statement (`for`). The `for` is perfectly matched for the majority of iterations over a list. In particular, you should be comfortable iterating over the values within the list as well as iterating over the indices for a given list (i.e., using `range(len(nums))` to iterate over the indices for list `nums`).

2.7 Sorting

You are expected to be familiar with selection sort and insertion sort.

2.8 Strings

CSC/CPE 101 introduces basic string operations (slices are not required). In addition, representation of characters as a mapping from integers based on a standard is explored, though the specific mappings are not expected knowledge (i.e., the ASCII value of character 'a' is not assumed knowledge). Regular expressions are not a CSC/CPE 101 topic.

2.9 Exceptions

CSC/CPE 101 introduces the basic concepts of exceptions and exception handling primarily as they relate to File I/O.

2.10 File I/O

In CSC/CPE 101, students work with text files primarily on a line-by-line basis using `split` to partition the line into a list of "words". Further processing is on this list using techniques explored earlier.

Students are expected to write programs that will read from a file, process the contents (including errors in expected format), and write to another file.

3 Internal Resources

A website for those with an alternate 101 experience has been setup at `http://users.csc.calpoly.edu/~akeen/courses/alt_101`.

A set of representative Lab exercises is available on the Alternate 101 website. Working through these lab exercises, following or in conjunction with a Python tutorial, will allow you to gain an experience similar to that of CSC/CPE 101 but without the time required for doing the larger projects.

A Piazza forum (linked to from Alternate 101 website) is available for questions from those trying to make the transition from an alternate 101 experience to Python. You may post questions to this forum for discussion with others skipping CSC/CPE 101 (membership is entirely voluntary) and with at least one faculty member.

4 External Resources

Links to external resources for Python, Unix, and editors are available at `http://users.csc.calpoly.edu/~akeen/courses/alt_101`.