SI 506: Programming I

Fall 2019

Lecture 06

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

UMSI

Slide deck revisions

errata: corrections and other changes

Slide no(s).	Fix ver.	Description
20-22	v1p1	corrected list slicing inclusive/exclusive highlight errors (reversed)
23	v1p1	Added slide illustrating index positional values and slicing examples.

UMSI

Revision note

slide deck updated: 24 Sept 2019

slides 20-22: corrected list slicing inclusive/exclusive errors

slide 23: new slide

UMSI

preliminaries

Week 2 scores

Canvas grade book updated

Lab Exercise 02 and Problem Set 01 scores transmitted successfully

UMSI

Lecture recordings

available (raw, unedited, clipping required)

SI 506 001 FA 2019 › SI 506 001 FA 2019

Fall 2019

- Home
- Syllabus
- Announcements
- Modules
- Piazza
- Assignments
- Gradescope
- Grades
- People
- Pages
- Files
- **Lecture Recordings**
- Quizzes

SI 506 001 Fall 2019

Manage Recordings

Click on a PLAY button below to view the selected media.

PLAY	**SI 506: Lecture 05** 9/17/2019 • 3:58 PM
PLAY	**SI 506: Lecture 04** 9/12/2019 • 3:58 PM
PLAY	**SI 506: Lecture 03** 9/10/2019 • 3:58 PM
PLAY	**SI 506: Lecture 02** 9/5/2019 • 3:58 PM

Please contact us if you have any problems, suggestions, or feedback.

UMSI

Lab exercise

extra credit rules adjustment

Start: Lab Exercise 04 (next 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).

UMSI

Gimme Shelter revisited

reading code

UMSI

List Exercise

Get file from Canvas; work on in Python Anywhere

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

Get ready to code

Start your Python console

Send feedback Forums Help Blog Account Log out

pythonanywhere

Dashboard Consoles Files Web Tasks Databases

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

Upgrade Account

Recent
Consoles

+ 5 –

You have no recent consoles.

New console:

$ Bash >>> Python

More...

Recent
Files

+ 5 –

You have no recently edited files.

+ Open another file

Browse files

Recent
Notebooks

+ 5 –

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

All
Web apps

You don't have any web apps.

Open Web tab

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

list indexing and slicing

List: indexing quiz

return element by index position for assignment

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Keith
keith = band[1]
        or
keith = band[2]
```


UMSI

List: indexing quiz

return element by index position for assignment

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Keith
keith = band[1]
```

not

```
keith = band[2]
```


UMSI

List: indexing

return element by index position for assignment

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Charlie
charlie = band[4]
          or
Charlie = band[-1]
```


UMSI

List: slicing syntax

returns list comprising subset of parent list elements

(start at index x)

inclusive

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

exclusive

(end at index y - 1)

UMSI

List: slicing syntax

returns list comprising subset of parent list elements

(start at index 0)

inclusive

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

exclusive

(end at index y - 1)

List: indexing quiz

return element by index position for assignment

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', ‘Brian Jones’,
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Charlie
charlie = band[4]
    both work
charlie = band[-1]
```

negative index

UMSI

List: slicing quiz

Gimme Shelter composers (Jagger and Richards)

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return composers
composers = band[:2]
or
composers = band[1:2]
or
composers = band[0:2]
```


UMSI

List: slicing

Gimme Shelter composers (Jagger and Richards)

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return composers
composers = band[:2]
```

not

```
composers = band[1:2]
```

or

```
composers = band[0:2]
```


UMSI

List: slicing syntax

returns list comprising subset of parent list elements

(start at index position)

inclusive

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

inclusive

(end at last index position)

List: slicing syntax

returns list comprising subset of parent list elements

(end at index -x)

inclusive

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

inclusive

(start at last index position)

List: slicing syntax

returns list comprising subset of parent list elements

(end at index -x)

inclusive

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

exclusive

(start at index -y - 1)

UMSI

Title Text

band list indexing and slicing

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]

UMSI

List: slicing quiz

Return Brian and Bill

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Brian & Bill
brian_bill = band[3:]
                or
brian_bill = band[-3:]
                or
brian_bill = band[-3:-1]
```


UMSI

List: slicing quiz

Return Brian and Bill

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return Brian & Bill
brian_bill = band[3:]

brian_bill = band[-3:]

brian_bill = band[-3:-1]
```

List: slicing quiz

Return Brian and Bill

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return composers
brian_bill = band[3:4]
or
brian_bill = band[2:4]
or
brian_bill = band[3:5]
```


UMSI

List: slicing quiz

Return Brian and Bill

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

```
# Return composers
brian_bill = band[3:4]
or
brian_bill = band[2:4]
or
brian_bill = band[3:5]
```


UMSI

List: slicing

make more sense?

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']
```

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

add two lists

UMSI

list mutation

append, extend, insert, pop

List: appending items

list.append()

```
# Studio musicians (piano and percussion)
piano = 'Nicky Hopkins'
percussion = 'Jimmy Miller'

# Add session musicians with .append()
session_musicians = []
session_musicians.append(piano)
session_musicians.append(percussion)
```


UMSI

List: extending a list with another list

list.extend()

```
# Concatenation (new assignment)
new = gimme_shelter + session_musicians

# In place extension
gimme_shelter.extend(session_musicians)
```

List: positional item insertion

list.insert()

```
# Gimme Shelter Session
gimmer_shelter = ['Mick Jagger', 'Keith Richards’,
                  ‘Bill Wyman', 'Charlie Watts’,
                  'Nicky Hopkins’, 'Jimmy Miller']


# Insert Merry between Mick and Keith
co_lead_vocals = 'Merry Clayton’
gimme_shelter.insert(?, co_lead_vocals)
```


UMSI

List: insert() quiz

insert item *before* given index position

```
# Gimme Shelter Session
gimmer_shelter = ['Mick Jagger', 'Keith Richards’,
                  ‘Bill Wyman', 'Charlie Watts’,
                  'Nicky Hopkins’, 'Jimmy Miller']

# Insert Merry between Mick and Keith
co_lead_vocals = 'Merry Clayton'

gimme_shelter.insert(1, co_lead_vocals)
                    or
gimme_shelter.insert(-5, co_lead_vocals)
                    or
gimme_shelter.insert(2, co_lead_vocals)
```


UMSI

List: insert() quiz

insert item *before* given index position

```
# Gimme Shelter Session
gimmer_shelter = ['Mick Jagger', 'Keith Richards’,
                  ‘Bill Wyman', 'Charlie Watts’,
                  'Nicky Hopkins’, 'Jimmy Miller']

# Insert Merry between Mick and Keith
co_lead_vocals = 'Merry Clayton'

gimme_shelter.insert(1, co_lead_vocals)
                    or
gimme_shelter.insert(-5, co_lead_vocals)
                    or
gimme_shelter.insert(2, co_lead_vocals)
```


UMSI

List: remove Brian Jones

list.pop() and list.index()

```
# The Rolling Stones (mid-1969)
band = ['Mick Jagger', 'Keith Richards', 'Brian Jones',
        'Bill Wyman', 'Charlie Watts']


# Band personnel shakeup (summer 1969):
# Brian Jones ousted
ex_members = []
ex_members.append(
                band.pop(
                  band.index('Brian Jones')))
```


UMSI

List: Mick Taylor joins band (1969) quiz

list.insert()

```
# The Rolling Stones (late-1969)
band = ['Mick Jagger', 'Keith Richards’, 'Bill Wyman’,
        'Charlie Watts']
```

```
# Mick Taylor joins the band
# Insert between Keith and Bill
band.insert(?, 'Mick Taylor')
```


UMSI

List: Mick Taylor joins band (1974) quiz

list.insert()

```
# The Rolling Stones (late-1969)
band = ['Mick Jagger', 'Keith Richards’, 'Bill Wyman’,
        'Charlie Watts']
```

```
# Mick Taylor joins the band
# Insert between Keith and Bill
band.insert(2, 'Mick Taylor')
```


UMSI

List: Mick Taylor quits band (1974)

del list[index]

```
# The Rolling Stones (1974)
band = ['Mick Jagger', 'Keith Richards', ‘Mick Taylor',
        'Bill Wyman', 'Charlie Watts']


# Move Taylor to ex_band_members then
# remove him from band using del:
ex_band_members.append(band[2])
del band[2]

OR

ex_band_members.append(
          band[band.index('Mick Taylor')])
del band[band.index('Mick Taylor')]
```

my advice: use .pop() rather than del

UMSI

for loop

for loop

iterate over lists, sets, dictionaries, tuples, and strings

```
# Syntax: for [element] in [sequence]
for person in gimme_shelter:
    # Do something . . .
```

no list indexing variable required (nice)

UMSI

range()

Built-in type: range()

generate sequence of numbers x to y by step

The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.

range(stop)
range(start, stop[, step])

The arguments to the range constructor must be integers (either built-in int or any object that implements the `__index__` special method). If the step argument is omitted, it defaults to `1`. If the start argument is omitted, it defaults to `0`. If step is zero, ValueError is raised.

Source: https://docs.python.org/3/library/stdtypes.html#ranges

UMSI

Built-in type: range()

for loop example

```
# Get length of list (num = 3)
num = len(gimme_shelter_vocalists)

# Loop over list elements
for i in range(num):
    # Do something . . .
```

loop over sequence 0, 1, 2

UMSI

control flow

UMSI

Conditionals: operators

comparison, logical, identity, membership

http://bit.ly/2mjSI4I

UMSI

Control flow: if statement

apply filter with comparison operator ('==')

line continuation character ('\')

```
# Add a knighthood
for person in band:
    if person == 'Mick Jagger':
        band[band.index(person)] = \
            ''.join(['Sir ', person])
```

if statement (comparison operator '==')

UMSI

Control flow: if statement

apply filter with membership operator; add to target list

```
# Remove elements
session_musicians.clear()

# Accumulator pattern w/membership check
for person in gimme_shelter:
    if person not in band:
        session_musicians.append(person)
```

conditional statement (membership operator 'not in')

UMSI

Control flow: if-else statement

apply comparison operator ('==')

index incremented dynamically

```
# Match role to musician
num = len(gimme_shelter)
for i in range(num):
    # Join session role and musician dynamically
    musician = ''.join([gimme_shelter_roles[i],
                        ': ',
                        gimme_shelter[i]])
    if i == num - 1:
        print(musician, '\n') # new line
    else:
        print(musician)
```

conditional if-else statement (comparison operator '==')

UMSI

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 ←——— terminate current iteration,
                      proceed to next (e.g., skip)
    else:
        gimme_shelter_roles.append(role)
```


UMSI

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:
        print(role)
    else:
        print('\n')
        break
```

contains

loop terminates

UMSI

list in place functions

sort, reverse, clear

UMSI

List: reverse order

list.reverse() in place re-ordering

```
# The Rolling Stones (current lineup)
band = ['Mick Jagger', 'Keith Richards',
        ‘Ronnie Wood', 'Charlie Watts']


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


# The Rolling Stones (Mick won’t like this order)
band = ['Charlie Watts', 'Ronnie Wood',
        ‘Keith Richards', 'Mick Jagger']
```


UMSI

List: alpha sort

list.sort() in place change

```
# The Rolling Stones (Mick won’t like this order)
band = ['Charlie Watts', 'Ronnie Wood',
        ‘Keith Richards', 'Mick Jagger']
```

```
# Perform in place alpha sort
band.sort()
```

```
# Charlie still first
band = ['Charlie Watts', 'Keith Richards',
        'Ronnie Wood', 'Mick Jagger']
```


UMSI

List: clear items

list.clear() in place removal of all elements

```
# Charlie still first
band = ['Charlie Watts', 'Keith Richards',
        'Ronnie Wood', 'Mick Jagger']
```

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

```
# The band goes silent
band = []
```


UMSI

Do these cats next?

Family Marsalis

Ellis (piano), Wynton (trumpet), Branford (saxophone), Delfeayo (trombone), Jason (drums)

UMSI

finis

directors cut

Office Hours

arwhyte

Friday, 11:30 am - 1:00 PM

NQ 3330

Starts 20 Sept 2019

(next week)

UMSI

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)

UMSI

Assignment due dates

weekly problem sets and lab exercises

Available

Tuesday, 4:00 PM Eastern

Submission due

following Monday by11:59 PM Eastern

UMSI

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

UMSI

Python console

write/execute Python code (only)

Python3.7 console 13351686

Share with others

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


UMSI

Unix shell (Bash)

interact with operating system, issue commands, run scripts

Bash console 13351749

Share with others

```
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 $
```


UMSI

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