CSCI 2000U: Assignment 3

due Nov. 15, 2015, 24:00 EDT

Instructions

- Pull the most recent version of the class `csci-2000-class-resources` repository from GitHub.
- Create the directory `csci-2000-personal/Assignments/Assignment-3/` within `csci-2000-person-al` (your personal repository).
- Create a text file `README.txt` in `csci-2000-personal/Assignments/Assignment-3/` with your name and student number at the top.
- Add the file `README.txt` into version control with the comment `"Created my README.txt file"`
- Commit changes between completing each question, i.e., *there should be five separate commits* (one for each question and one for the file `README.txt`) into your repository when the final solution is committed (you may choose to make more intermediate commits while constructing your solution).
- On completion, your directory `csci-2000-personal/Assignments/Assignment-3/` should contain four IPython Notebook `.ipynb` documents:
  - `pi_approx.ipynb` from Question 1;
  - `medal_pie_chart.ipynb` from Question 2;
  - `average_temperatures.ipynb` from Question 3; and
  - `xy_plot.ipynb` from Question 4.
- Your files and directories *must* be named *exactly* as specified (penalties will be applied).
- In all your Python programs:
  - Put your name and student number in a comment at the top of the program.
  - Provide sufficient documentation to make your program clear and readable.
- To submit your assignment:
  - Edit the `README.txt` file to describe the contents of your directory (i.e., what the files are in the directory). You can also use the `README.txt` file to let us know if you received help from anyone (the instructor, the TA, or any of your peers) and to let us know any other comments you have.
  - Push changes from your local machine to your remote repository `csci-2000-personal` on GitHub.

1. All of the following sequences are sums that converge to the number $\pi \simeq 3.141592653589793$ as $n \to \infty$:

$$\rho_n = 4\sum_{k=1}^{n} \frac{(-1)^{k+1}}{2k-1} = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + \frac{(-1)^{n+1}}{2n-1}\right), \tag{1a}$$

$$\tau_n = \left[6\sum_{k=1}^{n} \frac{1}{k^2}\right]^{\frac{1}{2}} = \left[6\left(\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}\right)\right]^{\frac{1}{2}}, \text{ and} \tag{1b}$$

$$\mu_n = \left[90\sum_{k=1}^{n} \frac{1}{k^4}\right]^{\frac{1}{4}} = \left[90\left(\frac{1}{1^4} + \frac{1}{2^4} + \frac{1}{3^4} + \cdots + \frac{1}{n^4}\right)\right]^{\frac{1}{4}}. \tag{1c}$$

Observe that the sequences converge at distinct rates. Write an IPython Notebook document `pi_approx.ipynb` that computes and prints the sums ρ_n, τ_n, and μ_n in (1) for the values $n = 10, 100, 1000, 10000, 100000$, and 1000000.

- You can use the NumPy function `sum` and vectorised arithmetic in each case to simplify the computations or you can use a more traditional `for` loop to compute the individual sums.
- You should use formatted output (i.e. `%d`, `%f` etc.) to display the computed values. Your output should appear something like this:

```
When n =      10, rho_n = 3.04183961893
When n =      10, tau_n = 3.04936163598
When n =      10,  mu_n = 3.14138462247
  <Intermediate results removed>
When n = 1000000, tau_n = 3.14159169866
When n = 1000000,  mu_n = 3.14159265359
```

- Provide sufficient documentation to make your program clear and readable.

When you are satisfied that your program works as intended, add the document `pi_approx.ipynb` to version control and commit the changes to your local GitHub repository with a suitable comment.

---

2. The 2012 Summer Olympic games were held in London, England. The table below summarises the distribution of gold medals won that year:

USA	CHN	GBR	RUS	KOR	GER	FRA	ITA	HUN	AUS	OTHER
46	38	29	24	13	11	11	8	8	7	107

Use IPython Notebook to construct a pie chart of the gold medal winners at the London Olympics using the data and labels from the preceding table.

- For reproducibility, place all your commands into an IPython Notebook document `medal_pie_chart.ipynb`.
- You will have to use the Pyplot documentation to find a suitable plotting command (you may also have to read the Matplotlib documentation).
- Figure out how to customise the pie chart for best aesthetic effect (e.g., fonts, colours, etc.).
- Add a title to your plot with the text "2012 Olympic gold medals (100XXXXXX)" with your own student number substituted in place of "100XXXXXX".

When you are satisfied that your IPython Notebook document `medal_pie_chart.ipynb` works as intended, add it to version control, and make a commit to your local GitHub repository with a suitable message.

---

3. Copy the data files `*_temperature_2012.dat` into your local directory `csci-2000-personal/Assignments/Assignment-3/` from the directory `csci-2000-class-resources/data/Assignment-3/` (there should be three such files corresponding to Toronto, Montreal, and Vancouver). The files each contain 12 measurements of average monthly temperatures during the year 2012 (as measured at the weather monitoring stations at their respective airports). Your task is to create a IPython Notebook document `average_temperatures.ipynb` that executes a sequence of Python commands that, when executed, generates and customises a plot of this temperature data. Make sure that `average_temperatures.ipynb` is clear, documented, and make sure that the resulting plot is reproducible.

   (a) Load the data from each file into your IPython Notebook workspace with suitable commands. Put comments into your program describing how data is arranged in the data files.

   (b) Plot the temperature data for each location against the number of the month all in a single graph. Choose colours and linestyles to make the curves visibly distinguishable.

   (c) Add a legend to the plot that describes the sources of the data in all the plots. Make sure you position the legend in a place that helps readability, i.e., that minimally obscures the data.

   (d) Place labels on the horizontal and vertical axis to indicate what they mean (i.e., the vertical axis corresponds to the average monthly temperature in Celcius while the horizontal axis corresponds to the month).

   (e) Figure out how to modify the tickmarks on the horizontal axis so that they are labelled precisely with three-letter abbreviations for the months (e.g., Jan., Feb., . . . , Dec.).

   (f) Add a title to your plot with the text "2012 average monthly temperatures (100XXXXXX)" with your own student number substituted in place of "100XXXXXX".

   When finished, commit `average_temperatures.ipynb` to version control.

---

4. One useful feature of NumPy's data parallel model of handling arrays is *logical indexing*. Logical indexing permits selected entries of an array to be accessed on the basis of logical (boolean) conditions *without* using nested `for` and `if` constructs. For example, you can use $y[y <= 0]$ to only pick those elements of the y-array which are negative. In this problem, you will construct a program `xy_plot.ipynb` that will use NumPy's logical indexing to customise a scatter plot of two-dimensional data points. To begin, copy the file `xy_points.dat` (containing 1000 randomly generated (x, y) coordinate pairs with $(x, y) \in [-10, 10] \times [-10, 10]$) from the directory `csci-2000-class-resources/data/Assignment-3/` into your `csci-2000-personal/Assignments/Assignment-3/` directory.

   (a) Load the data from `xy_points.dat` into the IPython Notebook workspace. Store the x-coordinates in a vector `x` and the y coordinates in a vector `y` in the IPython Notebook workspace.

   (b) Plot all the (x, y) coordinate pairs as isolated red dots.

   (c) Use logical indexing to identify all the entries of `x` and `y` that satisfy the inequalities $x < -2$ and $-5 \leq y \leq 0$ simultaneously. Draw blue, filled circles for every point identified in that region.

   (d) Use logical indexing to identify all (x, y) points that lie inside a circle of radius 5 of the point $(x_c, y_c) = (0, 5)$. You will likely have to first construct a new array of the radial distances of the points (x, y) from (x_c, y_c) according to the formula $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Draw green, filled circles for every point identified in that region.

   (e) Customise the axes so that the region viewed is indeed the square region $[-10, 10] \times [-10, 10] \subset \mathbb{R}^2$ and that the coordinate directions are scaled equally. In particular, the circular region should look like a circle.

   (f) Add a title to your plot with the text "Highlighted points (100XXXXXX)" with your own student number substituted in place of "100XXXXXX".

   When finished, commit `xy_plot.ipynb` to version control.