Comp 151

Control structures.
• first quiz this week
  - believe it or not – only 3 weeks from exam.
  - one a week each week after that.
Objectives

- To understand the programming pattern simple decision and its implementation using a Python `if` statement.
- To understand the programming pattern two-way decision and its implementation using a Python `if-else` statement.
Objectives (cont.)

- To understand the programming pattern multi-way decision and its implementation using a Python `if-elif-else` statement.
- To understand the idea of exception handling and be able to write simple exception handling code that catches standard Python run-time errors.
Objectives (cont.)

- To understand the concept of Boolean expressions and the `bool` data type.
- To be able to read, write, and implement algorithms that employ decision structures, including those that employ sequences of decisions and nested decision structures.
Simple Decisions

- So far, we’ve viewed programs as sequences of instructions that are followed one after the other.
- While this is a fundamental programming concept, it is not sufficient in itself to solve every problem. We need to be able to alter the sequential flow of a program to suit a particular situation.
Simple Decisions

- Control structures allow us to alter this sequential program flow.
- In this chapter, we’ll learn about decision structures, which are statements that allow a program to execute different sequences of instructions for different cases, allowing the program to “choose” an appropriate course of action.
Motivation

• up till now
  – written simple programs storing and manipulating small amounts of data.
  – first line in program executed, then each line after that.

• now add ability to take more control from computer
Making decision

• sometimes we don't want every line of code to execute
  – don't want to do the same thing every time
  – real life example?
Making decision

- sometimes we don't want every line of code to execute
  - don't want to do the same thing every time
  - real life example?
  - how about
  - if campus is open – go to class
- can do this in programming as well
  - technically called 'selection'
    - decision making
Selection in python

- syntax
  - if <something that evals to boolean> :
    - statements here
  - the <> brackets just indicate that whatever is between them is semantic and should be replaced
  - again note indent of statements
- example
  - value = True
  - if value:
    - print ("yes")
Boolean expressions

- laymans terms: something that evaluates to true/false value
- in python
  - True
  - False
- what sorts of things give true/false values?
Common booleans (in python)

- equality and inequality
  - == (equality)
  - >, <, >= (greater than or equal) <= != (not equal)
- straight negation
  - not
  - eg:
  - value = True
  - if not value:
    - print(“something”)
sometimes you want to do something else when the condition is false
  - if the campus is open go to class
  - otherwise go back to bed.

if today == 'Monday':
  - print ("Back to work")

else:
  - print("same old, same old")

else only after if.
code 'blocks'

- notice the indenting?
  - As you know indenting is how python determines code that logically belongs together
  - called a 'code block' in jargon terms
  - can have as many lines as desired in said code block
  - all indenting the same – same code block
multi line code block

- let's try one in pycharm.
multiple conditions

- can have multiple conditions
  - use elif keyword
  - if today == 'monday':
    - print ('blah')
  - elif today == 'friday':
    - print ('yup')
  - elif today == 'Sunday':
    - print('almost all over')
  - else:
    - print("same old same old")
Lets try a little
definite and indefinite loops

- two kinds of loops in programming
  - definite: know how many times to loop when program begins looping
  - indefinite
    - don't know how many times, but know how world will be when done
    - (ends on condition)
  - real world examples of each from class?
  - which is the range looping we did last week?
While: python indefinite loop

- Syntax
- `while <condition>:`
  - codeblock
- as long as the condition is true, codeblock will be executed again and again
- lets do one in idle
- stalker 'friend'
- then add ifs
potential loop problems I

• Infinite loop
  – you must update the loop variable somewhere in the code block for the loop.
  – i.e. the loop needs to have the chance to get closer to ending each time through.
  – bad:
    • counter = 1
    • sum = 0
    • while counter <=5:
      – sum = sum+counter
Potential loop problems II

- Off by one errors
  - want to make sure you loop the right number of times – not one too many or one too few
  - code below is supposed to be calculating how much a bank balance is after 10 years of 5% interest. what are problems?
    - balance = input(“how much to start?:”)
    - balance = float(balance)
    - years = 0
    - while years <= 10:
      - balance = balance + balance* 0.05
loop problems III

• problems in that loop:
  - infinite loop

• questions/problems
  - should years start at 0 or 1?
  - should the condition be < or <=
  - consider simple conditions
  - lets “hand execute” on board first.