

Pipelines - Input/Output Redirection

Q1.

(a) read-input.py asks for 1000 numbers from stdin. Execute read-input.py and make it say "TRIUMPH!"

(b) Thought question: when you execute read-input.py manually, each input will be on one line, so there would be 1000 lines by the end.

When you use IO redirection to give it input, all of the output is on one line rather than 1000.

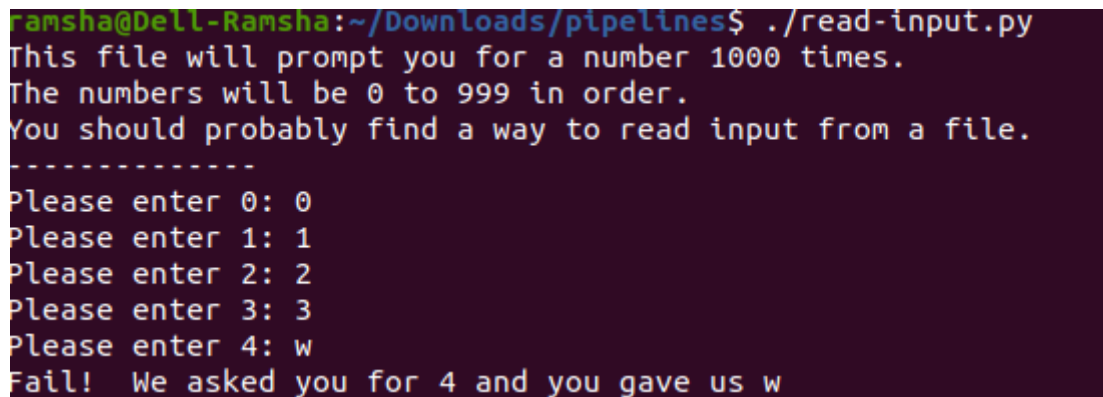
Why is that? Hint: What key do you use to signal a newline character?

Solution:

(a) Before running python file, I ran following command to be able to execute python file

```
chmod 777 read-input.py
```

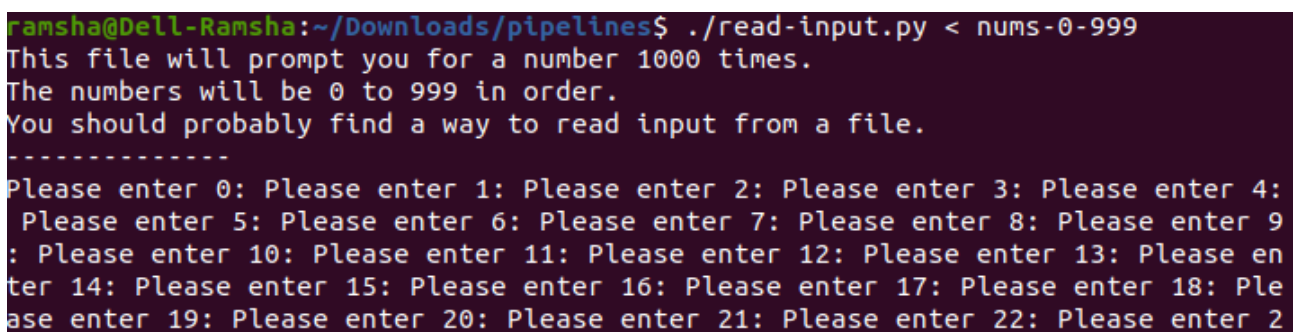
```
sudo ln -s /usr/bin/python3.8 /usr/bin/python2 && sudo ln -s /usr/bin/python3 /usr/bin/python
```



```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ ./read-input.py
This file will prompt you for a number 1000 times.
The numbers will be 0 to 999 in order.
You should probably find a way to read input from a file.
-----
Please enter 0: 0
Please enter 1: 1
Please enter 2: 2
Please enter 3: 3
Please enter 4: w
Fail! We asked you for 4 and you gave us w
```

If I would have entered all digits from 0-999, it would have given "TRIUMPH" as well.

(b) Reading input from file:



```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ ./read-input.py < nums-0-999
This file will prompt you for a number 1000 times.
The numbers will be 0 to 999 in order.
You should probably find a way to read input from a file.
-----
Please enter 0: Please enter 1: Please enter 2: Please enter 3: Please enter 4:
Please enter 5: Please enter 6: Please enter 7: Please enter 8: Please enter 9:
: Please enter 10: Please enter 11: Please enter 12: Please enter 13: Please en
ter 14: Please enter 15: Please enter 16: Please enter 17: Please enter 18: Ple
ase enter 19: Please enter 20: Please enter 21: Please enter 22: Please enter 2
```

Enter “\n” in print() while taking input in script file, it will print the each input line on new line.

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ ./read-input.py < nums-0-999
This file will prompt you for a number 1000 times.
The numbers will be 0 to 999 in order.
You should probably find a way to read input from a file.
-----

Please enter 0:
Please enter 1:
Please enter 2:
Please enter 3:
Please enter 4:
Please enter 5:
```

Q2. List the contents of your home directory and store that in a file

```
ramsha@Dell-Ramsha:~$ ls > ls-output.txt
ramsha@Dell-Ramsha:~$ cat ls-output.txt
conversationscript.sh
Desktop
Documents
Downloads
echoscript.sh
exercise2match.txt
exerciseHistory1.txt
extension.sh
GNU
history
ls-output.txt
mongodb
Music
Pictures
```

List the contents of your home directory and append it to the same file

Command: cat < ls-output.txt > ls-output.txt

Q3. head, tail

- **Print the last three lines of table**
Command: tail -3 table
- **Print the first three lines of table**
Command: head -3 table
- **Print only the second and third lines of table**
Command: sed -n '2,3p' table

Q4. tr

bad-new-line has a weird number of blank lines after each line of text. Create a file good-new-line that doesn't have a weird number of blank lines after each line of text. They should Replace it with just one line.

Command: cat -s bad-new-line

Q5. sort

- We want to know who won the pumpkin size competition (in pumpkinsizes). We are interested in a couple of stats.
 - 1 - the top three winners
 - 2 - the two smallest pumpkins
- Do part a without including the headers!

Solution:

First we remove the header lines from list then sort it and store it in a new file p_sorted.txt

Command: (tail -n +3 pumpkinsizes | sort -nk2 -t'|') > p_sorted.txt

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ (tail -n +3 pumpkinsizes | sort -nk2 -t'|') > p_sorted.txt
ramsha@Dell-Ramsha:~/Downloads/pipelines$ cat p_sorted
cat: p_sorted: No such file or directory
ramsha@Dell-Ramsha:~/Downloads/pipelines$ cat p_sorted.txt
Baz |0
Jill |1
Bill |2
Mir |3
Joe |5
Mar |5
Foo |9
Mary |10
Roe |10
Bar |12
Adam |20
```

Then for 1. The top three winners

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ tail -3 p_sorted.txt
Roe |10
Bar |12
Adam |20
```

2. The two smallest pumpkins

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ head -2 p_sorted.txt
Baz |0
Jill |1
```

Q6. uniq - words-sorted should contain the text of Week 5's lab with one word per line and all of the words sorted. Save all of the unique words to one file.

Command: sort shakes.txt | uniq -c | sort -n > w_sorted.txt

Q7. cut

Normally, `wc` (word count) prints out the number of lines, words, and bytes in a file in addition to the filename. You want to just get the number of words **WITH NO FILENAME**.

Command: `wc -w shakes.txt`

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ wc -w shakes.txt
456 shakes.txt
```

Q8. comm

If given two '.c' files, figure out how you would use `comm` to calculate the ratio of the number of lines that are in both files to the number of lines that are unique to a file (bonus points if you do this as one command!)

Command: `echo `echo scale=1.5\;`wc -l prog1.c | awk '{print $1 }' ^ wc -l prog2.c | awk '{print $1 }' ^ | bc`

```
ramsha@Dell-Ramsha:~/Downloads/pipelines$ echo `echo scale=1.5\;`wc -l prog1.c |
awk '{print $1 }' ^ `wc -l prog2.c | awk '{print $1 }' ^ | bc
1.4
```

GREP

Q1: Your task will be to make a regular expression that matches all of the well-formed phone numbers in every file in the directory.

numbers1 - Search for numbers with dashes (ie, 123-456-7890)

numbers2 - Search for numbers with dashes and those with no dashes (ie, 1234567890)

numbers3 - Search for numbers with dashes, no dashes, and those with parens (ie, (123)456-7890)

numbers4 - Again, you want dashes, no dashes, parens. But someone malicious introduced malformed output into our files: missing parens (ie, 123456-7890 or 123)456-7890) -- so you SHOULD NOT match malformed numbers.

numbers5 - Dashes, no dashes, parens. Now you have missing parens and missing dashes (ie, 123-4567890) as part of the malformations-- again you SHOULD NOT match malformed numbers.

Solution:

Following command will give desire output on each above given conditions:

```
grep --exclude="out.*" '\([([0-9]\{3\})\|[0-9]\{3\}\)[-]?[0-9]\{3\}[-]?[0-9]\{4\}' * >result.txt
```

Q2. In the match-only-one folder there are three group-references files.

You need to make a regular expression that will match every line in group-references-1 and no lines in group-references-2 or group-references-3.

Then, make one that matches 2 but not 1 or 3.

Then, make one that matches 3 but not 1 or 2.

Do the same for lookarounds.

Solution:

- For group reference match:

```
egrep -i '(.)\{1\}' group-references-1
```

- For lookarounds match:

```
grep -o '[+][0-9]\{12\}' lookarounds-1
```

```
grep -o '[0-9]\{12\}' lookarounds-2
```

```
grep -o '[0-9]\{10\}' lookarounds-3
```

Scripting

1. Write a Basic Script

Write a script that finds how many times the word "love" appears in the file. We'll count it even if it's a part of a word, say "lovely", and also if it's capitalized as in "Love". Verify with us!

If your script isn't already set up this way, make it so that your script can find the number of occurrences of ANY word. HINT: This involves pulling from the arguments to the script.

How many times does the word "thee" appear?

How about the word "to"?

How about the word "eternal"?

Solution:

- Send keyword in command: `./testShakespeare.sh love`
- Write following code in nano mode:
`var=$1`
`grep -c -i "$var" shakes.txt`
- Output:
7
- Similarly, you can write other keywords

2. Write a scripting script

Now write a script that you can run ONCE, which determines the number of occurrences of the words "love", "thee", "to", and "eternal" from shakespeare_sonnets.txt all at once. This is a script that will run a script. It's almost like "meta" scripting. You should only need to run this script once!

Solution:

- Save all the given keywords in sample.txt
- Create a file using vi testShakespeare.sh
- Open file in nano mode
- Write a script as follows:
`cat sample.txt "while read line;`
`do grep -c -i "$line" shakes.txt >> output.txt;`
`done`
`cat output.txt`
- Now run the file using command : `./testShakespeare.sh`
- Output:
7
2
14
2