SI 506: Programming I

Fall 2019

Lecture 07

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

Slide deck revisions

errata: corrections and other changes

Slide no(s).	Fix ver.	Description
16	v1p1	Fixed double quotation mark font (replace “ with ").
16-17	v1p1	Changed 2nd print() reference to upper_case; changed to lower_case.
21	v1p1	Fixed single quotation mark font (replace ‘ with ').
28, 30-36	v1p1	Added missing trailing parentheses in last print() statement.
36	v1p1	Removed trailing ‘2’ from function name create_umich_email_address().
	v1p1	Removed three redundant lecture 06 slides from “directors cut” section.

UMSI

preliminaries

Formatted string literal (f-string)

simple syntax, evaluated at runtime; new since 3.6

```
# a string
uniqname = 'arwhyte'

# f-string (formatted string literal)
print(f"My uniqname is {uniqname}\n")
```

prefix

expression

Source: https://www.python.org/dev/peps/pep-0498/

UMSI

List: slicing syntax

lecture 06 correction (lecture 06 slide deck updated)

(end at index -x)

inclusive

```python
# Slice
new_list = band[-x:]
```

inclusive

(start at last index position)

UMSI

List: slicing syntax

lecture 06 correction (lecture 06 slide deck updated)

(end at index -x)
inclusive

```
# Slice
new_list = band[-x:-y]
```

exclusive
(start at index -y - 1)

UMSI

List: index values (+/-)

band list; slide added to lecture 06 slide deck

Mick Jagger	Keith Richards	Brian Jones	Bill Wyman	Charlie Watts
0	1	2	3	4
-5	-4	-3	-2	-1

Index Position (examples)

var	+	-
charlie	band[4]	band[-1]
brian	band[2]	band[-3]
bill	band[3]	band[-2]

List slicing (examples)

var	+	-	Not
charlie	band[4:]	band[-1:]	band[:-1]
mick_keith	band[:2]	band[:-3], band[-5:-3]	
brian_bill	band[2:4]	band[-3:-1]	
not_mick	band[1:]	band[-4:]	band[-4:-1]

Lab exercise: scoring rules

reminder: extra credit rules adjustment

Start: Lab Exercise 04 (**this week**)

Change: extra credit *awarded on points earned* rather than on the attempt.

Rationale: aligns with already adjusted due date (*not* in-class submission; due on/before following Monday, 11:59 PM).

Gradescope: errors

import error: variable not set in uploaded file

ImportError: cannot import name 'great_wall_item_length'

```
Test Failed: Failed to import test module: tests_output
Traceback (most recent call last):
  File "/usr/lib/python3.6/unittest/loader.py", line 428, in _find_test_path
    module = self._get_module_from_name(name)
  File "/usr/lib/python3.6/unittest/loader.py", line 369, in
_get_module_from_name
    __import__(name)
  File "/autograder/source/tests/tests_output.py", line 3, in <module>
    from problem_set_02 import great_wall, great_wall_list,
great_wall_item_length, great_wall_string, \
ImportError: cannot import name 'great_wall_item_length'
```

Problem solving: pseudocode

break problem down into smaller problems or steps

Problem set 02, problem 06 (pseudocode)

```
for site in china_unesco_sites:
    1. site.split() string on delimiter ‘,’ & return a new list
       called site_info
       2. check if ‘Cultural’ in site_info category element:
          if True, then
           3. build a new string using site_info elements
              (extract by site_info[index]), format string per
              instructions
           4. unesco_sites.append() newly formatted string
              to target list
```

while loops

while loop

anatomy

```
while <expression>:
    # Do something
    <statement(s)>
```

while loop

example: definite iteration (i <=10)

```
# Modulus test: if remainder=0 then even, else odd
# Zero is considered an even number, see
# https://en.wikipedia.org/wiki/Parity_of_zero
# Warning: increment the counter; otherwise an
# infinite loop is triggered

i = 0
while i <= 10:
    if i%2 == 0:
        print(f"{i} is an even number")
    else:
        print(f"{i} is an odd number")
    i += 1 # increment counter
```

Truthy / Falsy
defined: truth value testing

"Any object can be tested for truth value, for use in an if or while condition or as [an] operand of the Boolean operations below.

By default, an object is considered true unless its class defines either a __bool__() method that returns False or a __len__() method that returns zero, when called with the object. Here are most of the built-in objects considered false:

- constants defined to be false: None and False.
- zero of any numeric type: 0, 0.0, 0j, Decimal(0), Fraction(0, 1)
- empty sequences and collections: '', (), [], {}, set(), range(0)

Operations and built-in functions that have a Boolean result always return 0 or False for false and 1 or True for true, unless otherwise stated. (Important exception: the Boolean operations *or* and *and* always return one of their operands.)"

Source: https://docs.python.org/3/library/stdtypes.html#truth-value-testing

UMSI

Truthy / Falsy

implications: type this

```
uniqnames = ['arwhyte', 'csev']

# A function
def truth_value(obj):
    if obj:
        return f"{obj} is truthy"
    else:
        return f"{obj} is falsy"


print(truth_value(uniqnames))

uniqnames.clear()

print(truth_value(uniqnames))
```

truth value test

while loop

example: definite iteration (while … else)

```python
# Evaluate loop in a Boolean context
# (truthy if it has elements, falsy otherwise)
# Check case of last element in list, then pop to
# appropriate list
# list.pop() removes element, shrinking list
uniqnames = ['ARWHYTE', 'csev', 'nantin', 'SSCIOLLA', 'zqian']
upper_case = []
lower_case = []

while uniqnames:
    if uniqnames[-1].isupper():
        upper_case.append(uniqnames.pop(-1))
    else:
        lower_case.append(uniqnames.pop(-1))
else:
    print(f"uniqnames empty = {uniqnames}")

print(f"lower_case = {lower_case}")
print(f"upper_case = {upper_case}")
```

truth value test

UMSI

while loop

example: indefinite iteration (true <> false)

```
# Expression True never evaluates to false
# Requires conditional statement that sets break to terminate loop
uniqnames = ['ARWHYTE', 'csev', 'nantin', 'SSCIOLLA', 'zqian']
upper_case = []
lower_case = []

while True:
    if not uniqnames:
        print(f"uniqnames empty = {uniqnames}")
        break
    else:
        if uniqnames[-1].isupper():
            upper_case.append(uniqnames.pop(-1))
        else:
            lower_case.append(uniqnames.pop(-1))

print(f"lower_case = {lower_case}")
print(f"upper_case = {upper_case}")
```

truth value test

terminate loop

UMSI

functions

another slow walk

Functions
defined

A named code block comprising a set of statements designed to perform (ideally) a single task or computation.

When called a function can process values passed to it and can return a value to the caller. Functions are reusable.

Python functions are considered *first-class* objects which means that they can be assigned to variables, stored in data structures, passed as arguments to other functions, nested inside other functions, and returned as values by other functions.

Functions

anatomy

optional

```
def func_name(<arg(s)>):
    # Do something
    <statement(s)>
    return <val>
```

optional

A function is not required to accept arguments (e.g., `func_name()`).
A function without a return statement specified returns None.
A function with a return statement but no value specified returns None.

Functions: exercise

create, call passing in your uniqname, return email address

```
# A simple function
def create_umich_email_address(name):
    return ''.join([name, '@', 'umich.edu'])


# Call simple function
uniqname = 'arwhyte'

umich_email_address = create_umich_email_address(uniqname)

print(f"U-M email address = {umich_email_address}")
```

Functions: example problem

have: uniqnames; need: U-M email addresses

Pseudocode

- Write function that accepts a uniqname and returns a U-M email address
- Loop over source list; for uniqname in list, call function, return email address, and append value to target list

Objects in play

- Default domain ('@umich.edu')
- Source: uniqname list
- Target: email address list

UMSI

Functions: example I

have: uniqnames; need: U-M email addresses

```python
umich_domain = 'umich.edu' # default domain value
uniqnames = ['arwhyte', 'csev', 'nantin'] # source
umich_email_addresses = [] # target


def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    return ''.join([name, '@', domain])


# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

Functions: example I

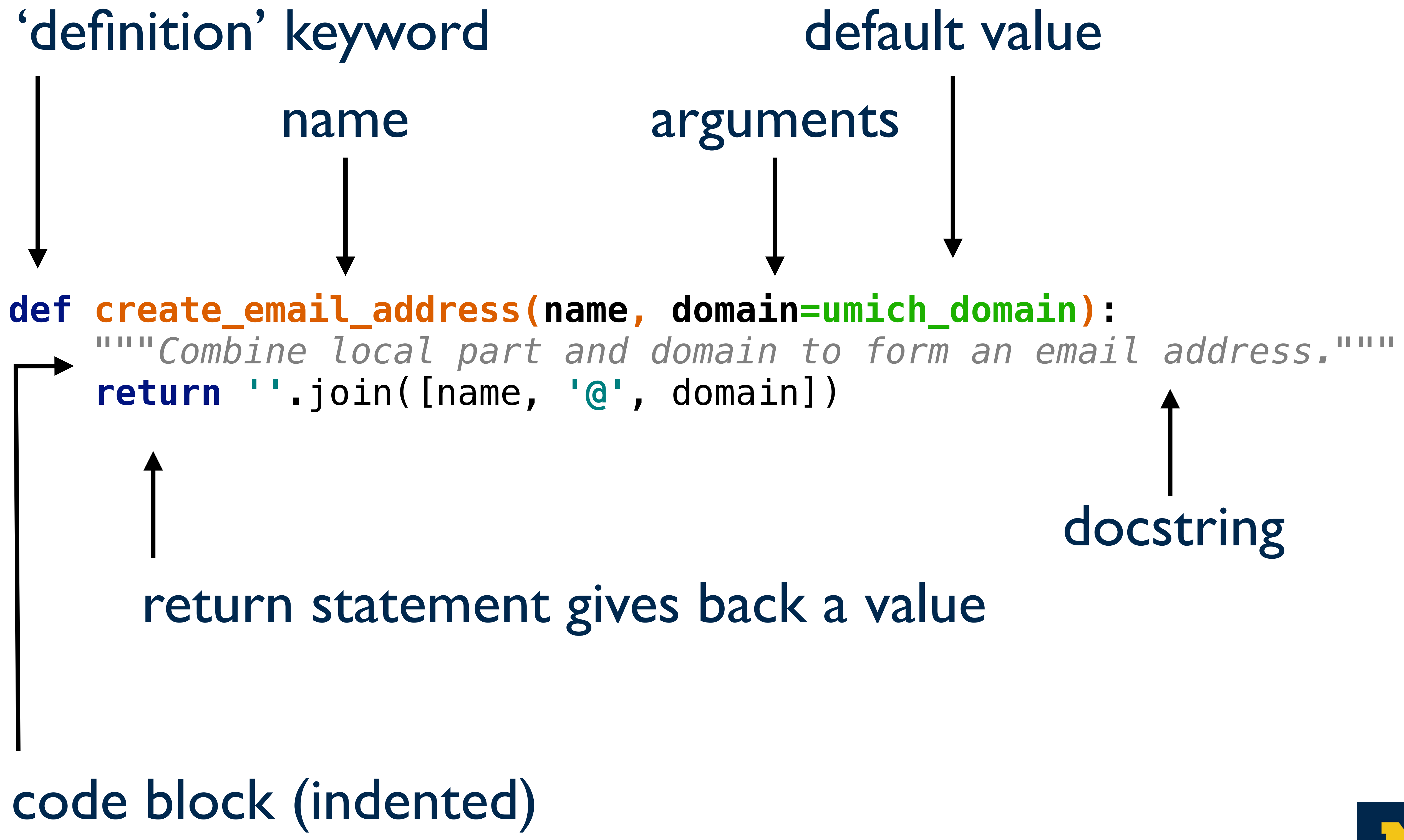
have: uniqnames; need: U-M email addresses

```
umich_domain = 'umich.edu' # default domain value
uniqnames = ['arwhyte', 'csev', 'nantin'] # source
umich_email_addresses = [] # target


def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    return ''.join([name, '@', domain])


# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

UMSI

Functions: example I

anatomy



```python
def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    return ''.join([name, '@', domain])
```

Functions: example I

have: uniqnames; need: U-M email addresses

```
umich_domain = 'umich.edu' # default domain value
uniqnames = ['arwhyte', 'csev', 'nantin'] # source
umich_email_addresses = [] # target


def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    return ''.join([name, '@', domain])


# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

call function

M UMSI

Functions: example 1

have: uniqnames; need: U-M email addresses

```python
umich_domain = 'umich.edu' # default domain value
uniqnames = ['arwhyte', 'csev', 'nantin'] # source
umich_email_addresses = [] # target


def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    return ''.join([name, '@', domain])


# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

UMSI

Functions: example II

have: uniqnames (messy); need: U-M email addresses

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name)
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

Functions: example II

have: uniqnames (messy); need: U-M email addresses

Pseudocode

- Write function that accepts a uniqname and returns a U-M email address
- Guard against all caps/mixed case uniqnames — catch / convert to lower case
- Loop over source list; for uniqname in list, call function(s), return email address, and append value to target list
- If U-M email address encountered in source list accept as is

Functions: example II

have: uniqnames (messy); need: U-M email addresses

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name)
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

UMSI

Functions: example II

have: uniqnames (messy); need: U-M email addresses

```
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name)
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

returns true if string ends with the specified value

UMSI

Functions: example II

add conditional statement to loop

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name)
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

UMSI

Functions: example II

case missed: U-M email address with capitalized chars

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name)  ←——  name.lower() missed
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

UMSI

Functions: example II

refactor this

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine uniqname (convert to lowercase) and
       domain to form an email address."""
    return ''.join([name.lower(), '@', domain])

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    if has_umich_domain(name):
        umich_email_addresses.append(name.lower())
    else:
        umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

.lower()
X2
refactor?

UMSI

Functions: example II

refactor create_email_address() and loop

```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    if has_umich_domain(name):
        email_address = name
    else:
        email_address = ''.join([name, '@', domain])

    return email_address.lower()

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

move domain check here

now called only once

loop simplified

Functions: example II

done for now

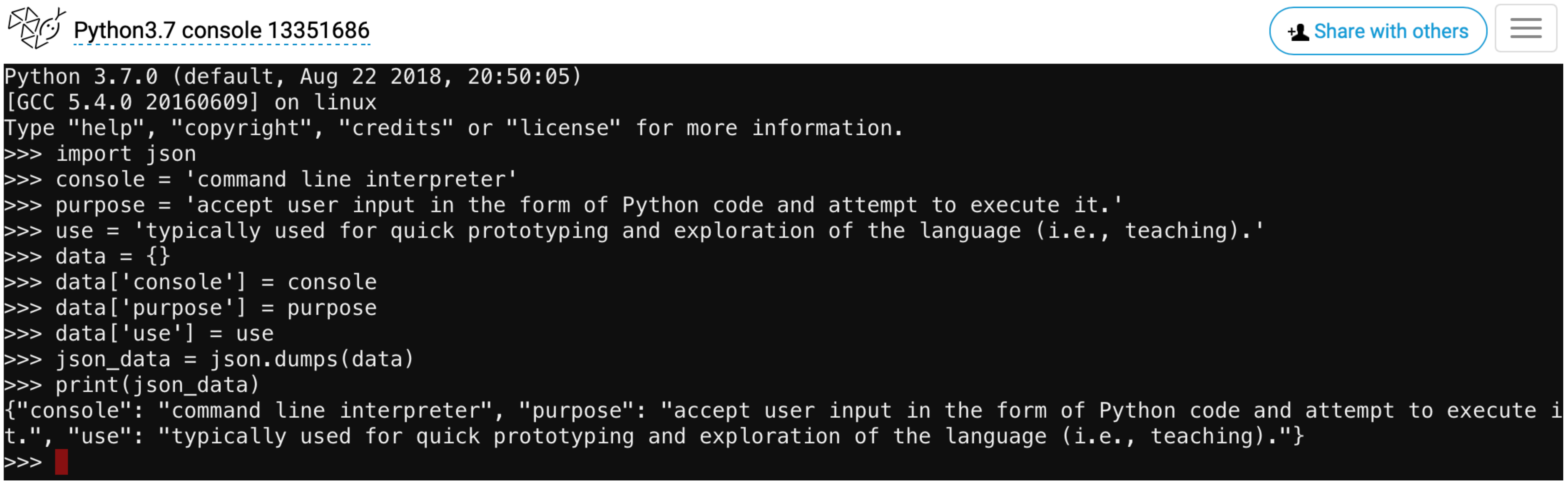
```python
umich_domain = 'umich.edu'
uniqnames = ['arwhyte', 'CSEV', 'nantin', 'ssciolla@umich.edu']
umich_email_addresses = []

def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address."""
    if has_umich_domain(name):
        email_address = name
    else:
        email_address = ''.join([name, '@', domain])

    return email_address.lower()

def has_umich_domain(name):
    """Check if domain suffix already added."""
    return name.endswith(umich_domain)

# Loop over uniqnames list and call create_email_address
for name in uniqnames:
    umich_email_addresses.append(create_email_address(name))

print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```

finis

directors cut

Python console

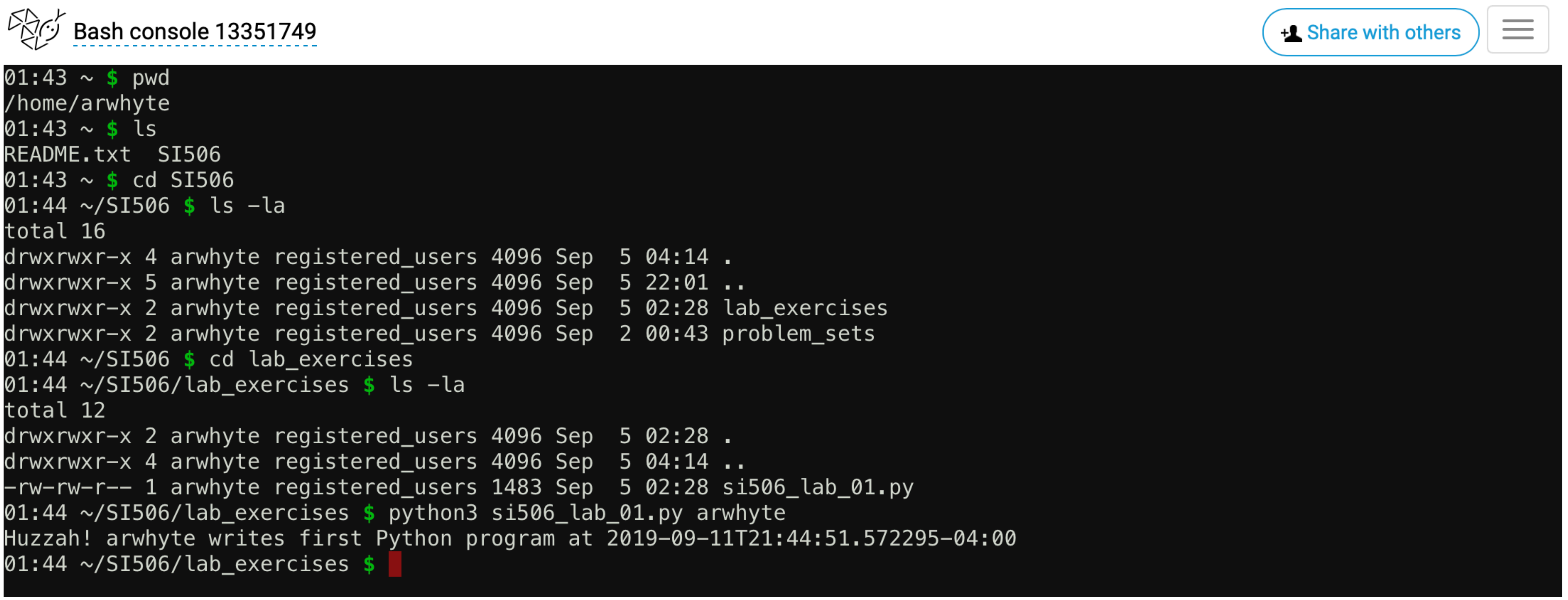
write/execute Python code (only)



```
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)."}
>>> 
```

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*

UMSI

String formatting

I like f-strings (formatted string literal)

```
# old school
print("Band personnel\n %s\n" % band)

# str.format()
print("Band personnel\n {0}\n".format(band))

# f-string (formatted string literal)
print(f"Band personnel\n {band}\n")
```

new line

Control flow: continue statement

terminate current loop iteration, proceed to next iteration

```
band_roles =['lead_vocals','lead_guitar',
'rhythm_guitar', 'bass', ‘drums']

gimme_shelter_roles = []
for role in band_roles:
    if role == 'rhythm_guitar':
        continue
    else:
        gimme_shelter_roles.append(role)
```

$\longleftarrow$ terminate current iteration, proceed to next (e.g., skip)

Control flow: break statement

terminate loop

```
gimme_shelter_roles =['lead_vocals',
'co-lead_vocals', 'lead_guitar',
'rhythm_guitar', 'bass', 'drums']

for role in gimme_shelter_roles:
    if 'vocals' in role:        ←――――――  contains
        print(role)
    else:
        print('\n')
        break        ←――――――  loop terminates
```

Assignment due dates

weekly problem sets and lab exercises

Available

Tuesday, 4:00 PM Eastern

Submission due

following Monday by11:59 PM Eastern

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)

Office Hours

arwhyte

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

Starts 20 Sept 2019
(next week)