CSCA08 Fall 2017

Week 4 - Selection

Brian Harrington & Marzieh Ahmadzadeh

University of Toronto Scarborough

September 25 – 29, 2017

Admin

- Test Runner: Run your own tests against your own code
- Extra FSGs this week: Details on Piazza
- Practice tool study: Details on Piazza
- MarkUs: Marks are in report file in your code repo

Term Tests!

- TT1 = This Saturday
- Rooms/details on course website
- Details:
    - Covering everything up to & including week 3 material (Design recipe)
    - Everything from lecture, tutorial, readings, exercises is fair game
    - Practical material = good study material
    - Testing understanding not memory
    - Previous tests on website
    - Know your tutorial #, student #, utorid (and bring your T-Card)
    - Read directions carefully

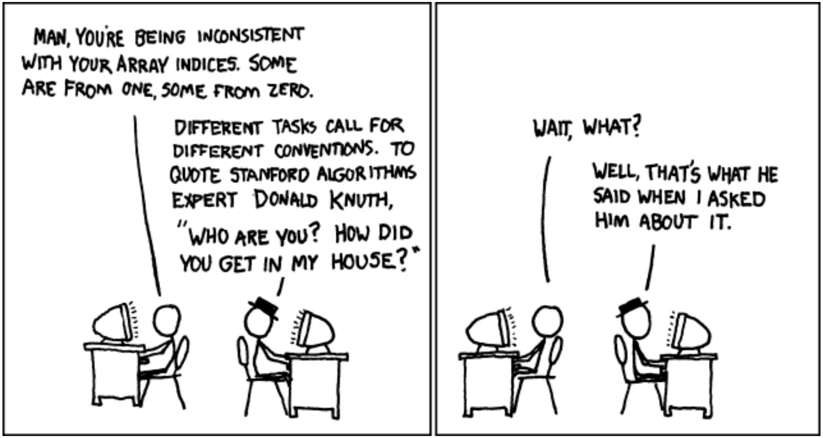
Strings

- Remember: To Python, these are just a series of symbols
- Lots of useful functions e.g.,: `len()`
- Also lots of useful methods e.g., `.upper()`
    - `function` - belong to a module, or built into python. Take variables as parameters.
    - `method` - belong to a type, operate on data of that type
- Can access individual letters in a string
    - e.g., `my_string[3]`
    - Note: We start counting at 0, so `my_string[0]` is the first letter
    - We cannot change individual characters in a string:
        - `my_string[3] = "C"` will crash
    - Have to be careful not to read past the end of the string
        - `my_string = "Hello"`
          `my_string[5]` will crash

Lists

- New Data Type!
- `list` is a sequential collection of values denoted by `[]`
    - `my_list = [1, 2, 3]`
- A lot of similarities to strings
- Can access individual members
    - `my_list[2]`
    - Still start counting at 0
- Unlike strings, we can change the elements of a list
    - `my_list[1] = 7`
- We can mix types in the list
    - `my_lists = [7, "Hello", True, [1, 2, 3]]`
- Lists are a **great** use-case for the memory model

Break


MAN, YOU'RE BEING INCONSISTENT WITH YOUR ARRAY INDICES. SOME ARE FROM ONE, SOME FROM ZERO.
DIFFERENT TASKS CALL FOR DIFFERENT CONVENTIONS. TO QUOTE STANFORD ALGORITHMS EXPERT DONALD KNUTH, "WHO ARE YOU? HOW DID YOU GET IN MY HOUSE?"
WAIT, WHAT?
WELL, THAT'S WHAT HE SAID WHEN I ASKED HIM ABOUT IT.

Adaptive Code

- Up to now, code has been linear
- Step 1 ... Step n. Finish
- Now we're going to learn how to write `adaptive code`

If Statement

- Allows us to execute code only if some condition is met
- General form:

  ```
  if condition:
      block
  ```
- `condition` is a boolean expression
- `block` is a series of python statements
- iff `condition` evaluates to `True`, then `block` is executed

If Statement

- Example:

```
if (grade >= 50):
   print("You passed!")
```

If-Else

- On previous slide: "iff `condition` evaluates to `True`, then `block` is executed"
- iff = "if and only if"
- if `condition`:

  ```
      block
  ```
- iff `condition` evaluates to `True` then execute `block`
  - if `condition` evaluates to `True` we will execute `block`, if it evaluates to `False`, we won't.

If-Else

- What if we want to execute different blocks of code based on the evaluation of a boolean expression?
- If-Else general form:

```
if condition:
    block1
else:
    block2
```

General If

- We can have more than 2 options
- elif = else if
- General if form:

```
if condition1:
    block1
elif condition2:
    block2
elif condition3:
    block3
else:
    block4
```

- "If condition 1 is true, execute block 1, otherwise if condition 2 is true execute block 2 ..."
- exactly 1 of the blocks will be executed