Data 8 Spring 2016 HW02, Due: 5pm Thursday, February 4

Please write your answers in the space provided. You can write on a printed copy or fill in the blanks with a PDF editor such as Acrobat Reader or Apple Preview. When you're done, upload a scanned copy to Gradescope (`gradescope.com`). If you cannot submit online, you may submit a printed copy in office hours. You can find blank printed copies outside of 781 Soda. This assignment is due 5pm Thursday, February 4. You will receive an early submission bonus point if you turn it in by 5pm Wednesday, February 3.

You are welcome to use `data8.berkeley.edu` to try out Python expressions. Directly sharing answers is not okay, but discussing problems with course staff or students is encouraged.

Collaborators:

Problem 1 (Arrays)

A Python array called `incomes` contains incomes **in increasing order**. Assume that all the incomes are different, and that the `numpy` module has been imported as `np`. Write Python expressions that generate each of the results described below.

- the total sum of all the incomes
  1(a):
- the second-largest income
  1(b):
- the maximum difference between any two incomes
  1(c):
- a boolean that answers the question, "Is the largest income at least as large at 10 times the smallest income?"
  1(d):
- the smallest positive difference between any two incomes
  1(e):
- the maximum number of incomes you can sum together without exceeding 1000000 [Hint: You can count how may `True` values appear in a boolean array using `np.count_nonzero`.]
  1(f):

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. The functions `max`, `min`, and `sum` all exist in two forms, built-in functions and part of the `numpy` module.

(a) `np.sum(incomes)`

(b) `incomes.item(len(incomes)-2)`

(c) `incomes.item(len(incomes)-1) - incomes.item(0) or max(incomes)-min(incomes)`

(d) `incomes.item(len(incomes)-1) >= 10*incomes.item(0) or max(incomes) >= 10*min(incomes)`

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

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

If you're confused about this last 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 2 (Interpreting Code)

Let N be a positive integer. Complete the qualitative descriptions of the results of the following expressions. Each description should include a number. The first is provided as an example.

- `np.arange(N+5) + 1`

  The first $N + 5$ positive integers.

- `np.arange(1, N-1)**2`

  2(a): *The first*

- `np.prod(1-np.arange(N)/365)`

  2(c): *The chance that*

  (Under the birthday assumptions from lecture)

Answer:

(a) $N - 2$ positive perfect squares

(b) N students all have different birthdays

Problem 3 (Creating a Table)

Fill in the missing arguments to the method call in each line of code below to create the table below to the left and name it `t`. Tables are described in `http://inferentialthinking.com/chapter1/tables.html`.

```
x | y          q = Table().with_column('z',
1 | 10
2 | 20         r = q.with_column(
3 | 30
4 | 40         s = r.with_row(

               t = s.relabeled(
```

Answer:

```
q = Table().with_column('z', [1, 2, 3])
r = q.with_column('y', [10, 20, 30])
s = r.with_row([4, 40])
t = s.relabeled('z', 'x')
```

Problem 4 (iTable)

Apple recently reported a Q1 quarterly profit of $18.4 billion, the largest ever recored in the history of any public corporation. Using Apple's released sales data, we have created a Table named `sales` for Q1 over the last few years:

```
Product     | Year  | Units Sold
iPhone      | 2013  | 47789
iPad        | 2014  | 26035
iPhone      | 2015  | 74468
iPad        | 2013  | 22860
iPhone      | 2014  | 51025
iPad        | 2015  | 21419
```

Write the value of each expression below. For any array-valued expressions, abbreviate by writing, for example, `[True, False]` instead of `array([True, False], dtype=bool`.)

- `sales.column('Year')`
  4(a):
- `len(sales.column('Year'))`
  4(b):
- `sales.column('Year') >= 2014`
  4(c):
- `len(sales.column('Year') >= 2014)`
  4(d):
- `np.all(sales.column(2) > 20000)`
  4(e):
- `sales.row(3).item(0)`
  4(f):
- `sales.select([1, 0]).column(0).item(2)`
  4(g):
- `np.diff(np.sort(sales.column('Year')))`
  4(h):

Answer:

(a) `[2013, 2014, 2015, 2013, 2014, 2015]`

(b) 6

(c) `[False, True, True, False, True, True]`

Careful with this one! Understanding this one leads easily to the following answer.

(d) 6

(e) True

(f) `'iPad'`

(g) 2015

(h) `[0, 1, 0, 1, 0]`

Problem 5 (Exponential Growth)

At the height of the Ebola outbreak in 2014, the World Health Organization monitored the countries Guinea, Liberia, and Sierra Leone and recorded the following metrics.

```
date       | total cases (cumulative)
2014-09-18 | 5325
2014-09-22 | 5843
```

(a) Write an expression that computes the *daily* growth rate over this 4-day period?

5(a):

(b) Assuming this growth rate had remained constant, write an expression that computes the total cases we would have expected by December 31, 2014. December 31 is 100 days after September 22.

5(b):

Answer:

(a) `(5843/5325) ** (1/4) - 1`, which is 0.0235.

(b) `((5843/5325) ** (1/4)) ** 100 * 5843`. This expression predicts 59504 cases of Ebola. In reality, the WHO had reported 20171 cases of Ebola by December 31 2014 – the growth rate fell from its peak as policies were implemented to prevent spread of the outbreak.

Problem 6 (Sequences)

In this problem we will review the behavior of lists and arrays in Python. Below are several attempts at snippets of code. For each, rewrite exactly one of the two lines of code in order to achieve the expected behavior. Assume that the `numpy` module has been imported as `np`.

(a) A restaurant wants to double prices of all the items on its menu.

```
prices = [20, 15, 10]
prices = prices * 2
```

6(a):

(b) A movie theatre wants to add a new $10.50 option to its prices of tickets.

```
tickets = np.array([8.50, 12.00])
tickets = tickets + [10.50]
```

6(b):

(c) A student wants to complete a list of the Beatles.

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

6(c):

Answer:

(a) Multiplying a list by a value will duplicate the list – the result will be `[20, 15, 10, 20, 15, 10]`. In order to perform element-wise operations, the programmer should be using an array instead of a list. Instead, one approach is:

```
prices = np.array([20, 15, 10])
prices = prices * 2
```

(b) `tickets` is an array and not a list. The `+` operation will be interpeted as an element-wise addition rather than as concatenation, and the result will be: `[19.0, 22.5]`. There are many ways to fix this, but one is to replace the array with a list:

```
tickets = [8.50, 12.00]
tickets = tickets + [10.50]
```

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