Python Exercises
Introduction - Advanced

1 Find Errors

Learned Skills: basic python skills

Find four programming errors in the below example (if you find five, win a candy).

```
1  #!/usr/bin/ env python
2
3  import sys, random
4  def compute(n):
5      i = 0; s = 0
6      while i <= n:
7          s += random.random()
8            i += 1
9      return s/n
10
11 n = sys.argv[1]
12 print 'the average of %d random numbers is %g' % (n, compute(n))
```

2 Cartesian/Polar Coordinates

Learned Skills: Writing functions

Points may be given in polar (r, θ) or cartesian coordinates (x, y), see Figure 1.

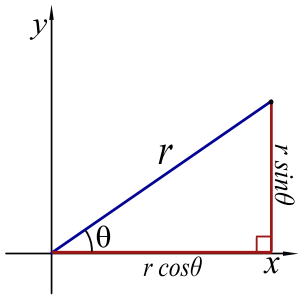

Figure 1: Relationship between polar and cartesian coordinates.

1. Write a function `pol2cart`, that takes a tuple (r, θ) in polar coordinates and returns a tuple in cartesian coordinates.

2. Write the inverse function `cart2pol`, such that `pol2cart( cart2pol( (x,y) ) )` is (x, y) for any input (x, y).

3. Extend the two functions, such that they can in addition handle lists of tuples.

3 Word counting

Learned Skills: File input/output

1. Create a script that opens a text file for reading and report the number of lines, words and characters. Assume that words are separated by whitespace (forget hifens!).

2. Open a text file for writing and save the count in it.

3. Extra: Choose a book from The Project Gutenberg (`www.gutenberg.org`, e.g., "The Merry Adventures of Robin Hood, by Howard Pyle") in text format and list the ten most frequently used words. You can use a Dictionary for counting.

Useful: `string.split()`, `open()`, `file.read()` and `file.write()`.

4 Party game

Learned Skills: Random numbers; Loops

One guessing game, called "squeezed", is very common in parties. It consists of a player, the chooser, who writes down a number between 00–99. The other players then take turns guessing numbers, with a catch: if one says the chosen number, he loses and has to do something daft. If the guessed number is not the chosen one, it splits the range. The chooser then states the part which contains the chosen number. If the new region only has one number, the chooser is said to be "squeezed" and is punished. An example of gameplay would be:

- Chooser writes down (secretly) his number (let's say, 30).

— Chooser: "State a number between 00 and 99."
— Player: "42".
— Chooser: "State a number between 00 and 42."
— Player: "26".
— Chooser: "State a number between 26 and 42."

⋮

— Chooser: "State a number between 29 and 32."
— Player: "31".

- Chooser dances some very silly children song.

Implement this game in Python, where the computer is the chooser.

Useful: `random.randint()` and `raw_input()`.

5 Rock-paper-scissors

Learned Skills: Random choices

Implement the game of rock-paper-scissors. Extra: make it rock-paper-scissors-lizard-spock (if you do not know it, Wikipedia should help).

Useful: `random.choice()`.

6 Dice Simulation

Learned Skills: Random numbers; Monte-Carlo Simulation

Estimate the chance of an event in a dice game. What is the probability of getting at least one 6 when throwing two dice? This question can be analyzed theoretically by methods from probability theory. However, a more general alternative is to let a computer program throw two dice a large number of times and count how many times a 6 shows up. Such type of computer experiments, involving uncertain events, is often called Monte Carlo simulation.

1. Create a script that in a loop from 1 to n draws two uniform random integers between 1 and 6 and counts how many times p a 6 shows up. Write out the estimated probability p/n together with the exact result 11/36. Run the script a few times with different n values and determine from the experiments how large n must be to get at least three decimals (0.306) of the probability correct. Use the random module to draw random uniformly distributed integers in a specified interval.

2. Generalize the script to an arbitrary number of dices, N.

3. Determine if you win or lose a hazard game. Somebody suggests the following game. You pay 1 unit of money and are allowed to throw four dice. If the sum of the eyes on the dice is less than 9, you win 10 units of money, otherwise you lose your investment. Should you play this game?

7 Run an external script

Learned Skills: Calling external applications; List handling

1. Write a script `runexternal.py` that runs the dice-simulation developed in the last exercise as an external program for a range of parameters (n and N). The parameters should be passed to the script as command-line parameters. Collect the result in a list that as each element contains a four-tuple (N, n, result, analytical result).

   Note: You could of course just put a loop into the monte-carlo script. However, calling it as an external application gives you the opportunity to put any program instead of the Monte-Carlo script (say a compiled simulation program you got from a collegue).

Use subprocess.call() for calling the script. Note that the subprocess module is most appropriate to learn because this module intends to replace several other, older modules and functions, such as: "os.system","os.spawn*","os.popen*","popen2.*" and "commands.*".

2. Output the result in a comma-separated (.csv) file.

3. Let `runexternal.py` take command line arguments that allow the user to specify ranges in MATLAB-style for which the external script should be called.

   E.g.

   ```
   $ python runexternal.py N=1:5 n=100:1000:10000
   ```

   should run the script with any combination of $N = 1, 2, 3, 4, 5$ and $n = 100, 1100, 2100, \ldots$.

   Use the **argparse** module to rewrite command-line parsing in `runexternal.py` from this exercise.

   Hint: A tutorial can be found at http://docs.python.org/2/howto/argparse.html

8 Calculating a Histogram

Learned Skills: Creating a function; List handling; Random numbers

1. Write a function `histgram(data, numbins)` which calculates the histogram of a given data set, where `numbins` gives the number of intervals in which the data range is divided.
   The function should return a tuple of two lists of equal lengths. The first list contains the midpoints of the intervals and the second list contains the counts of data points in the interval.

2. Give a pseudo-graphical representation of the distribution, by drawing a number of stars corresponding to the number of data elements in a given interval. Example:

   ```
   0.0 ***
   0.5 *****
   1.0 ********
   1.5 ******************
   2.0 *************
   2.5 **********
   3.0 ********
   3.5 ******
   4.0 **
   ```

3. Test the function by drawing samples from different probability distributions from the package `random`.