SI 506: Programming I

Fall 2019

Lecture 08

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
	v1p1	

UMSI

preliminaries

Poll: SI 507 (Winter 2020)

Do you plan on enrolling in SI 507 next semester?

http://bit.ly/2mHahGo

QUESTIONS RESPONSES

Poll: SI 507 (Winter 2020)

The UMSI registrar is engaged in planning for next semester and would like to know if you are planning on enrolling in SI 507 next semester (Winter 2020). Please select the option that best reflects your current plans. The poll is anonymous, gauging your interest only and not your firm commitment to enroll in SI 507 next semester.

Do you plan on enrolling in SI 507 next semester (Winter 2020)?

Multiple choice

- Yes
- No
- Unsure
- Add option or ADD "OTHER"

UMSI

Python Tutor

http://pythontutor.com

Python 3.6

```
1  umich_domain = 'umich.edu' # default domain value
2  uniqnames = ['arwhyte', 'csev', 'nantin'] # source
3  umich_email_addresses = [] # target
4
5
6  def create_email_address(name, domain=umich_domain):
7      """Combine local part & domain."""
8      return ''.join([name, '@', domain])
9
10
11 # Loop over uniqnames list and call create_email_address
12 for name in uniqnames:
13     umich_email_addresses.append(create_email_address(na
14
15 print(f"uniqnames = {uniqnames}\n")
16 print(f"umich_email_addresses = {umich_email_addresses}\
```

Edit this code

line that has just executed

next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First | < Back | Step 14 of 22 | Forward > | Last >>

Print output (drag lower right corner to resize)

Frames | Objects

Global frame

umich_domain	"umich.edu"
uniqnames	
umich_email_addresses	
create_email_address	
name	"csev"

create_email_address

name	"csev"
domain	"umich.edu"
Return value	"csev@umich.edu"

list

0	1	2
"arwhyte"	"csev"	"nantin"

list

0
"arwhyte@umich.edu"

function

create_email_address(name, domain)

default arguments:

domain	"umich.edu"

Video overview: https://youtu.be/McYTtgl8ogI

UMSI

list slicing

another look

UMSI

List slicing: problem 1

extract indices using range() and list slicing

```
regions = ['Eastern Africa', 'Western Africa', 'Southern Africa’]

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

# PROBLEM 1
# Extract indices of Eastern African countries from
# countries_regions list (source) and store in list
# named eastern_african_indices (target)

eastern_african_indices = []

for index in range(len(countries_regions)):
    if countries_regions[index].split(', ')[1] == regions[0]:
        eastern_african_indices.append(index)

print(f"eastern_african_indices = {eastern_african_indices}\n")
```


UMSI

List slicing: problem 1

extract indices using range() and index position

```
regions = ['Eastern Africa', 'Western Africa', 'Southern Africa’]

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

# PROBLEM 1
# Extract indices of Eastern Africa countries from
# countries_regions list (source) and store in list
# named eastern_africa (target)

eastern_african_indices = []

for index in range(len(countries_regions)):
    if countries_regions[index].split(', ')[1] == regions[0]:
        eastern_african_indices.append(index)

print(f"eastern_african_indices = {eastern_african_indices}\n")
```

split

position

index

UMSI

List slicing: problem 2

use indices to identify East African countries

```
countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

eastern_african_indices = [1, 3] # derived from problem 1


# PROBLEM 2
# Use the indices in eastern_african_indices to identify
# East African countries in the countries_regions list
# and then store the country name (only) in the list
# eastern_african_countries.

eastern_african_countries = []

for index in eastern_african_indices:
    eastern_african_countries.append(countries_regions[index].split(', ')[0])

print(f"eastern_african_countries = {eastern_african_countries}\n")
```


UMSI

List slicing: problem 2

use indices to identify East African countries

```
countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

eastern_african_indices = [1, 3] # derived from problem 1


# PROBLEM 2
# Use the indices in eastern_african_indices to identify
# East African countries in the countries_regions list
# and then store the country name (only) in the list
# eastern_african_countries.

eastern_african_countries = []

for index in eastern_african_indices:
    eastern_african_countries.append(countries_regions[index].split(', ')[0])

print(f"eastern_african_countries = {eastern_african_countries}\n")
```

split

position

Index

UMSI

functions

and control flow

UMSI

Functions: quick review

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`.

UMSI

Functions: quick review

anatomy

'definition' keyword

default value

name

arguments

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

docstring

return statement gives back a value

code block (indented)

UMSI

Functions: quick review

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 & domain."""
    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

exercise

Canvas: get `problem_06.py`

Functions: problem 3

match ISO codes to countries (suspend disbelief)

```
iso_codes = ['KEN', 'UGA', 'GHA', 'NGA', 'BWA']

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

countries_regions_iso = []


def match_country_to_iso_code(country, codes):
    """Match country to ISO country code. Perform matching by
    comparing the 1st code letter to the 1st country name letter.
    """
    for code in codes:
        if code[0].lower() == country[0].lower():
            return ''.join([country, ', ', code])

    # if no matches function returns None (implicit)


# Loop over countries_regions using sorted() to alpha sort list without altering the original.
for country_region in sorted(countries_regions):
    countries_regions_iso.append(match_country_to_iso_code(country_region, iso_codes))

print(f"countries_regions_iso = {countries_regions_iso}\n")
```


UMSI

Functions: problem 3

match ISO codes to countries (suspend disbelief)

lists of sequences of type str

```
iso_codes = ['KEN', 'UGA', 'GHA', 'NGA', 'BWA']

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

countries_regions_iso = []
```

1st char

1st char

```
def match_country_to_iso_code(country, codes):
    """Match country to ISO country code. Perform matching by
    comparing the 1st code letter to the 1st country name letter.
    """
    for code in codes:
        if code[0].lower() == country[0].lower():
            return ''.join([country, ', ', code])

    # if no matches, returns None (implicit)
```

sorted()

```
# Loop over countries_regions using sorted() to alpha sort list without altering the original.
for country_region in sorted(countries_regions):
    countries_regions_iso.append(match_country_to_iso_code(country_region, iso_codes))

print(f"countries_regions_iso = {countries_regions_iso}\n")
```


UMSI

Functions: problem 3A

match ISO codes to countries (check all code letters)

```
iso_codes = ['KEN', 'UGA', 'GHA', 'NGA', 'BWA']

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

countries_regions_iso = []


def match_country_to_iso_code(country, codes):
    """Match country to ISO country code. Match by 1) comparing 1st code letter to 1st country name letter
        and 2) comparing each subsequent code letter to any country name letter from index position 1 onwards.
    """
    matched = False # flag
    for code in codes:
        if code[0].lower() == country[0].lower():
            matched = True

        for letter in code[1:]:
            if letter.lower() not in country.split(',')[0][1:].lower():
                matched = False

        if matched:
            country = ''.join([country, ', ', code])

    return country


# Loop over countries_regions using sorted() to alpha sort list without altering the original.
for country_region in sorted(countries_regions):
    countries_regions_iso.append(match_country_to_iso_code(country_region, iso_codes))

print(f"countries_regions_iso = {countries_regions_iso}\n")
```


UMSI

Functions: problem 3A

match ISO codes to countries (check all code letters)

```
def match_country_to_iso_code(country, codes):
    """Match country to ISO country code. . . .
    """
    matched = False # flag
    for code in codes:
        if code[0].lower() == country[0].lower():
            matched = True

        for letter in code[1:]:
            if letter.lower() not in country.split(',')[0][1:].lower():
                matched = False

        if matched:
            country = ''.join([country, ', ', code])

    return country
```

boolean flag

2nd & 3rd chars

if True:

split & index & slice

UMSI

Split, index, slice

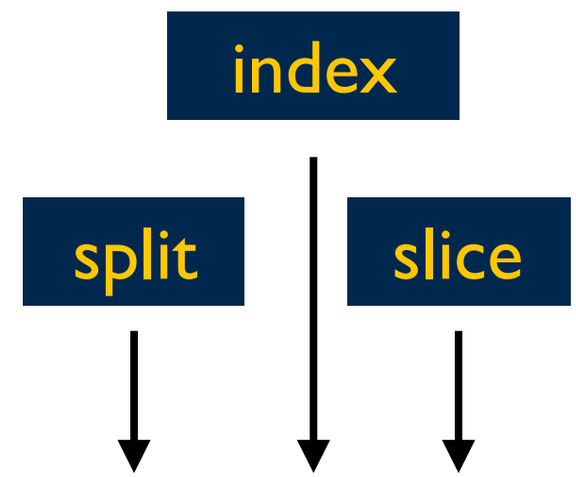
daisy chaining operations on lists and strings

```
def match_country_to_iso_code(country, codes):
    """Match country to ISO country code. . . .
    """
    matched = False # flag
    for code in codes:
        if code[0].lower() == country[0].lower():
            matched = True

    for letter in code[1:]:
        if letter.lower() not in country.split(',')[0][1:].lower():
            matched = False

    if matched:
        country = ''.join([country, ', ', code])

return country
```



UMSI

Functions: problem 4

while loop: identify ISO code that does not end with ‘A’

```
iso_codes = ['KEN', 'UGA', 'GHA', 'NGA', 'BWA']

countries_regions_iso = ['Botswana, Southern Africa, BWA', 'Ghana, Western Africa, GHA',
                         'Kenya, Eastern Africa, KEN', 'Nigeria, Western Africa, NGA',
                         'Uganda, Eastern Africa, UGA’] # from problem 3

# Wrapper function
def format_country_name_iso_code(name, code):
    """Return <country_name> (<iso_code>)"""
    return f"{name} ({code})”


# Use a while loop to iterate the length of countries_regions_iso,
# identify countries with ISO code that does not end in 'A'
# and write to target list countries_last_char_not_a using
# a function that formats the string as "<country_name> (<iso_code>)"

countries_last_char_not_a = []

i = 0
while i < len(countries_regions_iso):
    country_name = countries_regions_iso[i].split(', ')[0]
    iso_code = countries_regions_iso[i].split(', ')[2]
    if iso_code[-1] != 'A':
        countries_last_char_not_a.append(format_country_name_iso_code(country_name, iso_code))
    i += 1

print(f"countries_last_char_not_a = {countries_last_char_not_a}\n")
```

Functions: problem 4

while loop: identify ISO code that does not end with 'A'

```
iso_codes = ['KEN', 'UGA', 'GHA', 'NGA', 'BWA']

countries_regions_iso = ['Botswana, Southern Africa, BWA', 'Ghana, Western Africa, GHA',
                         'Kenya, Eastern Africa, KEN', 'Nigeria, Western Africa, NGA',
                         'Uganda, Eastern Africa, UGA'] # from problem 3

# Wrapper function
def format_country_name_iso_code(name, code):
    """Return <country_name> (<iso_code>)"""
    return f"{name} ({code})"
```

utility function

```
# Use a while loop to iterate the length of countries_regions_iso,
# identify countries with ISO code that does not end in 'A'
# and write to target list countries_last_char_not_a using
# a function that formats the string as "<country_name> (<iso_code>)"

count                                   = []
```

last char

```
i = 0
while i < len(countries_regions_iso):
    country_name = countries_regions_iso[i].split(', ')[0]
    iso_code = countries_regions_iso[i].split(', ')[2]
    if iso_code[-1] != 'A':
        countries_last_char_not_a.append(format_country_name_iso_code(country_name, iso_code))
    i += 1

print(f"countries_last_char_not_a = {countries_last_char_not_a}\n")
```

split & index

UMSI

Functions: problem 5

char count of country names (ISO code ends with ‘A’)

```
countries_regions_iso = ['Botswana, Southern Africa, BWA', 'Ghana, Western Africa, GHA',
                         'Kenya, Eastern Africa, KEN', 'Nigeria, Western Africa, NGA',
                         'Uganda, Eastern Africa, UGA']

# Write two utility functions that 1) return the length of an object
# (e.g., list) and 2) return the last character in a value.

def count_chars(val):
    """Wrapper function. Return length of value."""
    return len(val)


def last_char(val):
    """Wrapper function. Return last character in value."""
    return val[-1]


# Loop over country_regions_iso list and count the number of characters
# in country names (only), filtering on ISO codes that ends in 'A'.
# Call utility functions while looping over the country_regions_iso elements.
# Print each country_name char count as "<country_name> = <char_count> chars"
# Then print the total count as "total chars = <num_chars>"
num_chars = 0
for country in countries_regions_iso:
    if last_char(country.split(', ')[2]) == 'A':
        country_name = country.split(', ')[0]
        print(f"{country_name} = {count_chars(country_name)} chars")
        num_chars += count_chars(country_name)

print(f"total chars = {num_chars}")
```


UMSI

Functions: problem 5

char count of country names (ISO code ends with 'A')

```
countries_regions_iso = ['Botswana, Southern Africa, BWA', 'Ghana, Western Africa, GHA',
                         'Kenya, Eastern Africa, KEN', 'Nigeria, Western Africa, NGA',
                         'Uganda, Eastern Africa, UGA']

# Write two utility functions that 1) return the length of an object
# (e.g., list) and 2) return the last character in a value.

def count_chars(val):
    """Wrapper function. Return length of value."""
    return len(val)


def last_char(val):
    """Wrapper function. Return last character in value."""
    return val[-1]


# Loop over country_regions_iso list and count the number of characters
# in country names (only), filtering on ISO codes that ends in 'A'.
# Call utility functions while looping over the country_regions_iso elements.
# Print each country_name char count as "<country_name> = <char_count> chars"
# Then print the total count as "total chars = <num_chars>"
num_chars = 0
for country in countries_regions_iso:
    if last_char(country.split(', ')[2]) == 'A':
        country_name = country.split(', ')[0]
        print(f"{country_name} = {count_chars(country_name)} chars")
        num_chars += count_chars(country_name)

print(f"total chars = {num_chars}")
```

utility function

utility function

split & index

UMSI

finis

directors cut

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
```


UMSI

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

UMSI

while loop

example: definite iteration (while . . . else)

```
# 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: exercise

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

UMSI

Functions: exercise

generate list of email addresses from uniqname

```
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")
```


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

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).

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

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

Office Hours

arwhyte

Friday, 11:30 am - 1:00 PM

NQ 3330

Starts 20 Sept 2019

(next week)

UMSI