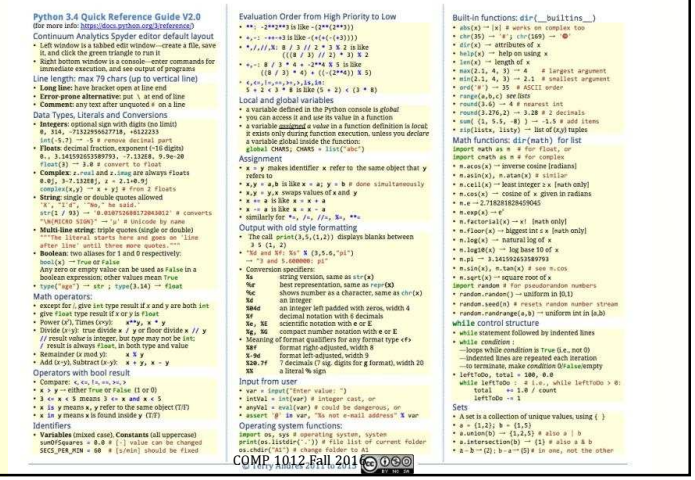
How to prepare for the exam

WEEK 13: REVIEW

Go Over the Python Guide

http://www.cs.umanitoba.ca/~comp1012/PythonGuide_V2.0.pdf


Python 3.4 Quick Reference Guide V2.0

Evaluation Order from High Priority to Low

Built-in functions: dir(__builtins__)

Final exam: 50 marks, 3 hours

- Approx 10 marks: Predict the output
  - more than half from QuizMaster
- Approx 10 marks: Multiple Choice
- Approx 5 marks: Debugging
  - find inserted defects in code similar to that in the notes or sample solutions
- Approx 25 marks: Programming
  - on paper
  - similar to code in programs on the website (e.g., in sample solutions, notes, sample code)
- You will get a copy of the latest Python Guide

Posted exams

- Links on website
  - Final 2013A (with answers)
  - Final 2013C (with answers)
  - Final 2014A (with answers)
  - Final 2014C (with answers)
  - Final 2015A (with answers)
- Don't spend ***all*** your time with old exams
  - Questions typically based on current assignments
  - Read/understand published solutions to assignments

Posted exams QuizMaster!

- Final 2015A (with answers)
- A. Predict output: average ≥ 60% on all questions
  - Except 3 (50%!) and 6 (30%)
  - Your place to pick up marks
- B. Big program: average ~ 60% overall
  - Practice programming on paper!
- C. Short programs: avg 50-60% overall
  - Review latest material, not covered by assignments
- D. Multiple choice: 50%!
  - hard questions 1, 2, 3, 6, 8 (especially 8)

Posted exams QuizMaster!

- Final 2014C (with answers)
- A. Predict output: average ≥ 60% on all questions
  - Except 9 (0%!) Your place to pick up marks
- B. Big program: average ~ 60% overall
  - Practice programming on paper!
- C. Short programs: avg ~60% overall
  - Review latest material, not covered by assignments
- D. Multiple choice: hard questions 1, 4, 6

Posted exams

- Final 2014A (with answers)
- A. Predict output: average ≥ 60% on all questions
  - Your place to pick up marks
- B. Big program: average ~ 50% overall
  - Practice programming on paper!
- C. Short programs: avg >60% overall; ~50% on C2
  - Review latest material, not covered by assignments
- D. Multiple choice: hard questions 4, 5, 8

Studying

- Studies have shown…
  - recall is better in the same environment under the same conditions that you learned the material
    - so reduce the differences between study conditions (posture, music, etc.) and exam conditions
    - better: ***study in several conditions and locations***
  - rehearsal: practise the activity you will be tested on
    - ***write out programs by hand***
    - if you did not get great assignment marks, ***review the posted solutions,*** not your own
- http://www.nytimes.com/2010/09/07/health/views/07mind.html?_r=1&pagewanted=all
  http://tinyurl.com/phjzf5p

Prime Directive

Read the Question!

How to prepare to write programs

1. Take some code (e.g., a function definition)
   - write a description of what it is supposed to do
2. Code a solution from the written description
   - on paper
   - without looking at the published solution
3. Give yourself a mark based on the published solution
   - if you are not sure if yours would work, try it
4. If your mark is not good enough, study the solution
   - go back to step 2
5. Change the description slightly and repeat

Example from a 2014 Winter Exam

MARKS

[3]

C.2 Simulate Dice [3 MARKS]

Write a script (it doesn't have to have any input or any functions) to estimate the probability that when you throw two fair 6-sided dice, the number of spots on the two dice differs by 1 (e.g., 4 and 3, or 1 and 2). Using either `random` or `numpy.random`, simulate throwing two dice 10000 times and estimate the probability from those results. Initialize the random seed first. Print the results like this:

```
Estimated probability two dice differ by 1 is 0.2791

Programmed by <your name>
```

```
# no leading comments are required
import numpy as np
import random
```


Example from a previous midterm

MARKS

[3]

C.2 Simulate Dice [3 MARKS]

Write a script (it doesn't have to have any input or any functions) to estimate the probability that when you throw two fair 6-sided dice, the number of spots on the two dice differs by 1 (e.g., 4 and 3, or 1 and 2). Using either `random` or `numpy.random`, simulate throwing two dice 10000 times and estimate the probability from those results. Initialize the random seed first. Print the results like this:

```
Estimated probability two dice differ by 1 is 0.2791

Programmed by <your name>
```

```
# no leading comments are required
import numpy as np
import random
```

Use a highlighter to pick out key parts

Published solution

```
np.random.seed(2014)
NUM_ROLLS =  10000
dice1  =  np.random.randint(1,7,NUM_ROLLS)
dice2  =  np.random.randint(1,7,NUM_ROLLS)
diff   =  np.abs(dice1 - dice2)
count  =  np.sum(diff  == 1)
print ("Estimated probability two dice differ by 1"
       + " is %0.4f"  % (float(count) / NUM_ROLLS))

print "Programmed  by <student  name>"
```

A. [1] generate dice, including seed
B. [1] count event
C. [1] calc and print result

What was your mark?

Single loop: while or for

- Examine each entry in a sequence one at a time
  - test them
  - count them
  - total them
  - add to the end of a list
- Can typically be replaced by a single numpy operation
  - ```
    numbers = np.arange(1,nn)
    div = nn % numbers == 0
    return np.sum(numbers[div])
    ```


Nested loops: while or for

- Examine each entry in a sequence in turn
  - generate and test another sequence
  - determine some outcome from that sequence
  - then test, total, count, ...
- If you get confused, use a function for the inner loop
  - ```
    for row in range(numRows) :
        printLine(row)
    ```
  - ```
    for num in range(numToExamine) :
        divisorCount = countDivisors(num)
    ```


Input

prompt

```
usrIn = input("Enter 3 numbers: ")
usrIn = input("Enter your first and last "
       "names: ")
```

- REMEMBER TO CONVERT STRING INPUT TO INT OR FLOAT
- Other conversions
  - get rid of extra space: `usrIn = usrIn.strip()`
  - convert to lower case: `usrIn = usrIn.lower()`
  - break into parts: `usrIn = usrIn.split()`
  - `usrIn = usrIn.strip().lower().split()`
```

Output

```
print ("%10s has %-5d fields averaging to %8.3g"
       % (recordId, numField, avgSize))
```

- How do you print a tuple?
- Other questions?

What to take

- Final exam:
  - Take ID (student card preferred) or a winning smile
  - Take at least 2 pencils and 2 **good** erasers and a highlighter
  - Take water (a gruelling test of endurance)
  - Take fruit/candy for quick energy
  - No aids: do NOT take a calculator, translator, iPod, cellphone, or any other electronic device
  - Your pack/books ***must fit under your desk***

My Best Wishes to You All

- I hope all goes well with all your exams
- I will be answering email between now and exam time
- Make appointments to see me; you may or may not find me in otherwise