Agenda

• Administrative Items

• Lecture Recap

  • L01 - Intro to Python knowledge & fundamental concepts
Collaboration: don’t plagiarise. Write your own code.
  • All PSET submissions are checked against each other and against historical submissions.
• Halfway hand-in - submit some code before the deadline.
• Checkoffs occur at Office Hours - typically due 1 week after deadline.
• Late day policy
  • 3 late days total
  • 1 late day = 24hr extension
  • Late days are discrete (no half days)
  • Apply only to problem sets
• The last submitted pset is used for grading and late day calculation
• Submit on course website
• Link on course website
• Due before each lecture
• Lots of small, quick problems - one poor score won’t have much impact
• Starting with PS1, you need a checkoff for each pset (generally worth 30% of your overall pset grade)
• Go through your code with a TA or LA, and answer some simple questions about the pset. Score is based on code style and understanding of the pset code
• Carefully check due date of Checkoff for each p set
• Late days cannot be used for checkoffs
• The queue gets long around the checkoff deadline, so get them done early!
• **Anaconda** is a Python Distribution, which contains Python, a set of Python packages, a code editor (**Spyder**), and an interactive interpreter/shell (**IPython**)
• Spyder: Scientific Python Development Environment
  • A place to edit code, run it, and debug it
• We encourage this dev environment for this class
• **Make sure you run your code before you turn it in.**
Lecture 1 Recap: Intro to Python + Fundamental Programming Concepts

• **Python programs**
  • Set of instructions telling the computer exactly what to do.
  • Can be run from a script (e.g script1.py) or directly from the console
  • Each line of code is executed in the order it’s written in.
  • It’s good practice to write tidy code & comments.

• **Objects**
  • Programs manipulate data objects.
  • Typically define an object with a variable name (e.g my_name = “Nicole”)
  • Each object has a type (e.g. string, list, integer, float, boolean etc…)
  • Scalar objects cannot be subdivided.
  • Non-scalar objects have an internal structure that can be assessed.
  • The type defines what you can do with the object
• **Operations & Expressions**
  • Operations are carried out on objects (what operations are valid is controlled by object type)
  • Expressions are formed by a combination of operations and objects.
  • Complex & long expressions often evaluate to one value.
# declaring a simple string
basic_string = "My name is Nicole!"

# integers and simple operations
a = 1
b = 20
sum_ab = a + b
difference_ab = b - a

# example operations on integers and strings
my_numbers_as_string = str(9) + ' ' + str(10) + ' ' + str(7)