lab01

February 22, 2022

Name: Allan Gongora

Section: 0131

1 Lab 1: Introduction to Python

Welcome to Lab 1! Each week you will complete a lab assignment like this one. In this lab, you'll get started with the Python programming language through numbers, names, and expressions.

Recommended Reading: - Programming in Python

1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include:
   A) Sentence reponses to questions that ask for an explanation
   B) Numeric responses to multiple choice questions
   C) Programming code
2) Moreover, throughout this lab and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Otherwise, you will fail tests that you thought you were passing previously!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

1.1 1. Numbers

Quantitative information arises everywhere in data science. In addition to representing commands to print out lines, expressions can represent numbers and methods of combining numbers. The expression `3.2500` evaluates to the number 3.25. (Run the cell and see.)

```
[1]: 3.2500
```

```
[1]: 3.25
```

Notice that we didn't have to `print`. When you run a notebook cell, if the last line has a value, then Jupyter helpfully prints out that value for you. However, it won't print out prior lines automatically. If you want to print out a prior line, you need to add the `print` statement. Run the cell below to check.

```
[2]: print(2)
     3
     4
```

```
2
```

```
[2]: 4
```

Above, you should see that 4 is the value of the last expression, 2 is printed, but 3 is lost forever because it was neither printed nor last.

You don't want to print everything all the time anyway. But if you feel sorry for 3, change the cell above to print it.

1.1.1 Arithmetic

The line in the next cell subtracts. Its value is what you'd expect. Run it.

```
[3]: 3.25 - 1.5
```

```
[3]: 1.75
```

Many basic arithmetic operations are built in to Python. The textbook section on Expressions describes all the arithmetic operators used in the course. The common operator that differs from typical math notation is `**`, which raises one number to the power of the other. So, `2**3` stands for 2^3 and evaluates to 8.

The order of operations is what you learned in elementary school, and Python also has parentheses. For example, compare the outputs of the cells below. Use parentheses for a happy new year!

```
[4]: 5+6*5-6*3**2*2**3/4*7
```

```
[4]: -721.0
```

```
[5]: 5+(6*5-(6*3))**2*((2**3)/4*7)
```

```
[5]: 2021.0
```

In standard math notation, the first expression is

$$5 + 6 \times 5 - 6 \times 3^2 \times \frac{2^3}{4} \times 7,$$

while the second expression is

$$5 + (6 \times 5 - (6 \times 3))^2 \times (\frac{(2^3)}{4} \times 7).$$

Question 1.1 Write a Python expression in this next cell that's equal to $5 \times (3\frac{10}{11}) - 50\frac{1}{3} + 2^{.5\times 22} - \frac{7}{33} + 4$. That's five times three and ten elevenths, minus 50 and a third, plus two to the power of half of 22, minus seven 33rds plus four. By "$3\frac{10}{11}$" we mean $3 + \frac{10}{11}$, not $3 \times \frac{10}{11}$.

Replace the ellipses (`...`) with your expression. Try to use parentheses only when necessary.

Hint: The correct output should be a familiar number.

```
[6]: (5*(3+(10/11))) - (50+(1/3)) + 2**(.5*22) - (7/33) + 4
```

```
[6]: 2021.0
```

1.2 2. Names

In natural language, we have terminology that lets us quickly reference very complicated concepts. We don't say, "That's a large mammal with brown fur and sharp teeth!" Instead, we just say, "Bear!"

Similarly, an effective strategy for writing code is to define names for data as we compute it, like a lawyer would define terms for complex ideas at the start of a legal document to simplify the rest of the writing.

In Python, we do this with *assignment statements*. An assignment statement has a name on the left side of an `=` sign and an expression to be evaluated on the right.

```
[7]: ten = 3 * 2 + 4
```

When you run that cell, Python first evaluates the first line. It computes the value of the expression `3 * 2 + 4`, which is the number 10. Then it gives that value the name `ten`. At that point, the code in the cell is done running.

After you run that cell, the value 10 is bound to the name `ten`:

```
[8]: ten
```

```
[8]: 10
```

The statement `ten = 3 * 2 + 4` is not asserting that `ten` is already equal to `3 * 2 + 4`, as we might expect by analogy with math notation. Rather, that line of code changes what `ten` means; it now refers to the value 10, whereas before it meant nothing at all.

If the designers of Python had been ruthlessly pedantic, they might have made us write

```
define the name ten to hereafter have the value of 3 * 2 + 4
```

instead. You will probably appreciate the brevity of "`=`"! But keep in mind that this is the real meaning.

Question 2.1 Try writing code that uses a name (like `eleven`) that hasn't been assigned to anything. You'll see an error!

```
[9]: eleven = ten + 1
```

A common pattern in Jupyter notebooks is to assign a value to a name and then immediately evaluate the name in the last line in the cell so that the value is displayed as output.

```
[10]: close_to_pi = 355/113
      close_to_pi
```

```
[10]: 3.1415929203539825
```

Another common pattern is that a series of lines in a single cell will build up a complex computation in stages, naming the intermediate results.

```
[11]: bimonthly_salary = 840
      monthly_salary = 2 * bimonthly_salary
      number_of_months_in_a_year = 12
      yearly_salary = number_of_months_in_a_year * monthly_salary
      yearly_salary
```

```
[11]: 20160
```

Names in Python can have letters (upper- and lower-case letters are both okay and count as different letters), underscores, and numbers. The first character can't be a number (otherwise a name might look like a number). And names can't contain spaces, since spaces are used to separate pieces of code from each other.

Other than those rules, what you name something doesn't matter *to Python*. For example, this cell does the same thing as the above cell, except everything has a different name:

```
[12]: a = 840
      b = 2 * a
      c = 12
      d = c * b
      d
```

```
[12]: 20160
```

However, names are very important for making your code *readable* to yourself and others. The cell above is shorter, but it's totally useless without an explanation of what it does.

According to a famous joke among computer scientists, naming things is one of the two hardest problems in computer science. (The other two are cache invalidation and "off-by-one" errors. And people say computer scientists have an odd sense of humor…)

Question 2.2 Assign the name `seconds_in_a_decade` to the number of seconds between midnight January 1, 2010 and midnight January 1, 2020. Use Python to perform any required arithmetic.

Hint: If you're stuck, the next section shows you how to get hints.

```
[13]: # Change the next line so that it computes the number of
      # seconds in a decade and assigns that number the name
      # seconds_in_a_decade.
      seconds_in_a_decade = (8 * (60*60*24*365)) + (2 * (60*60*24*366))

      # We've put this line in this cell so that it will print
      # the value you've given to seconds_in_a_decade when you
      # run it.  You don't need to change this.
      seconds_in_a_decade
```

```
[13]: 315532800
```

1.2.1 Checking Your Code

Now that you know how to name things, you can start using the built-in *tests* to check whether your work is correct. Try not to change the contents of the test cells.

The cell below appears only once in the notebook and loads all of the tests so that they can be run later. You can load all of the tests before you answer all questions in the notebook. You will run tests as you go to check your work along the way, and you can also run all of the tests at the end to make sure that you will receive full credit on the lab.

```
[14]: pip install gofer-grader
```

```
Collecting gofer-grader
  Using cached gofer_grader-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
packages (from jinja2->gofer-grader) (2.0.1)
Installing collected packages: gofer-grader
Successfully installed gofer-grader-1.1.0
Note: you may need to restart the kernel to use updated packages.
```

```
[15]: # These lines load the tests.
      from gofer.ok import check
```

Running the following cell will test whether you have assigned `seconds_in_a_decade` correctly in Question 2.2.

Sometimes the tests will give hints about what went wrong. If the test doesn't pass, read the output, adjust your answer to the question, run the answer cell again to update the name `seconds_in_a_decade`, then run this test cell again.

Sometimes the tests will tell you the answer. Rather than copying the answer, try to understand how it was reached.

[16]:
```
# Test cell; please do not change!
check('tests/q22.py')
```

[16]:
```
<gofer.ok.OKTestsResult at 0x7f4570071190>
```

1.2.2 Comments

You may have noticed this line in the cell above:

```
# Test cell; please do not change!
```

That is called a *comment*. It doesn't make anything happen in Python; Python ignores anything on a line after a #. Instead, it's there to communicate something about the code to you, the human reader. Comments are extremely useful.

1.2.3 Application: A Physics Experiment

On the Apollo 15 mission to the Moon, astronaut David Scott famously replicated Galileo's physics experiment in which he showed that gravity accelerates objects of different masses at the same rate. Because there is no air resistance for a falling object on the surface of the Moon, even two objects with very different masses and densities should fall at the same rate. David Scott compared a feather and a hammer.

You can run the following cell to watch a video of the experiment.

[17]:
```
from IPython.display import YouTubeVideo
# The original URL is:
#   https://www.youtube.com/watch?v=U7db6ZeLR5s
YouTubeVideo("U7db6ZeLR5s")
```

[17]:

Here's the transcript of the video:

167:22:06 Scott: Well, in my left hand, I have a feather; in my right hand, a hammer. And I guess one of the reasons we got here today was because of a gentleman named Galileo, a long time ago, who made a rather significant discovery about falling objects in gravity fields. And we thought where would be a better place to confirm his findings than on the Moon. And so we thought we'd try it here for you. The feather happens to be, appropriately, a falcon feather for our Falcon. And I'll drop the two of them here and, hopefully, they'll hit the ground at the same time.

167:22:43 Scott: How about that!

167:22:45 Allen: How about that! (Applause in Houston)

167:22:46 Scott: Which proves that Mr. Galileo was correct in his findings.

Newton's Law. Using this footage, we can also attempt to confirm another famous bit of physics: Newton's law of universal gravitation. Newton's laws predict that any object dropped near the surface of the Moon should fall:

$$\frac{1}{2}G\frac{M}{R^2}t^2 \text{ meters}$$

after t seconds, where G is a universal constant, M is the moon's mass in kilograms, and R is the moon's radius in meters. So if we know G, M, and R, then Newton's laws let us predict how far an object will fall over any amount of time.

To verify the accuracy of this law, we will calculate the difference between the predicted distance the hammer drops and the actual distance. (If they are different, it might be because Newton's laws are wrong, or because our measurements are imprecise, or because there are other factors affecting the hammer for which we haven't accounted.)

Someone studied the video and estimated that the hammer was dropped 113 cm from the surface. Counting frames in the video, the hammer falls for 1.2 seconds (36 frames).

Question 2.3 Complete the code in the next cell to fill in the *data* from the experiment.

```
[18]: # t, the duration of the fall in the experiment, in seconds.
      # Fill this in.
      time = 1.2

      # The estimated distance the hammer actually fell, in meters.
      # Fill this in.
      estimated_distance_m = 1.13
```

```
[19]: check('tests/q231.py')
```

```
[19]: <gofer.ok.OKTestsResult at 0x7f455adce9d0>
```

Question 2.4 Now, complete the code in the next cell to compute the difference between the predicted and estimated distances (in meters) that the hammer fell in this experiment.

This just means translating the formula above ($\frac{1}{2}G\frac{M}{R^2}t^2$) into Python code. You'll have to replace each variable in the math formula with the name we gave that number in Python code.

```
[20]: # First, we've written down the values of the 3 universal
      # constants that show up in Newton's formula.

      # G, the universal constant measuring the strength of gravity.
      gravity_constant = 6.674 * 10**-11

      # M, the moon's mass, in kilograms.
      moon_mass_kg = 7.34767309 * 10**22

      # R, the radius of the moon, in meters.
      moon_radius_m = 1.737 * 10**6

      # The distance the hammer should have fallen over the
      # duration of the fall, in meters, according to Newton's
      # law of gravity.  The text above describes the formula
      # for this distance given by Newton's law.
      # **YOU FILL THIS PART IN.**
      predicted_distance_m = (1/2) * gravity_constant * (moon_mass_kg/
       ↪moon_radius_m**2) * time**2
```

```
# Here we've computed the difference between the predicted
# fall distance and the distance we actually measured.
# If you've filled in the above code, this should just work.
difference = predicted_distance_m - estimated_distance_m
difference
```

[20]: 0.040223694659304865

[21]:
```
check('tests/q232.py')
```

[21]: <gofer.ok.OKTestsResult at 0x7f457020c110>

1.3 3. Calling Functions

The most common way to combine or manipulate values in Python is by calling functions. Python comes with many built-in functions that perform common operations.

For example, the `abs` function takes a single number as its argument and returns the absolute value of that number. The absolute value of a number is its distance from 0 on the number line, so `abs(5)` is 5 and `abs(-5)` is also 5.

[22]:
```
abs(5)
```

[22]: 5

[23]:
```
abs(-5)
```

[23]: 5

1.3.1 Application: Computing Walking Distances

Chunhua is on the corner of 7th Avenue and 42nd Street in Midtown Manhattan, and she wants to know far she'd have to walk to get to Gramercy School on the corner of 10th Avenue and 34th Street.

She can't cut across blocks diagonally, since there are buildings in the way. She has to walk along the sidewalks. Using the map below, she sees she'd have to walk 3 avenues (long blocks) and 8 streets (short blocks). In terms of the given numbers, she computed 3 as the difference between 7 and 10, *in absolute value*, and 8 similarly.

Chunhua also knows that blocks in Manhattan are all about 80m by 274m (avenues are farther apart than streets). So in total, she'd have to walk $(80 \times |42 - 34| + 274 \times |7 - 10|)$ meters to get to the park.

Question 3.1 Finish the line `num_avenues_away = ...` in the next cell so that the cell calculates the distance Chunhua must walk and give it the name `manhattan_distance`. Everything else has been filled in for you. **Use the `abs` function.**

[24]:
```python
# Here's the number of streets away:
num_streets_away = abs(42-34)

# Compute the number of avenues away in a similar way:
num_avenues_away = abs(7 - 10)

street_length_m = 80
avenue_length_m = 274

# Now we compute the total distance Chunhua must walk.
manhattan_distance = street_length_m*num_streets_away +␣
 ↪avenue_length_m*num_avenues_away

# We've included this line so that you see the distance
# you've computed when you run this cell.  You don't need
# to change it, but you can if you want.
manhattan_distance
```

[24]:
```
1462
```

Be sure to run the next cell to test your code.

[25]:
```python
check('tests/q311.py')
```

[25]:
```
<gofer.ok.OKTestsResult at 0x7f455adce5d0>
```

1.3.2 Multiple Arguments

Some functions take multiple arguments, separated by commas. For example, the built-in `max` function returns the maximum argument passed to it.

[26]:
```python
max(2, -3, 4, -5)
```

[26]:
```
4
```

1.4 4. Understanding Nested Expressions

Function calls and arithmetic expressions can themselves contain expressions. You saw an example in the last question:

`abs(42-34)`

has 2 number expressions in a subtraction expression in a function call expression. And you probably wrote something like `abs(7-10)` to compute `num_avenues_away`.

Nested expressions can turn into complicated-looking code. However, the way in which complicated expressions break down is very regular.

Suppose we are interested in heights that are very unusual. We'll say that a height is unusual to the extent that it's far away on the number line from the average human height. An estimate of the average adult human height (averaging, we hope, over all humans on Earth today) is 1.688 meters.

So if Aditya is 1.21 meters tall, then his height is $|1.21 - 1.688|$, or .478, meters away from the average. Here's a picture of that:

And here's how we'd write that in one line of Python code:

```
[27]: abs(1.21 - 1.688)
```

```
[27]: 0.478
```

What's going on here? `abs` takes just one argument, so the stuff inside the parentheses is all part of that *single argument*. Specifically, the argument is the value of the expression `1.21 - 1.688`. The value of that expression is `-.478`. That value is the argument to `abs`. The absolute value of that is `.478`, so `.478` is the value of the full expression `abs(1.21 - 1.688)`.

Picture simplifying the expression in several steps:

1. `abs(1.21 - 1.688)`
2. `abs(-.478)`
3. `.478`

In fact, that's basically what Python does to compute the value of the expression.

Question 4.1 Say that Botan's height is 1.85 meters. In the next cell, use `abs` to compute the absolute value of the difference between Botan's height and the average human height. Give that value the name `botan_distance_from_average_m`.

```
[28]: # Replace the ... with an expression to compute the absolute
      # value of the difference between Botan's height (1.85m) and
      # the average human height.
      botan_distance_from_average_m = 1.85 - 1.688

      # Again, we've written this here so that the distance you
      # compute will get printed when you run this cell.
      botan_distance_from_average_m
```

```
[28]: 0.16200000000000014
```

```
[29]: check('tests/q41.py')
```

```
[29]: <gofer.ok.OKTestsResult at 0x7f45701f3350>
```

1.4.1 More Nesting

Now say that we want to compute the most unusual height among Aditya's and Botan's heights. We'll use the function `max`, which (again) takes two numbers as arguments and returns the larger of the two arguments. Combining that with the `abs` function, we can compute the biggest distance from the average among the two heights:

[30]:
```
# Just read and run this cell.

aditya_height_m = 1.21
botan_height_m = 1.85
average_adult_human_height_m = 1.688

# The biggest distance from the average human height, among the two heights:
biggest_distance_m = max(abs(aditya_height_m - average_adult_human_height_m),␣
 ↪abs(botan_height_m - average_adult_human_height_m))

# Print out our results in a nice readable format:
print("The biggest distance from the average height among these two people is",␣
 ↪biggest_distance_m, "meters.")
```

```
The biggest distance from the average height among these two people is 0.478
meters.
```

The line where `biggest_distance_m` is computed looks complicated, but we can break it down into simpler components just like we did before.

The basic recipe is repeated simplification of small parts of the expression: * We start with the simplest components whose values we know, like plain names or numbers. (Examples: `aditya_height_m` or `5`.) * **Find a simple-enough group of expressions:** We look for a group of simple expressions that are directly connected to each other in the code, for example by arithmetic or as arguments to a function call. * **Evaluate that group:** We evaluate the arithmetic expressions or function calls they're part of, and replace the whole group with whatever we compute. (Example: `aditya_height_m - average_adult_human_height_m` becomes `-.478`.) * **Repeat:** We continue this process, using the values of the glommed-together stuff as our new basic components. (Example: `abs(-.478)` becomes `.478`, and `max(.478, .162)` later becomes `.478`.) * We keep doing that until we've evaluated the whole expression.

You can run the next cell to see a slideshow of that process.

[31]:
```
from IPython.display import IFrame
IFrame('https://docs.google.com/presentation/d/
 ↪1urkX-nRsD8VJvcOnJsjmCy0Jpv752Ssn5Pphg2sMC-0/embed?
 ↪start=false&loop=false&delayms=3000', 800, 600)
```

[31]:
```
<IPython.lib.display.IFrame at 0x7f455acefc90>
```

Ok, your turn.

Question 4.2 Given the heights of the Splash Triplets from the Golden State Warriors, write an expression that computes the smallest difference between any of the three heights. Your expression shouldn't have any numbers in it, only function calls and the names `klay`, `steph`, and `kevin`. Give the value of your expression the name `min_height_difference`.

[32]:
```
# The three players' heights, in meters:
klay =  2.01 # Klay Thompson is 6'7"
steph = 1.91 # Steph Curry is 6'3"
```

```
kevin = 2.06 # Kevin Durant is officially 6'9", but many suspect that he is␣
↪taller.
             # (Further complicating matters, membership of the "Splash␣
↪Triplets"
             #  is disputed, since it was originally used in reference to
             #  Klay Thompson, Steph Curry, and Draymond Green.)

# We'd like to look at all 3 pairs of heights, compute the absolute
# difference between each pair, and then find the smallest of those
# 3 absolute differences.  This is left to you!  If you're stuck,
# try computing the value for each step of the process (like the
# difference between Klay's heigh and Steph's height) on a separate
# line and giving it a name (like klay_steph_height_diff).
min_height_difference = min([abs(klay - steph), abs(klay - kevin), abs(steph -␣
↪kevin)])
```

[33]:
```
check('tests/q411.py')
```

[33]:
```
<gofer.ok.OKTestsResult at 0x7f455acea990>
```

1.5 5. Tables

A website called Gapminder collects a large variety of measurements of human health, education, and progress. Each measurement is published in a table that has one row per country and one column per year, describing how the measurement varies over time and place.

For example, this table describes the average number of years of school attended by all women 25 and older. The table has a row for each of 175 countries and a column for each year from 1970 through 2009. The data were estimated for a study by the Institute for Health Metrics and Evaluation called "Increased educational attainment and its impact on child mortality: a systematic analysis in 175 countries from 1970 to 2009" (link).

To load tables into Python, you must first import the `datascience` module. The second line below makes sure that charts appear on the screen when you create them. You only need to execute these lines once per notebook (and each time you restart your kernel).

[34]:
```
# Don't change this cell
from datascience import *
%matplotlib inline
```

Now, run the next cell in order to load the table describing years of school attended by women around the world and over time. Only the first 10 rows of the table will be displayed.

[35]:
```
school = Table.read_table('school.csv')
school
```

[35]:
```
Row Labels           | 1970 | 1971 | 1972 | 1973 | 1974 | 1975 | 1976 | 1977 |
1978 | 1979 | 1980 | 1981 | 1982 | 1983 | 1984 | 1985 | 1986 | 1987 | 1988 |
1989 | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 |
2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009
Afghanistan         | 0    | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  |
0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1
| 0.1  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  | 0.2  |
0.3  | 0.3  | 0.3  | 0.3  | 0.3  | 0.3  | 0.3  | 0.3  | 0.4
Albania             | 3.9  | 4    | 4.1  | 4.2  | 4.3  | 4.5  | 4.6  | 4.7  |
4.8  | 4.9  | 5.1  | 5.2  | 5.3  | 5.4  | 5.6  | 5.7  | 5.9  | 6    | 6.2  | 6.3
| 6.5  | 6.6  | 6.8  | 6.9  | 7.1  | 7.2  | 7.4  | 7.5  | 7.7  | 7.8  | 8    |
8.2  | 8.3  | 8.5  | 8.6  | 8.8  | 8.9  | 9.1  | 9.2  | 9.4
Algeria             | 0.6  | 0.6  | 0.6  | 0.7  | 0.7  | 0.8  | 0.8  | 0.9  |
0.9  | 1    | 1.1  | 1.1  | 1.2  | 1.3  | 1.4  | 1.5  | 1.5  | 1.6  | 1.7  | 1.8
| 1.9  | 2    | 2.1  | 2.2  | 2.3  | 2.4  | 2.5  | 2.6  | 2.8  | 2.9  | 3    |
3.1  | 3.2  | 3.4  | 3.5  | 3.6  | 3.8  | 3.9  | 4    | 4.2
Angola              | 0.5  | 0.5  | 0.5  | 0.5  | 0.6  | 0.6  | 0.6  | 0.7  |
0.7  | 0.7  | 0.8  | 0.8  | 0.8  | 0.9  | 0.9  | 1    | 1    | 1.1  | 1.1  | 1.2
| 1.2  | 1.3  | 1.3  | 1.4  | 1.5  | 1.5  | 1.6  | 1.7  | 1.8  | 1.8  | 1.9  | 2
| 2.1  | 2.2  | 2.3  | 2.4  | 2.5  | 2.6  | 2.7  | 2.8
Antigua and Barbuda | 7    | 7.1  | 7.2  | 7.4  | 7.5  | 7.7  | 7.8  | 8    |
8.1  | 8.3  | 8.4  | 8.6  | 8.7  | 8.9  | 9.1  | 9.2  | 9.4  | 9.5  | 9.7  | 9.8
| 10   | 10.1 | 10.2 | 10.4 | 10.5 | 10.6 | 10.8 | 10.9 | 11   | 11.1 | 11.2 |
11.4 | 11.5 | 11.6 | 11.7 | 11.8 | 11.9 | 12   | 12.1 | 12.2
Argentina           | 5.5  | 5.6  | 5.7  | 5.9  | 6    | 6.1  | 6.2  | 6.3  |
6.5  | 6.6  | 6.7  | 6.8  | 7    | 7.1  | 7.2  | 7.3  | 7.4  | 7.6  | 7.7  | 7.8
| 7.9  | 8    | 8.1  | 8.3  | 8.4  | 8.5  | 8.6  | 8.7  | 8.8  | 9    | 9.1  |
9.2  | 9.3  | 9.4  | 9.5  | 9.6  | 9.8  | 9.9  | 10   | 10.1
Armenia             | 5.9  | 6    | 6.2  | 6.3  | 6.5  | 6.6  | 6.8  | 6.9  |
7.1  | 7.2  | 7.4  | 7.5  | 7.7  | 7.8  | 8    | 8.1  | 8.3  | 8.4  | 8.6  | 8.7
| 8.9  | 9    | 9.1  | 9.3  | 9.4  | 9.6  | 9.7  | 9.8  | 10   | 10.1 | 10.2 |
10.3 | 10.5 | 10.6 | 10.7 | 10.8 | 11   | 11.1 | 11.2 | 11.3
Australia           | 8.4  | 8.5  | 8.6  | 8.7  | 8.8  | 8.9  | 9    | 9.1  |
9.2  | 9.3  | 9.4  | 9.5  | 9.6  | 9.6  | 9.7  | 9.8  | 9.9  | 10   | 10.1 |
10.1 | 10.2 | 10.3 | 10.4 | 10.4 | 10.5 | 10.6 | 10.7 | 10.7 | 10.8 | 10.9 |
10.9 | 11   | 11.1 | 11.1 | 11.2 | 11.3 | 11.3 | 11.4 | 11.4 | 11.5
Austria             | 7.2  | 7.3  | 7.4  | 7.5  | 7.6  | 7.7  | 7.9  | 8    |
8.1  | 8.2  | 8.3  | 8.4  | 8.5  | 8.6  | 8.7  | 8.8  | 8.9  | 9    | 9.1  | 9.1
| 9.2  | 9.3  | 9.4  | 9.5  | 9.6  | 9.7  | 9.8  | 9.9  | 10   | 10.1 | 10.2 |
10.3 | 10.4 | 10.4 | 10.5 | 10.6 | 10.7 | 10.8 | 10.9 | 11
Azerbaijan          | 4.5  | 4.6  | 4.8  | 5    | 5.1  | 5.3  | 5.5  | 5.7  |
5.8  | 6    | 6.2  | 6.4  | 6.6  | 6.8  | 7    | 7.2  | 7.4  | 7.6  | 7.8  | 7.9
| 8.1  | 8.3  | 8.5  | 8.7  | 8.9  | 9.1  | 9.2  | 9.4  | 9.6  | 9.8  | 9.9  |
10.1 | 10.2 | 10.4 | 10.6 | 10.7 | 10.9 | 11   | 11.2 | 11.3
… (165 rows omitted)
```

Question 5.1 Assign the name `top_1970` to a two-column table that has the column of country names (labeled `"Row Labels"`) and the years in school in 1970, sorted by the second column in decreasing order. Notice the large difference between the country with the most average years of school and the rest in the top 10.

Hint: Even though 1970 is a number, treat it as text by placing it within quotation marks when using it as a label. For example, `school.select("1970")` rather than `school.select(1970)`. Column labels are always text.

```
[36]: top_1970 = school.select("Row Labels", "1970").sort('1970', descending=True)
      top_1970
```

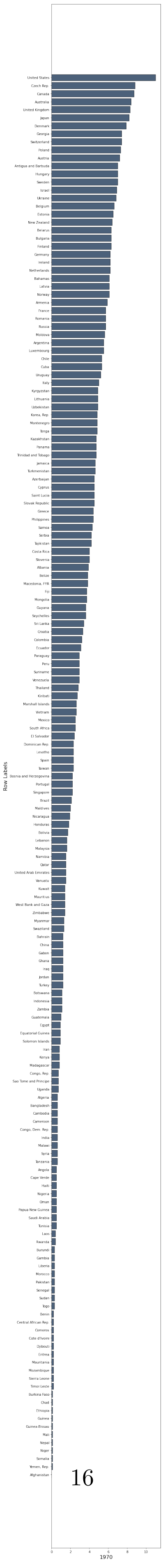
```
[36]: Row Labels     | 1970
      United States  | 11
      Czech Rep.     | 8.8
      Canada         | 8.7
      Australia      | 8.4
      United Kingdom | 8.3
      Japan          | 8.2
      Denmark        | 7.9
      Georgia        | 7.4
      Switzerland    | 7.4
      Poland         | 7.3
      … (165 rows omitted)
```

```
[37]: check('tests/q51.py')
```

```
[37]: <gofer.ok.OKTestsResult at 0x7f4540b89a90>
```

You can create a bar chart of all the countries in the data set using the expression below.

```
[38]: top_1970.barh('Row Labels')
```


United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan
Row Labels
0
2
4
6
8
10
1970

Question 5.2 Now, to see how much these numbers have changed, assign `top_1970_with_2009` to a table with the rows in the same order, but include a third column for 2009 as well. The differences between countries are much smaller in 2009.

```
[39]: top_1970_with_2009 = school.select("Row Labels", "1970", "2009").sort("1970",␣
        ↪descending=True)
      top_1970_with_2009
```

```
[39]: Row Labels     | 1970 | 2009
      United States  | 11   | 13.7
      Czech Rep.     | 8.8  | 13.3
      Canada         | 8.7  | 14.2
      Australia      | 8.4  | 11.5
      United Kingdom | 8.3  | 13
      Japan          | 8.2  | 12.2
      Denmark        | 7.9  | 12.8
      Georgia        | 7.4  | 12.5
      Switzerland    | 7.4  | 12.6
      Poland         | 7.3  | 12.4
      … (165 rows omitted)
```

```
[40]: top_1970_with_2009.labels
```

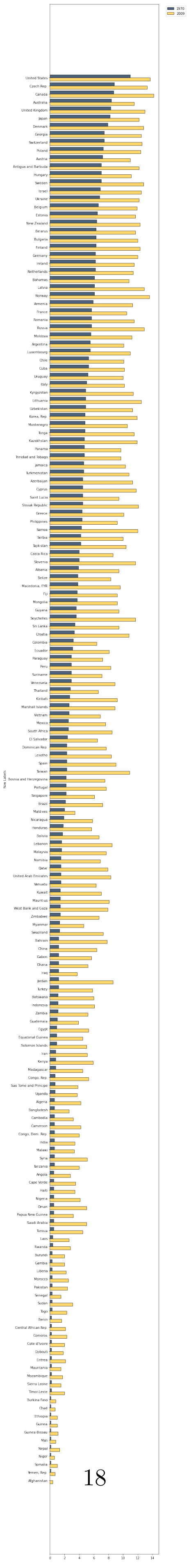
```
[40]: ('Row Labels', '1970', '2009')
```

```
[41]: check('tests/q52.py')
```

```
[41]: <gofer.ok.OKTestsResult at 0x7f4540a12d10>
```

A bar chart for this three-column table will compare 1970 to 2009 for each country. Everywhere in the world, the average number of years that women attend school has increased, in some cases dramatically!

```
[42]: top_1970_with_2009.barh('Row Labels')
```


1970
2009
United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Taiwan
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan
New Labels
0
2
4
6
8
10
12
14

The Gapminder data browser includes many other tables that you can explore as well. For more information on how to load a table from the web, try the course discussion forum.

1.6 6. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the autograder does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this lab. When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

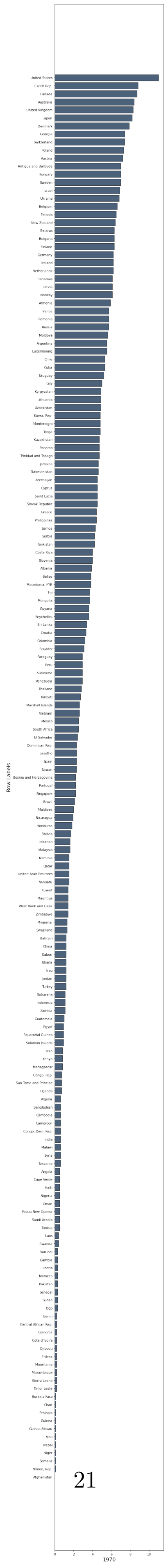
The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

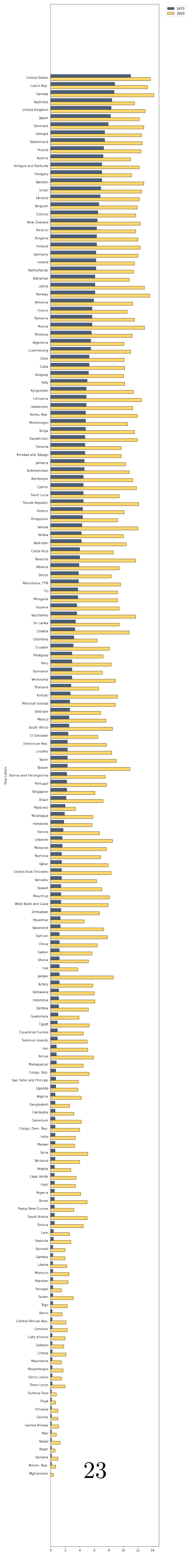
Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

[43]:
```python
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('lab01.ipynb', sorted(glob.glob('tests/q*.py'))))
```

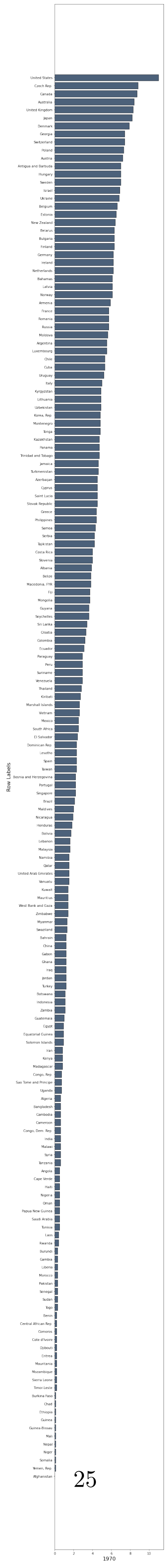
```
2
The biggest distance from the average height among these two people is 0.478
meters.
['tests/q22.py', 'tests/q231.py', 'tests/q232.py', 'tests/q311.py',
'tests/q41.py', 'tests/q411.py', 'tests/q51.py', 'tests/q52.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7f4524dcadd0>

Question 2:

<gofer.ok.OKTestsResult at 0x7f4524dd0c50>

Question 3:

<gofer.ok.OKTestsResult at 0x7f4524dd0f10>

Question 4:

<gofer.ok.OKTestsResult at 0x7f4524dd0e10>

Question 5:
```

```
<gofer.ok.OKTestsResult at 0x7f4524dd0c90>

Question 6:

<gofer.ok.OKTestsResult at 0x7f4524dd0f50>

Question 7:

<gofer.ok.OKTestsResult at 0x7f4524d5bd10>

Question 8:

<gofer.ok.OKTestsResult at 0x7f4524bbc950>

1.0
```

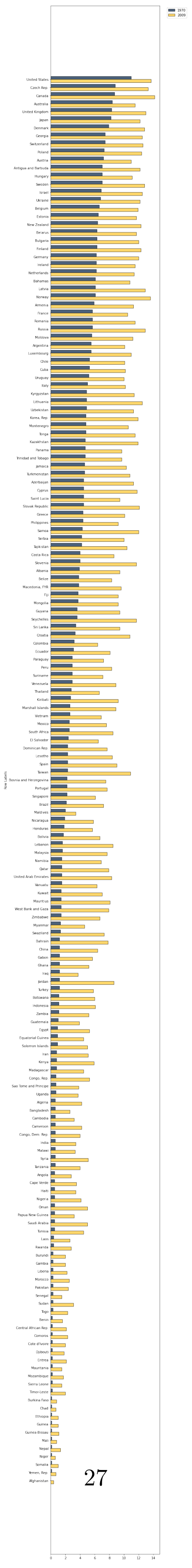

United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Taiwan
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan
Row Labels
0
2
4
6
8
10
1970



1970
2009
United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Taiwan
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan
0
2
4
6
8
10
12
14



Row Labels
1970
0
2
4
6
8
10
United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan



1970
2009
Row Labels
United States
Czech Rep.
Canada
Australia
United Kingdom
Japan
Denmark
Georgia
Switzerland
Poland
Austria
Antigua and Barbuda
Hungary
Sweden
Israel
Ukraine
Belgium
Estonia
New Zealand
Belarus
Bulgaria
Finland
Germany
Ireland
Netherlands
Bahamas
Latvia
Norway
Armenia
France
Romania
Russia
Moldova
Argentina
Luxembourg
Chile
Cuba
Uruguay
Italy
Kyrgyzstan
Lithuania
Uzbekistan
Korea, Rep.
Montenegro
Tonga
Kazakhstan
Panama
Trinidad and Tobago
Jamaica
Turkmenistan
Azerbaijan
Cyprus
Saint Lucia
Slovak Republic
Greece
Philippines
Samoa
Serbia
Tajikistan
Costa Rica
Slovenia
Albania
Belize
Macedonia, FYR
Fiji
Mongolia
Guyana
Seychelles
Sri Lanka
Croatia
Colombia
Ecuador
Paraguay
Peru
Suriname
Venezuela
Thailand
Kiribati
Marshall Islands
Vietnam
Mexico
South Africa
El Salvador
Dominican Rep.
Lesotho
Spain
Taiwan
Bosnia and Herzegovina
Portugal
Singapore
Brazil
Maldives
Nicaragua
Honduras
Bolivia
Lebanon
Malaysia
Namibia
Qatar
United Arab Emirates
Vanuatu
Kuwait
Mauritius
West Bank and Gaza
Zimbabwe
Myanmar
Swaziland
Bahrain
China
Gabon
Ghana
Iraq
Jordan
Turkey
Botswana
Indonesia
Zambia
Guatemala
Egypt
Equatorial Guinea
Solomon Islands
Iran
Kenya
Madagascar
Congo, Rep.
Sao Tome and Principe
Uganda
Algeria
Bangladesh
Cambodia
Cameroon
Congo, Dem. Rep.
India
Malawi
Syria
Tanzania
Angola
Cape Verde
Haiti
Nigeria
Oman
Papua New Guinea
Saudi Arabia
Tunisia
Laos
Rwanda
Burundi
Gambia
Liberia
Morocco
Pakistan
Senegal
Sudan
Togo
Benin
Central African Rep.
Comoros
Cote d'Ivoire
Djibouti
Eritrea
Mauritania
Mozambique
Sierra Leone
Timor-Leste
Burkina Faso
Chad
Ethiopia
Guinea
Guinea-Bissau
Mali
Nepal
Niger
Somalia
Yemen, Rep.
Afghanistan
0
2
4
6
8
10
12
14

Name: Allan Gongora

Section: 0131