SI 506: Programming I

Fall 2019

Lecture 05

Anthony Whyte <arwhyte@umich.edu>
Lecturer III, School of Information
715 N. University Ave, Ann Arbor, MI 48109
Roumanis Square, 2nd floor ("the loft")

preliminaries

Office Hours

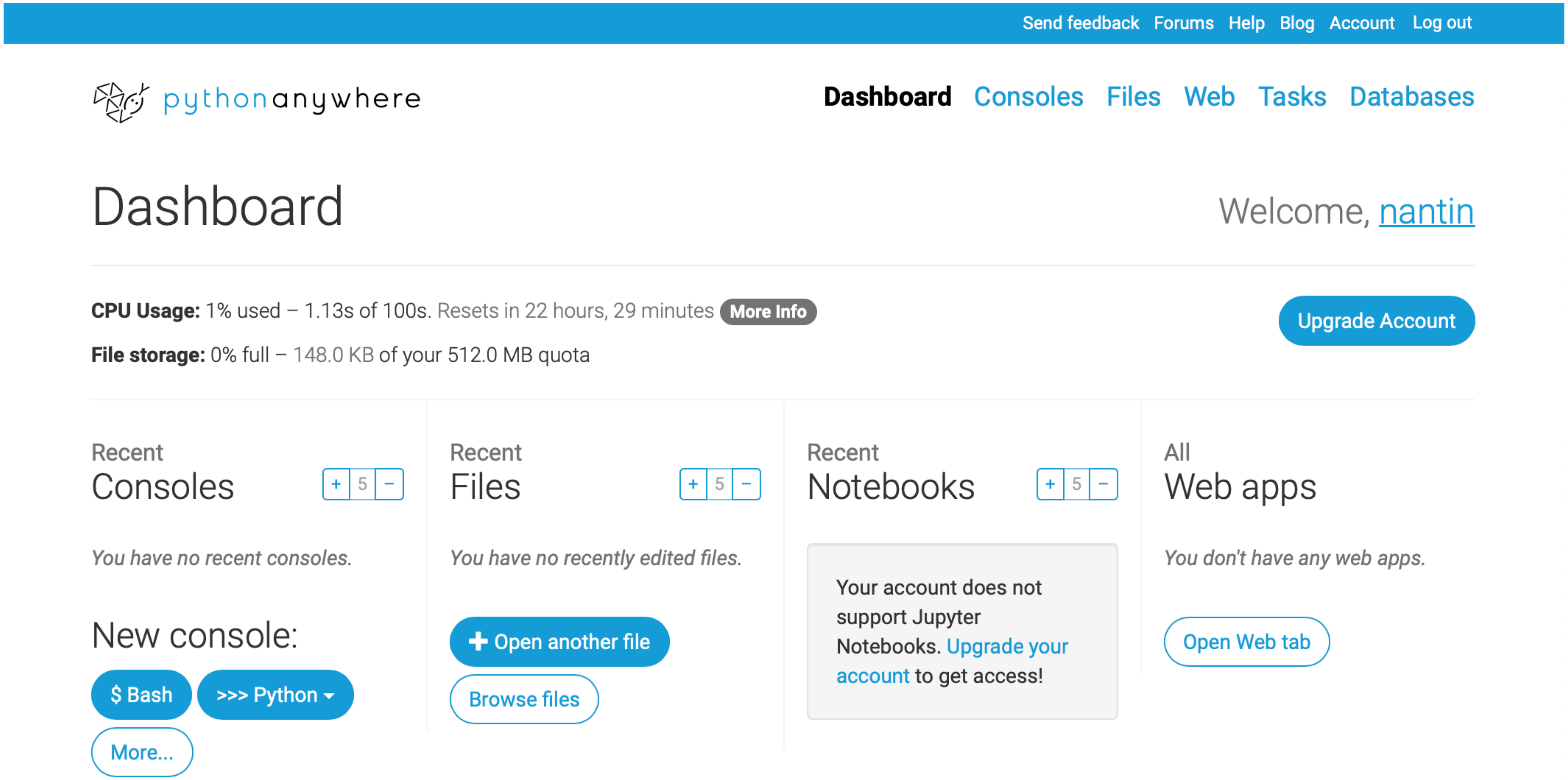
arwhyte

Friday, 11:30 am - 1:00 PM
NQ 3330

Starts 20 Sept 2019
(next week)

Get ready to code

Start your Python console



Dashboard

Welcome, nantin

CPU Usage: 1% used – 1.13s of 100s. Resets in 22 hours, 29 minutes More Info

File storage: 0% full – 148.0 KB of your 512.0 MB quota

Recent Consoles

You have no recent consoles.

New console:

Recent Files

You have no recently edited files.

Recent Notebooks

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All Web apps

You don't have any web apps.

cc BY NC

Lists
characteristics

collection

ordered

mutable

Lists: built in functions

list object methods

list.append(item)
list.extend(iterable)
list.insert(index, item)
list.remove(item)
list.pop(index)
list.index(item, start[,end])
list.count(item)
list.sort(key=None, reverse=False)
list.reverse()
list.copy()
list.clear()

Source: https://docs.python.org/3/tutorial/datastructures.html

BY NC

UMSI

List Exercise

Get file from Canvas; work on in Python Anywhere

1. Get `band_template.py`
2. Create `SI506/lectures` folder
3. Upload file to folder

String formatting

“f” strings

```python
print(f"Band personnel: {band}")
```

List: slicing

filter list items and add to new list

```
# Drop Brian . . .
gimme_shelter = band[:2] + band[3:]
```

List: appending items

list.append()

```
# Add additional personnel . . .
additional_personnel = []
additional_personnel.append(piano)
additional_personnel.append(percussion)
```

List: extending a list with another list

list.extend()

```
# Add additional personnel . . .
gimme_shelter.extend(additional_personnel)
```

List: accumulator pattern

filter items and add to target list

```
# Use for loop, filter on additional
personnel and add to studio_musicians
# (Accumulator pattern)
studio_musicians = []
for person in gimme_shelter:
    if person not in band:
        studio_musicians.append(person)
```

List: insert item at a given position

list.insert()

```
# Mick's vocals are not enough.
# We need a female vocalist.
# Add Merry Clayton to gimme_shelter
# as second item in list using .insert().
co_lead_vocals = 'merry'
gimme_shelter.insert(1, co_lead_vocals)
```

List: slice again

Get vocalists, add to new list, and print using for loop

```
# Use list slicing to extract Gimme Shelter
vocalists (Keith also sings backup)
gimme_shelter_vocalists = gimme_shelter[:?]

# Use for loop to print out vocalists

for vocalist in gimme_shelter_vocalists:
    print(vocalist)
```

List: slice again

Get vocalists, add to new list, and print using for loop

```
# Use list slicing to extract Gimme Shelter
vocalists (Keith also sings backup)
gimme_shelter_vocalists = gimme_shelter[:3]

# Use for loop to print out vocalists

for vocalist in gimme_shelter_vocalists:
    print(vocalist)
```

List: using a counter

counter i in range() to print role & musician using .join()

```
# For each member in the lineup prepend
# their role as <role: > when printing out
# their name

for i in range(len(gimme_shelter)):
    print(''.join([gimme_shelter_roles[i],
                   ': ',
                   gimme_shelter[i]]))
```

List: pop out Brian

list.pop() and list.index()

```
# Band personnel shakeup . . .
# return item with .pop(index)
# using .index() to surface index value.
# Note .pop() also removes item
# from list (very handy in this case)
ex_band_members = []
ex_band_members.append(band.pop(band.index('brian')))
```

BY NC

List: add Mick Taylor

list.insert()

```
# Band personnel shakeup . . .
# add Mick same list position as Brian
band.insert(?, 'mick')
```

List: add Mick Taylor

list.insert()

```
# Band personnel shakeup . . .
# add Mick same list position as Brian
band.insert(2, 'mick')
```

List: count items

list.count()

```python
# How many band members with the name
# Mick are now in the band?
mick_count = band.count('mick')
```

List: Taylor quits band (1974)

Add to ex_band_members; remove from band with del

```
# Add 2nd Mick to ex_band_members then
Use del to remove him from band:
ex_band_members.append(band[2])
del band[2]
```

List: Taylor quits band (1974)

Add to ex_band_members; remove from band with del

```
# Add Ronnie Wood to band in same list
# index position formerly occupied by
# Mick Taylor
band.insert(?, 'ronnie')
```

List: Taylor quits band (1974)

Add to ex_band_members; remove from band with del

```python
# Add Ronnie Wood to band in same list
# index position formerly occupied by
# Mick Taylor
band.insert(2, 'ronnie')
```

List: Ronnie Wood joins band (1975)

list.insert()

```python
# Add Ronnie Wood to band in same list
# index position formerly occupied by
# Mick Taylor
band.insert(2, 'ronnie')
```

List: Bassist Bill Wyman retires (1992)

use list.pop(), list.index() to add to ex_band_members

```
# Bill Wyman retires (1992). Add to ex_band_members
# then remove from band.
ex_band_members.append(band.pop(band.index('bill')))
```

List: reverse item order

list.reverse() in place change

```
# Reverse order in place
# (List drummer first -- as it should be)
band.reverse()
```

List: alpha sort

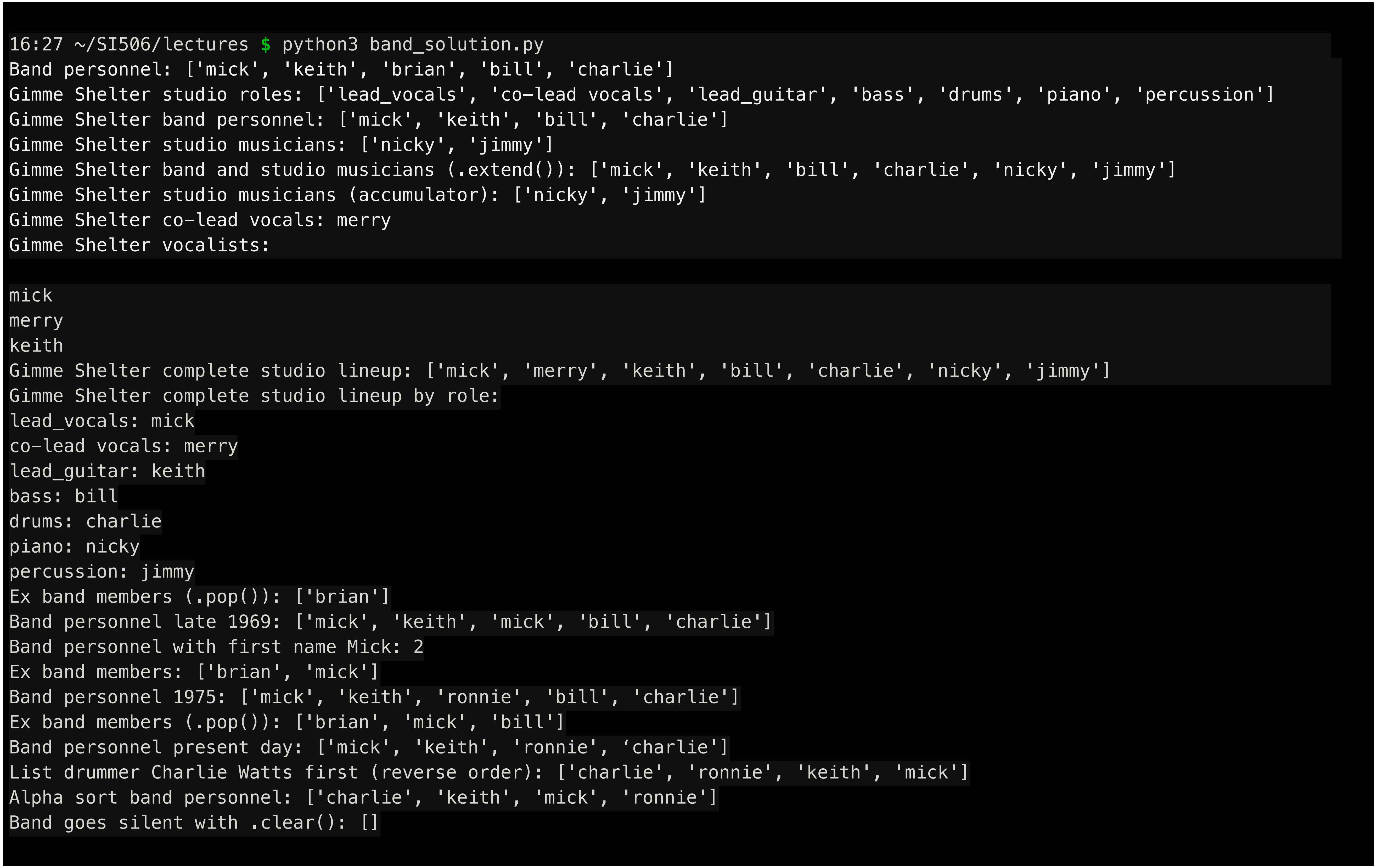
list.sort() in place change

```
# Perform alpha sort in place (Jagger
won't like this)
band.sort()
```

List: clear items

list.clear() in place change

```
# Remove all values in place
band.clear()
```



```
16:27 ~/SI506/lectures $ python3 band_solution.py
Band personnel: ['mick', 'keith', 'brian', 'bill', 'charlie']
Gimme Shelter studio roles: ['lead_vocals', 'co-lead vocals', 'lead_guitar', 'bass', 'drums', 'piano', 'percussion']
Gimme Shelter band personnel: ['mick', 'keith', 'bill', 'charlie']
Gimme Shelter studio musicians: ['nicky', 'jimmy']
Gimme Shelter band and studio musicians (.extend()): ['mick', 'keith', 'bill', 'charlie', 'nicky', 'jimmy']
Gimme Shelter studio musicians (accumulator): ['nicky', 'jimmy']
Gimme Shelter co-lead vocals: merry
Gimme Shelter vocalists:

mick
merry
keith
Gimme Shelter complete studio lineup: ['mick', 'merry', 'keith', 'bill', 'charlie', 'nicky', 'jimmy']
Gimme Shelter complete studio lineup by role:
lead_vocals: mick
co-lead vocals: merry
lead_guitar: keith
bass: bill
drums: charlie
piano: nicky
percussion: jimmy
Ex band members (.pop()): ['brian']
Band personnel late 1969: ['mick', 'keith', 'mick', 'bill', 'charlie']
Band personnel with first name Mick: 2
Ex band members: ['brian', 'mick']
Band personnel 1975: ['mick', 'keith', 'ronnie', 'bill', 'charlie']
Ex band members (.pop()): ['brian', 'mick', 'bill']
Band personnel present day: ['mick', 'keith', 'ronnie', ‘charlie']
List drummer Charlie Watts first (reverse order): ['charlie', 'ronnie', 'keith', 'mick']
Alpha sort band personnel: ['charlie', 'keith', 'mick', 'ronnie']
Band goes silent with .clear(): []
```

finis

directors cut

Lab attendance
small group learning

lab section != lab exercise

- Ask Questions
- Discuss lecture topics
- GSI demos
- Practice coding
- Do lab exercise (extra credit)
- Start problem set
- Help classmates (learn by teaching)

Assignment due dates
weekly problem sets and lab exercises

Available

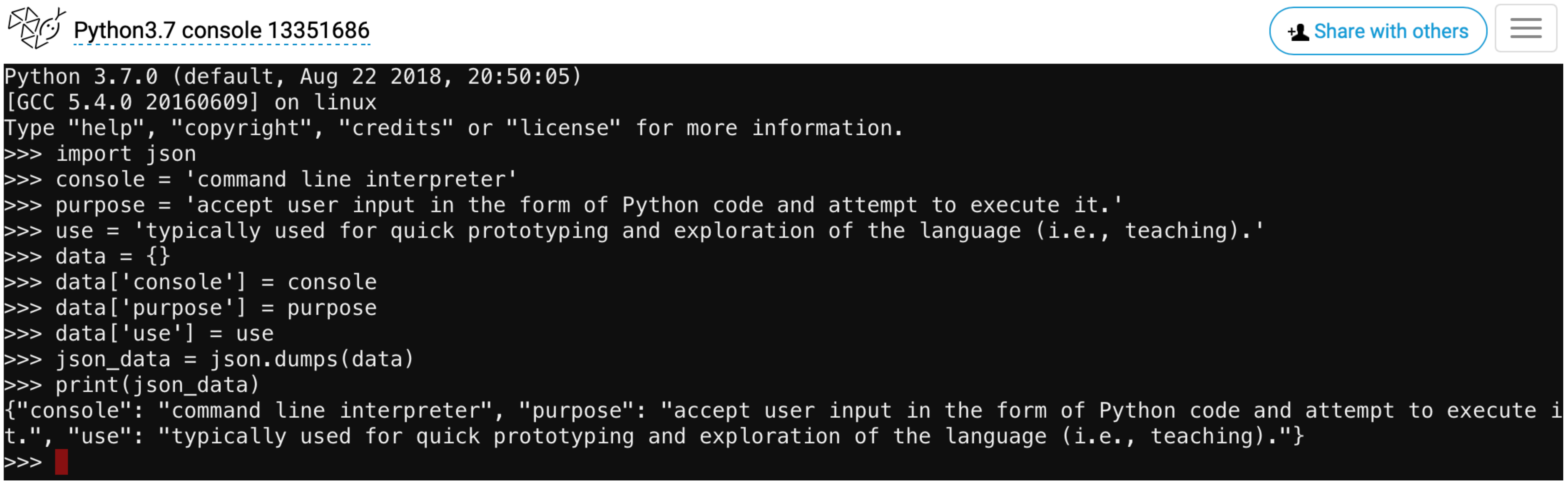
Tuesday, 4:00 PM Eastern

Submission due

following Monday by11:59 PM Eastern

Python console

write/execute Python code (only)



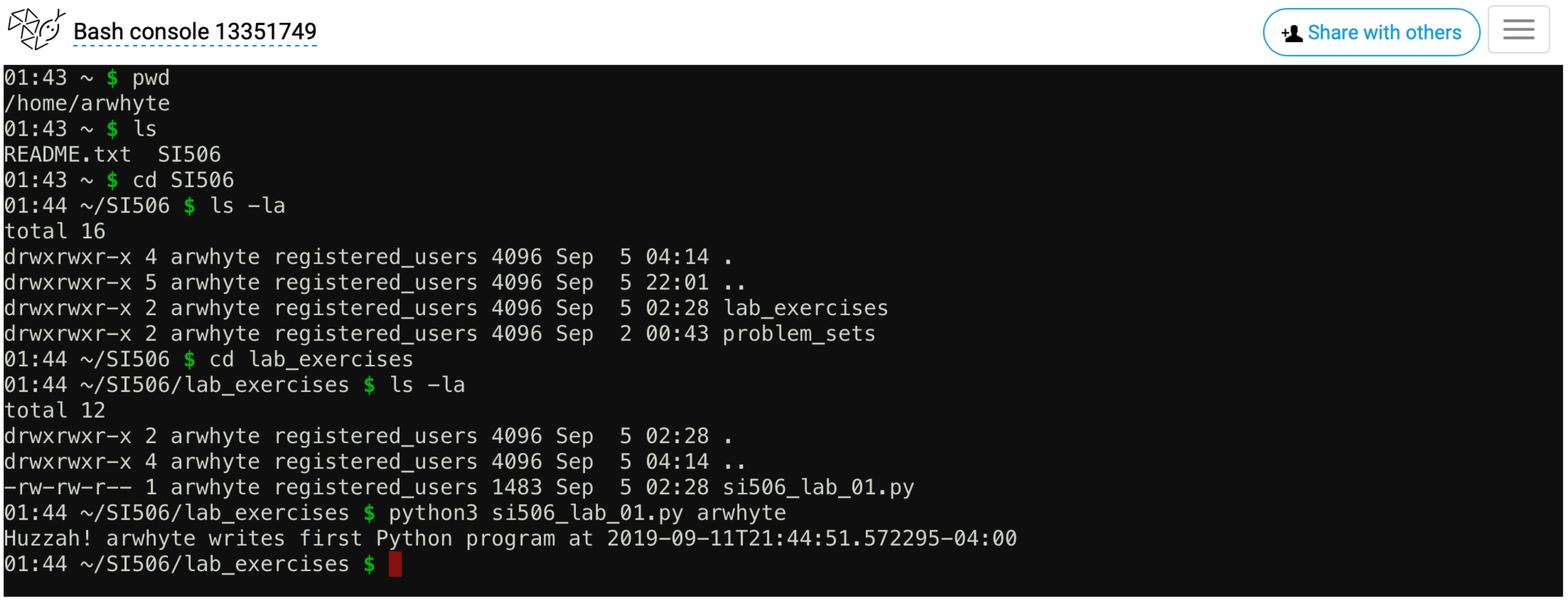
```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> console = 'command line interpreter'
>>> purpose = 'accept user input in the form of Python code and attempt to execute it.'
>>> use = 'typically used for quick prototyping and exploration of the language (i.e., teaching).'
>>> data = {}
>>> data['console'] = console
>>> data['purpose'] = purpose
>>> data['use'] = use
>>> json_data = json.dumps(data)
>>> print(json_data)
{"console": "command line interpreter", "purpose": "accept user input in the form of Python code and attempt to execute i
t.", "use": "typically used for quick prototyping and exploration of the language (i.e., teaching)."}
>>> 
```

cc BY NC

UMSI

Unix shell (Bash)

interact with operating system, issue commands, run scripts



```
01:43 ~ $ pwd
/home/arwhyte
01:43 ~ $ ls
README.txt  SI506
01:43 ~ $ cd SI506
01:44 ~/SI506 $ ls -la
total 16
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 .
drwxrwxr-x 5 arwhyte registered_users 4096 Sep  5 22:01 ..
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 lab_exercises
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  2 00:43 problem_sets
01:44 ~/SI506 $ cd lab_exercises
01:44 ~/SI506/lab_exercises $ ls -la
total 12
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 .
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 ..
-rw-rw-r-- 1 arwhyte registered_users 1483 Sep  5 02:28 si506_lab_01.py
01:44 ~/SI506/lab_exercises $ python3 si506_lab_01.py arwhyte
Huzzah! arwhyte writes first Python program at 2019-09-11T21:44:51.572295-04:00
01:44 ~/SI506/lab_exercises $
```

Keywords

reserved: cannot be used as ordinary identifiers

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Source: *https://docs.python.org/3/reference/lexical_analysis.html?highlight=reserved%20keywords#keywords*