STAT/CS 94 Fall 2015 Adhikari HW02, Due: 9/9/2015

NAME: SID:

This homework asks you to provide some answers that involve math and some that involve code.

• **Math:** Your answers should be written just as they would be in a class where you weren't using a computer. You can leave answers unsimplified unless the question says otherwise. For example, answers like $(1/2) + (1/2)^3$ or $(364/365)(363/365)\cdots(350/365)$ are fine.

• **Code:** One of the most effective ways to improve your ability to program is to think through your first draft of a program using pencil and paper. Try to solve these questions without a computer. You can use a computer afterwards to check your work.

Problem 1

A student has written the following Python code:

```
beatles = ['John', 'Paul']
beatles + 'George' + 'Ringo'
```

hoping to create the list

```
['John', 'Paul', 'George', 'Ringo']
```

Has the student succeeded? If not, edit or rewrite the statements so that you use `beatles` and create the correct list for the student.

Answer: No, because adding a list and a string is an error. In general, only things of the same type can be added together. (It might seem like Python is being overly pedantic in this case. But notice this: if this expression were allowed, it would be hard to figure out, without carefully reading the Python specification, whether the result should be the desired list or `['John', 'Paul', 'GeorgeRingo']`.)

There are several reasonable ways to get the desired list, but one way is: `beatles + ['George', 'Ringo']`.

Problem 2

A Python array called `incomes` consists of 213 incomes arranged in increasing order. You can assume that all the incomes are different from each other, and that the `numpy` modules has been imported as `np`. Write Python expressions that generate:

(a) an array consisting of the integers 1 through 213

(b) the second-largest income

(c) the total of all the incomes

(d) a boolean that answers the question, "Is the largest income at least as large at 10 times the smallest income?"

(e) the smallest difference between any two incomes

(f) the maximum number of incomes you can sum together without exceeding 1000000 [Hint: You can cumulatively sum all elements of an array using `np.cumsum` and count how may `True` values appear in a boolean array using `np.count_nonzero`.]

Answer: As usual, there are many ways to accomplish things in Python, so your answer might be correct even if it doesn't exactly match ours.

(a) `np.arange(1, 214)`

(b) `incomes[len(incomes)-2]`

(c) `np.sum(incomes)`

(d) `incomes[len(incomes)-1] >= 10*incomes[0]`

(e) `np.min(np.diff(incomes))`

Some explanation is in order for this question. (You didn't need to explain your answer, of course.) First, the most reasonable interpretation is that the question is asking for the smallest nonnegative difference (else the answer would be `incomes[0] - incomes[len(incomes)-1]`). Now, our answer finds only the smallest difference between consecutive incomes. Why can't there be a smaller difference between some pair of nonconsecutive incomes? Because if there were, say between elements 2 and 0 of `incomes`, then the difference between elements 1 and 0 must be even smaller.

(f) `np.count_nonzero(np.cumsum(incomes) <= 1000000)`

If you're confused about this answer, break it down into its components, and use simple examples of `incomes` if you need to. What is `np.cumsum(incomes)`? Then, what is `np.cumsum(incomes) <= 1000000`? Why does counting the number of `True`s in that array give us the answer we wanted?

Problem 3

A *random number generator* draws repeatedly at random from among the 10 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, in such a way that each digit is equally likely to be drawn each time, regardless of the results of the other draws.

(a) Write a Python expression for $(9/10)^6 \times (1/10)$.

(b) The expression in part (a) is equal to the chance that (pick one option and explain your choice):

(i) the digit 0 appears on the 7th draw

(ii) the digit 0 appears exactly once in 7 draws

(iii) the digit 0 appears for the first time on the 7th draw

(iv) it takes more than 6 draws for the digit 0 to appear

Answer:

(a) `(9/10)**6 * (1/10)`

(b) (iii).

(i) is just 1/10.

We can think of (ii) as the chance that the digit 0 appears in (and only in) the first draw, or in (and only in) the second draw, or $\cdots$, or in (and only in) the seventh draw. There are seven of these events, and they are mutually exclusive (meaning no two events can both happen), since 0 cannot both appear exactly once and appear in two different draws. So the probability that any of them happens equals the sum of their individual probabilities. That is,

Probability(0 appears exactly once in 7 draws)
= Probability(0 appears only in the first draw, or 0 appears only in the second draw, or ...)
= Probability(0 appears only in the first draw)
+ Probability(0 appears only in the second draw)
$+ \cdots$
+ Probability(0 appears only in the seventh draw)

. Note that the second equality (in shorter, more abstract notation, $P(A \texttt{ or } B) = P(A) + P(B)$) is not true in general, but it is true for mutually exclusive events like this.

Each of those events has probability $(9/10)^6 \times (1/10)$, since six draws are constrained to be anything except 0 (which has probability 9/10 for each of the six draws); one draw is constrained to be 0 (which has probability 1/10); and the draws are independent, so the probabilities multiply. Thus (ii) has probability $7 \times (9/10)^6 \times (1/10)$.

(iii) is like (ii), but it only includes one of the seven events we listed above: 0 appears in (and only in) the seventh draw. So its probabiilty is just $(9/10)^6 \times (1/10)$.

We can think of (iv) as the chance that none of the first six draws is a 0; it doesn't constrain any other draws. So the probability of (iv) is just $(9/10)^6$.

Problem 4

A monkey is banging repeatedly on the keys of a typewriter. Each time, the monkey is equally likely to hit any of the 26 letters of the English alphabet, regardless of what it hits at other times. There are no other keys on the keyboard. Find a numerical expression (no code) for the chance that:

(a) the monkey types the sequence `datascience`

(b) the first ten letters that the monkey types are all different

(c) there is at least one `e` among the first four letters that the monkey types

Answer:

(a) $(1/26)^{11}$

The question does not specify the number of keys the monkey types, but the most reasonable interpretation is that the monkey hits just 11 keys, or that we at least do not care about the keys hit after the first 11. Since the chance of hitting any single letter is 1/26, the chance that the first character is a `'d'` is 1/26, the chance that the second character is a `'a'` is 1/26, and so on. Since the letters are independent, probability of getting exactly `'datascience'` is just 1/26 of 1/26 of 1/26 $\cdots$, 11 times, or $(1/26)^{11}$.

(b) $\frac{26!}{(26-10)!}(\frac{1}{26})^{10}$.

We have seen this kind of calculation in the birthday problem. The way to think about this is letter-by-letter. For a sequence of 10 letters to be all different, each letter must be different from the previous letters (in fact, if you think about it, this is *equivalent* to the sequence being all different).

The first letter has a 26/26 chance of being unique among all the letters up to it. The second letter has a 25/26 of not matching the first. In general, if the 1st through nth letters are all different, then the $(n+1)$th letter has a $(26-n)/26$ chance of not matching any of them, since there are n possible letters for it to match and $(26-n)$ remaining "safe" letters. So the probability is $\frac{26}{26} \times \frac{25}{26} \times \frac{24}{26} \times \frac{23}{26} \times \frac{22}{26} \times \frac{21}{26} \times \frac{20}{26} \times \frac{19}{26} \times \frac{18}{26} \times \frac{17}{26}$. Our answer above is just a more succinct way of writing this. (It is of course fine if you wrote your answer in a different, equivalent way.)

(c) $1 - (\frac{25}{26})^4$.

Here the trick of taking the complement is useful. "At least one `e`" means there could be exactly one `e`, exactly two, exactly three, or exactly four. Rather than compute each of those complicated probabilities, we can compute the probability of the only case that is left – that there are exactly zero `e`'s – and take its complement. The chance that there are zero `e`'s is just $(\frac{25}{26})^4$, since the chance that each letter is not an `e` is 25/26, and the letters are typed independently. The complement of an event has probability 1 minus the probability of the event.

Problem 5

Use the function `np.arange` to write a Python expression for the answer to part (b) in the problem above.

Answer: `np.prod(np.arange(17, 27)) / (26**10)` (To be clear, we did not expect you to use *only* the function `np.arange`.)

Problem 6

Let N be an even positive integer. Qualitatively describe the result of the following expressions in terms of N (for example, "a tuple consisting of the first N even numbers").

(a) `(np.arange(1, N+1))**2`

(b) `np.cumsum(np.arange(1, 2*N+1, 2))`

Answer:

(a) The first N perfect squares, starting with 1.

(b) The same – the first N perfect squares, starting with 1! The reason this is true is the following formula for a perfect square in terms of the previous one: $(x+1)^2 = x^2 + 2x + 1$. The xth element of `np.arange(1, 2*N+1, 2)` is $2x + 1$, so the cumulative sums obey this formula. (It's fine if you just ran the code and noticed the pattern.)