# Practice Midterm Exam for Programming Tools for the Data Scientist

**Name:** _____
**Net ID** _____

## Instructions

There are two sections of this test. The first section consists of 8 multiple choice questions, worth 8 points each, for a total of 64 points points. The second section consists of two longer questions worth 18 points each, for a total of 36 points. Therefore, a perfect score on the test would be 100 points.
At the end of this file there are two blank pages, should you need additional space to add answers (see below). There is also a glossary of defining the Python and terminal commands that we used in this course. As per the instructions, you are permitted to consult this glossary during the exam.
This test booklet is a .pdf file. You should prepare a .pdf file with your answers to submit to Gradescope in the permitted time period. This .pdf can take one of the following forms:

- If you are able to edit .pdf files, you can revise this test booklet and submit that to GradeScope.
- You can print out the test, fill in your answers in pen or pencil (please make the writing dark enough so it is legible). Then you can scan this file and create a .pdf file. You can submit this. Very clear pictures taken with a cell phone camera are acceptable. However, poor quality photographs that the grader cannot read are not acceptable.
- You can create a .pdf file with a Word processor that contains all the answers in sequence, numbered as appropriate. For example, Microsoft Word has a "save as pdf or export to pdf or something like that" option.

**This test has the following time restrictions:**

- This test must be completed less than 24 hours after it is released.
- After downloading the test, you have 1 hour and 15 minutes to complete the test. In addition, there is a 15 minute window in case you have technical difficulties submitting the test.
- The above 1 hour and 15 minute window will be modified as a appropriate, if you submitted an accommodation letter through NYU's Moses Center. For a 1.5 times accommodation, the window should be 1 hour and 55 minutes. For a 2 times accommodation, the window should be 2 hours and 30 minutes. The 15 minute window for technical difficulties will be the same.

This test is **partially** open book. The following are permitted:

- You can look at slides from class during the test.
- You can look at the glossary provided as part of the test. This will be the same glossary as is provided with the practice test.
- You can look at UNIX "man" pages, e.g, type "man ls" in Linux to find out about the "ls" command.
- You can use a simple calculator.
- For Part 2 questions, you can test your code in IDLE or similar IDE for Python. For shell scripts, you can test in a linux shell.

Since the test is partially open book, it is in part **closed** book as well. The following are not permitted:

- Do not communicate with others.

- Do not do websearches.
- Do not look at any materials not specifically listed above as being OK. For example, do not look at your notes.

Please (digitally or manually) sign in any way that you can the following statement:
**I agree to complete this test, independently, in the appropriate amount of time as per the instructions:**

_____

**Answer all questions on the test. There will be opportunities for partial credit on questions.**

# Section 1

**Answer all 8 multiple choice questions**. As per each question, indicate which answers are correct. You can circle the right answers with a pen, or put a dark X or asterisk next to the correct answers. For individual questions, it may be appropriate to either choose a single answer or to choose multiple answers. For example, in sample question **i**, there is a single correct answer, but for sample question **ii**, the correct answer entails choosing more than one of the possibilities enumerated as **a** through **e**. There also may be a different number of options for each question. The correct answers are indicated with blue Xs.

**Sample Question I.** Which of the following is the name of a month in the standard Gregorian Calendar?

(a) March **X**

(b) Earth Day

(c) Wednesday

(d) Noon

(e) Springtime

**Sample Question II.** Which of the following are names months in the standard Gregorian Calendar? Choose all that apply.

(a) March **X**

(b) December **X**

(c) Wednesday

(d) January **X**

(e) Springtime

**Part I Questions 1-8**

1. Which of the following would be an effective way to find "-f" in a man page, e.g., a shell command like "man ls":

   (a) The command: "/-f" (possibly more than once) **X**

   (b) The command "search -f" (possibly more than once)

   (c) The command "f" (possibly more than once)

   (d) The command "F" (possibly more than once)

2. What does $1 mean in a shell script? Choose the best answer.

   (a) One dollar.

   (b) The first parameter on the command line, when you execute the script **X**

   (c) A regular expression for matching a character that is after the end of a string.

3. Given the following regular expression:

   ```
   ([A-Z][a-z.]+ ?){3}
   ```

   Which of the strings below (a–e) does the expression either "match" the whole string, or match a substring. Mark all that apply.

   (a) John Q. Public **X**

   (b) Mary Jean Public **X**

   (c) Pongo, The Dalmatian

   (d) John Smith

   (e) The Great Depression **X**

4. Assuming the present working directory is /home/biden/white/house/ and there is a file with an absolute path of /home/biden/white/house/rooms/kitchen.txt, what is the relative path of that file relative to the present working directory (choose only one answer)?

   (a) kitchen.txt

   (b) rooms/kitchen.txt **X**

   (c) house/rooms/kitchen.txt

   (d) white/house/rooms/kitchen.txt

5. Which of the following are true of simple .csv files (mark all that apply). Note that for purposes of this question, we assume that in a simple .csv file. No field contains a newline character or a comma.

   (a) The file name ends with the string ".csv" **X**

   (b) On each line of the file, fields are delimited by commas. **X**

   (c) The file will be recognized as spreadsheet by standard software including Microsoft Excel and LibreOffice Calc. **X**

   (d) The first line of the file may consist of title headings for each column.**X**

6. The BeautifulSoup function (in the Beautiful Soup module), processes html and similarly formatted file, creating a ”parse” of such files. If the variable ”soup” is set to the output of a call to this BeautifulSoup function, consider the following code:

   **souplist = soup.findall('b')**

   (a) A list of the boldface items from the original text with html markup **X**

   (b) A list of the boldface items from the original text without html markup

   (c) The first boldface item from the original text with html markup

   (d) The first boldface item from the original text without html markup

7. 1000 people filled out questionaires about their personality traits and other information. On these surveys, approximately 100 out of the 1000 people described themselves as “aggressive”. Furthermore, 250 out of the 1000 people preferred the television show “The Walking Dead” to all television shows on a list of shows. Of those 250 people, only 40 people described themselves as “aggressive”. Based on this observation, you hypothesize that watching “The Walking Dead” can be used to predict that a person is less likely to describe themselves as aggressive. What is the best next step from the list below:

   (a) You are done, you have all the evidence you need. 40/250 is significantly less than 250/1000.

   (b) You need to do a new survey with additional people to test if this prediction holds up with new data. **X**

   (c) You should interview the people from the original survey to verify their answer.

8. Given a file ”three_numbers.csv” that consists of the the three lines listed below, suppose you execute the shell command ”grep 1 three_numbers.csv”

   **Contents of three_numbers.csv**

```
41,522,6
200,400,600
1,2,3
```

   Which of the following is true about the output?

   (a) Nothing would print out

   (b) All the lines of the file would print out

   (c) Only columns containing ”1” would print out

   (d) Only the first and third line would print out **X**

# Section 2

**Answer both of the following two questions.** Depending on the question, the answer can be a shell script, a Python program, a regular expression or some other programmatic solution to some problem. You can test the program on a linux terminal or in an IDE like Idle. However, it does not need to be a fully debugged program to get a good score.

**Question 9:** Write a one line shell command that uses **grep** and **cut** to search through a set of files (in the same directory) in the format of the babynames files that we processed in class. Remember those filenames were yob1880.txt, yob1881.txt, ... yob1991.txt. Remember that each line consisted of a name, a gender and a frequency, e.g., 'Joseph,M,57' would mean that there were 57 male babies named Joseph in that year. The script should save the following info in a file "called "female_joes.txt": counts for female babies whose name include either "joe" or "Joe" as a substring. The resulting files should only include numbers (the counts) and no other information.

```
## Assume the argument is a directory name
grep   -i joe $1/yob*.txt |grep ``,F,'' |cut -f3 -d, > female\_joes.txt
```

**Question 10.** Write a python program that converts words in **Base Form** column to words in the **Comparative Form** column, in a way that would be generalizable to new words that may be encountered (that are not in the list). The program should apply endings to the first column words according to rules that are based on the final 1,2,3 or 4 letters. You should take into account the same sort of considerations as with the similar homework problem. Use exception_dictionary to take care of exceptional cases that cannot be covered by rules. Note that your system should calculate that *priceyer* is the comparative of *pricey*, even though the correct comparative is *pricier*.

**exception_dictionary =** {**'bad':'worse','far':'further'**}

**Comparative Adjectives**

| Base Form | Comparative Form |
|---|---|
| bland | blander |
| bright | brighter |
| able | abler |
| bare | barer |
| big | bigger |
| prim | primmer |
| tan | tanner |
| airy | airier |
| beastly | beastlier |
| gluey | glueyer |
| gray | grayer |
| pricey | priceyer |

```
exception_dictionary = {'bad':'worse','far':'further'}

def make_comparative(word):
    if len(word) <= 1:
       return(word+'er' )
    elif word in exception_dictionary:
        return(exception_dictionary[word])
    elif word[-1] == 'e':
        return(word+'r')
    elif word[-1]=='y' and (not word[-2] in 'aeiou'):
        return(word[:-1]+'ier')
    elif (not word[-1] in 'aeiouy') and (word[-2] in 'aeiou'):
        return(word + word[-1] + 'er')
    else:
        return(word + 'er')

for word in ['bland','bright','able','bare','big','prim',\
             'tan','airy','beastly','gluey','gray','pricey']:
    print(word, make_comparative(word))
\
```

**Glossary of Terms**

1. **Python's os module includes global variables like** *os.linesep* **(end of line strings: '$\backslash n$' or '$\backslash r\backslash n$') and** *os.sep* **(path separators – forward slash *'/'* or backward slash '$\backslash$'). The os module also includes functions that interact with the operating system.** *os.getcwd()* **returns the current working directory.** *os.listdir(PATH)* **returns a list of files in PATH;** *os.path.isdir(PATH)* **returns True if PATH is a directory and False otherwise;**
   *os.path.isfile(PATH)* **returns True if PATH is the name of an existing file and False otherwise.**

2. **File Object Streams – Python objects used for reading files and writing to files.**

   - *instream = open('my_file.txt','r')* **sets the variable** *instream* **to the contents of the file** *'my_file.txt'***.** *for* **loops will treat** *instream* **as a list of strings, each ending with** *os.linesep***. For most applications, it makes sense to remove these.**
   - *outstream = open('my_file.txt','w')* **sets the variable outstream to an object that will ultimately be saved as the file** *my_file.txt***. The method** *outstream.write(string)* **will write a string to that file. It is a good idea to include $\backslash n$ anywhere you would like a line break in the file as end of lines are not automatic. $\backslash n$ should be used, rather than os.linesep, even in Windows.**
   - *stream.close()* **will close an opened stream. This ends the connection between Python and a file. In the case of output streams (like** *outstream***), the content of the stream is written to the file.**
   - *with open(file,'r') as instream:* **or** *with open(file,'w') as outstream:* **starts a block in which a stream is opened. The body of code indented under these statements can read from or write to the stream. After the block ends, the stream is closed.**

3. **requests.get(url) is part of the requests library. It produces a input stream containing the content of the url (web address). requests.get(url).text (the .text value of that stream object) is the html text from that website. It is possible to obtain text between  and  or paragraphs and remove all html from that text using other filters. For the class exercises, I wrote such filters using regular expressions and Pythons re package.**

4. **urllib.request.urlopen is a command in the urllib.request library. It is for obtaining text from websites that we used in conjunction with BeautifulSoup.**

5. **bs4.BeautifulSoup (or BeautifulSoup.BeautifulSoup) is a html (and xml) parser that takes two arguments: the input stream produced by urllib.request.urlopen ; and 'lxml' is the name of a library that BeautifulSoup uses to process the html. The soup variable in: soup = bs4.BeautifulSoup(input_stream) is an object that**

**contains the website in a sorted form, so it is possible to lookup html fields. We used it to find all the paragraphs, by means of the command paragraphs = soup.find_all('p'). This produced a list of paragraph structures and assigns them to the variable paragraphs. The text from each paragraph is accessible using .text, e.g., paragrpahs[0].text refers to the text of the first paragraph.**

6. **Python re package includes a variety of ways to use regular expressions. The most basic use is with the command re.search(REGEXP,TEXT_TO_SEARCH). This produces a re.Match object. For example, the command**
   **abc = re.search('(ABC)+','123ABCABCABC123')**
   **returns a regexp object representing the substring ABCABCABC. The object has several values including abc.group(0) (the whole match) and abc.group(1) (the first instance of ABC). The group number N after 0 refers to a piece of the matching regep, matching the part of the regexp starting at the Nth parenthesis. So the 1st parenthesis is around '(ABC) and thus, it corresponds to the first match of the string "ABC". In addition to MATCH.group(number), there is also MATCH.start(N) and MATCH.end(N), referring to the beginning and end of the block of text matching group N. In addition to re.search, other re functions include re.findall, re.finditer and re.sub. re.findall and re.finditer identify multiple instances of a regexp inside a string. re.sub substitutes a regexp with a replacement string.**

7. **Python csv package is a package for processing comma separated value files (and tab separated value files). csv.reader(instream) returns a list of lists from a csv file. Unlike string.split(','), it accounts for more complex csv structure that is used for spreadsheets, e.g., where columns are surrounded by quotes and a field can contain a comma. There are other commands such as csv.writer.writerow that we did not cover in class.**

8. **Python sys.exit and sys.argv. sys.exit exits Python. sys.argv is a list of arguments (strings) taken from command line use of Python. sys.argv[0] is the name of the Python File you are executing and the remaining items in the list are the command line parameters. For example, if you executed the command "python3 do_stuff.py 57 100, sys.argv would be equal to ['do_stuff.py', '57', '100'].**

9. **Linux shell commands:**

   - **ls lists the files in a directory. There are numerous flags (e.g., -l) that you can use to provide additional detail or present the results in different ways.**
   - **cp, mkdir, ln -s are for copying files, making directories and making symbolic links.**
   - **chmod changes the file permissions on a file or directory.**

- **ˆC, kill, fg, bg, ˆZ, &, top, ps, free are commands that have to do with manipulating jobs (computer processes) or finding out information about jobs.**
- **command for printing things to the screen or printing parts of files: echo, cat, grep, cut, sort, uniq**
- **$>$ and $|$ are operators used in shell commands. $|$ directs the output of the process on its left to be input for the process on its right. $>$ sends the output of the process on its left to the filename on its write, creating a file in the process.**
- **Shell scripts are executable files that contain shell commands. Very simple shell scripts can be created which are just sequences of shell commands. In addition, command line parameters can be referred to. $1 is the first parameter on the command line; $2 is the second argument and so on.**

10. **Git commands: git push, git pull, git add, git commit, git clone**
11. **Short guide to regular expresions**
    (a) **Repeat operators: *, +, {number}, {minimum,maxium}**
    (b) **Optional and disjunction: ? and $|$**
    (c) **Parentheses () mark scope of repeat, optional and disjunction operators**
    (d) **Beginning and End of line: ˆand $**
    (e) **Disjunction (and ranges) for single characters: [a-z], [A-Z], [0-9], [a-zA-Z,0-9], [123abc]**
    (f) **Negation for single characters [ˆa-z]**
    (g) **The period . represents any character, e.g., ".*" represents any string. Thus the expression ' ˆA.*Z$' matches any line that starts with A and ends with Z.**