# SI 506: Programming I Fall 2019

#### Lecture 17

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





#### Exercises

#### paths

```
lectures/lecture 16/
  lecture_16_exercise.py
  scary films.json
lectures/lecture 17/
  lecture 17 exercise solution.py
  un-americas country areas dev.csv
  un-americas_country areas.csv
  un-americas_regions_country_areas.csv
  world bank americas economies.csv
```





# tuples





# Tuples data type characteristics

- sequence: siblings: list, string
- ordered: access items via indexing, slicing
- immutable: item reassignment not permitted
- comparable: comparison operator friendly
- hashable: can be used as dictionary keys





### Tuple create (packing)

```
>>> jamaica = ('Jamaica', 'JAM', 'Kingston')
>>> type(jamaica)
<class 'tuple'>
                           iterable
>>> cuba = tuple(['Cuba', 'CUB', 'Havana'])
>>> print(cuba)
('Cuba', 'CUB', 'Havana')
```



single item syntax (trailing comma)

```
>>> puerto rico = ('PRI')
>>> type(puerto_rico)
<class 'str'>
comma
>>> type(puerto_rico)
<class 'tuple'>
```





### Tuple delete

```
>>> del cuba
>>> print(cuba)

Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
NameError: name 'cuba' is not defined
```





use indexing to access individual items

```
>>> print(jamaica[0], jamaica[1], jamaica[2])
```

Jamaica JAM Kingston





use slicing to return a "filtered" tuple

```
>>> print(jamaica[1:])
('JAM', 'Kingston')
>>> print(jamaica[0::2])
('Jamaica', 'Kingston')
```





immutable: item reassignment not permitted

```
>>> jamaica[0] = 'Jamaican Jerk Pit'

Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support
item assignment
```





### Tuple concatenation

```
>>> jamaica = ('Jamaica', 'JAM', 'Kingston')
>>> economy = ('Developing', 'Upper middle income')
>>> jamaica_economy = jamaica + economy
>>> print(jamaica_economy)

('Jamaica', 'JAM', 'Kingston', 'Developing', 'Upper middle income')
```





# Tuple multiplication

```
>>> jamaica * 2
('Jamaica', 'JAM', 'Kingston', 'Jamaica', 'JAM',
'Kingston')
```





### Tuple useful built-in functions

```
>>> len(jamaica)
3
>>> fibonacci = (0, 1, 1, 2, 3, 5, 8, 13, 21)
>>> max(fibonacci)
21
>>> min(fibonacci)
```





# Tuple comparison operators

```
>>> (0, 1, 2) < (2, 3, 4)
True

>>> (0, 5, 10) < (0, 4, 9)
False
```





# Tuple unpacking

```
>>> (country, iso_alpha3_code, capital) = jamaica
>>> print(country)
Jamaica
>>> print(iso alpha3 code)
JAM
>>> print(capital)
Kingston
>>> country, iso_alpha3_code, capital = cuba
```





### Tuple convert to a list

```
>>> jamaica = list(jamaica)
>>> type(jamaica)
<class 'list'>
>>> print(jamaica)
['Jamaica', 'JAM', 'Kingston']
```





#### Dictionaries and tuples

dict.items() returns tuples

```
json_str = """{
    "name": "Wolverine",
    "powers": ["accelerated_healing", "super_strength",
        "animal_oriented_powers", "enhanced_senses",
        "regeneration", "natural_weapons"
hero = json.loads(json_str)
print(hero.items())
output:
dict_items([('name', 'Wolverine'),
('powers', ['accelerated_healing', 'super_strength',
'animal_oriented_powers', 'enhanced_senses', 'regeneration',
'natural_weapons'])])
```





#### Dictionaries and tuples

dict.items() returns tuples

```
for k, v in hero.items():
    print(k, v)

output:
name Wolverine

powers ['accelerated_healing', 'super_strength',
'animal_oriented_powers', 'enhanced_senses',
'regeneration', 'natural weapons']
```





# JSON





#### JSON

two structures: object {} (unordered), list [] (ordered)

```
"name": "Black Panther",
"full name": "T'Challa",
"team": "Avengers",
"place_of_birth": {
  "country": "Wakanda",
  "continent": "Africa"
"place_of_death": null,
"powers": ["agility",
   "stealth",
   "weapons_master",
   "enhanced_senses"
"first_appearance": 1966,
"active": true
```

```
{} = unordered set of name/value pairs
```

[] = ordered list of values





#### JSON

data types: string, integer, boolean, null, object, array

```
"name": "Black Panther",
"full name": "T'Challa",
"team": "Avengers",
"place of birth": {
  "country": "Wakanda",
  "continent": "Africa"
"place_of_death": null,
"powers": ["agility",
   "stealth",
   "weapons_master",
   "enhanced senses"
"first_appearance": 1966,
"active": true
```





#### Read a JSON file

use json.load(); returns a dictionary

```
import json
def read data(filename):
    """Get JSON""
    with open(filename, 'r') as file_obj:
        data = json.load(file obj)
    return data
data file = 'data.json'
scary_source_data = read_data(data_file)
```





#### Read a JSON string

use json.loads(): returns a dictionary

```
import json
json_str = """{
  "name": "Black Panther",
  "powers": ["agility",
     "stealth",
     "weapons_master",
     "enhanced senses"
region_counts = json.loads(json_str)
print(f"type = {type(region_counts)}\n")
type = <class 'dict'>
```





#### Write to a JSON file

use json module

```
import json # a treat

def write_data(filename, data):
    """Write dictionary to JSON file"""
    with open(filename, 'w') as file_obj:
        json.dump(data, file_obj, indent=2)

filename = 'characters.json'
write_scary_data(filename, scary_output_data)
```





### CSV





#### CSV file

\*.csv: comma-delimited strings

```
subregion, intermediate region, country area
Latin America/Caribbean, Caribbean, Anguilla
Latin America/Caribbean, Caribbean, Antigua and Barbuda
Latin America/Caribbean, Caribbean, Aruba
Latin America/Caribbean, Caribbean, Bahamas
                                               strings with commas "x, y"
Latin America/Caribbean, Caribbean, Barbados
Latin America/Caribbean, Caribbean, "Bonaire, Sint Eustatius and Saba"
Latin America/Caribbean, Caribbean, British Virgin Islands
Latin America/Caribbean, Caribbean, Cayman Islands
Latin America/Caribbean, Caribbean, Cuba
                                                 special character (UTF-8)
Latin America/Caribbean, Caribbean, Curação
Latin America/Caribbean, Caribbean, Dominica
Latin America/Caribbean, Caribbean, Dominican Republic
Latin America/Caribbean, Caribbean, Grenada
Latin America/Caribbean, Caribbean, Guadeloupe
Latin America/Caribbean, Caribbean, Haiti
Latin America/Caribbean, Caribbean, Jamaica
Latin America/Caribbean, Caribbean, Martinique
Latin America/Caribbean, Caribbean, Montserrat
Latin America/Caribbean, Caribbean, Puerto Rico
```



#### Read a CSV file: return lists

use csv.reader(); rows returned as lists

```
import csv

def read_csv(filename):
    """Read csv file. Specify encoding; filter out byte order
    mark (BOM) \ufeff. """
    data = []

with open(filename, 'r', encoding='utf-8-sig') as file_obj:
    reader = csv.reader(file_obj, delimiter=',')
    for row in reader:
        data.append(row)

return data
```





#### Read a CSV file: return lists

use csv.writer(); write to file row by row

```
import csv

def write_csv(filename, data):
    """Write csv file."""
    with open(filename, 'w') as csv_file:
        csv_writer = csv.writer(csv_file, delimiter = ',')

        csv_writer.writerow(["id", "name"])

        for item in data:
            csv_writer.writerow([item[0], item[1]])
```





#### Read a CSV file; return dictionaries

use csv.DictReader(); rows returned as OrderedDict

```
import csv
def read_csv_to_dict(filename):
    """Read csv file. Generate dictionaries"""
    data = []
    with open(filename, 'r', encoding='utf-8-sig') as file_obj:
        reader = csv.DictReader(file_obj, fieldnames=['country',
                                                'iso_alpha3_code'])
        row = 0
                                                 gimme plain dictionaries
        for dictionary in map(dict, reader):
                                                   ignore header row
            if row > 0:
                 data.append(dictionary)
            row += 1
```

OrderedDict: dictionary that maintains insertion order



return data



### exercise





#### lecture\_I7\_exercise\_solution.py

script anatomy

**CONSTANTS** 

```
load
```

```
import csv
import json
```

```
4
```

```
entry
```

```
call
```

```
main() process, delegate tasks
```

```
format_keys(name)
get_region_count(data, region)
get_country_area_locations(regions, headers, country_area)
get_dev_status(dev_status, country_area)
get_economic_status(world_bank_economies, iso_code)
read_csv(filename, seq_type=SEQ_TYPE[0])
write_json(filename, data)
```

```
if __name__ == '__main__':
    main() # call main method
```





### finis





### directors cut





#### Exceptions

traceback horror

```
Traceback (most recent call last):

File "/path/to/example.py", line 4, in <module>
    greet('Chad')
...

File "/path/to/example.py", line 2, in greet
    print('Hello, ' + someon)

NameError: name 'someon' is not defined
```

Source: <a href="https://realpython.com/python-traceback/">https://realpython.com/python-traceback/</a>





# It: Chapter One (2017) scary clown







## It: Chapter One (2017)

scary movie, scary clown

```
it = {}
it['title'] = 'It: Chapter One'
it['year_released'] = 2017
it['budget'] = 35000000
it['box_office'] = 7000000000
it['scary_character'] = {}. # nested dictionary
it['scary_character']['name'] = 'Pennywise the Dancing Clown'
it['scary_character']['signature_weapon'] = None
```





# Friday the 13th (1980) scary hockey mask







# Friday the 13th (1980)

scary movie, scary hockey mask

```
friday_13th = {
    'title': 'Friday the 13th',
    'year_released': 1980,
    'budget': 5500000,
    'box_office': 59800000,
    'scary_character': {
        'name': 'Jason Vorhees',
        'signature_weapon': 'machete'
    }
}
```







### JSON

Javascript Object Notation (data interchange format)

```
"scary_films": [
    "title": "The Wizard of Oz",
    "year released": 1939,
    "budget": 2800000,
    "box_office": 26100000,
    "scary_character": {
        "name": "The Wicked Witch of the West",
        "signature_weapon": "evil spells"
```

nested object {}





### Read a JSON file

use json module

```
import json # a treat
def read_scary_data(filename):
    """Get JSON""
    with open(filename, 'r') as scary_file_obj:
        data = json.load(scary_file_obj) # a trick
    return data
scary_file = 'scary_films.json'
scary_source_data = read_scary_data(scary_file)
```





## Write to a JSON file

use json module

```
import json # a treat

def write_scary_data(filename, data):
    """Write dictionary to JSON file"""
    with open(filename, 'w') as scary_file_obj:
        json.dump(data, scary_file_obj, indent=4)

filename = 'scary_characters.json'
write_scary_data(filename, scary_output_data)
```





# Utility functions

return dictionary, return string

```
def get scary film character(film):
    """Return dictionary object."
    return film['scary_character']
def get scary film character name(film):
    """Return scary character name.""
    character = get scary_film_character(film)
    return character['name']
```





# Utility functions

check films list for film

```
def has_film_credit(films, title):
    """Check if film is in the films list."""
    if films:
        for film in films:
        if film['title'] == title:
        return True
```

return False





## Utility functions

return characters

### add film credits to scary character

```
def add_film_credits_to_scary_characters(characters, films):
    """Add scary character film credits."""
    for character in characters:
        character.setdefault('films', []) # add property if missing
        for film in films:

        if character['name'] == film['scary_character']['name'] \
            and not has_film_credit(character['films'], film['title']):

        character['films'].append({
            'title': film['title'],
            'year_released': film['year_released'],
            'budget': film['budget'],
            'box_office': film['box_office'],
        })
```





# start exercise





### When your code misbehaves debug flowchart

#### Attribute Error

You are calling a method on the wrong type of object

#### SyntaxError

You've forgotten the quotes around a string

You have forgotten to put a colon at the end of a def/if/for line

You have different number of open and close brackets in a statement

#### TypeError

You're trying to use an operator on the wrong type of objects

An object which you expect to have a value is actually None

You've used non-integer numbers in a list slice

You've called a method/ function with the wrong number or type of arguments

#### Indentation Error

You've used a mixture of tabs and spaces You haven't indented all

lines in a block equally

## My code isn't working:-(

Start here...

Do you get an

error when you

run the code?

Does the code

use loops or if

statements?

Two numbers which should

be equal are not

You are comparing a number

with a string representation

of a number (e.g. if 3 == "3")

A complex condition is not

giving the expected result

The order of precedence in the

condition is ambiguous - add

some parentheses

What type of error do you get?

#### NameError

You've misspelt a variable, function or method name

> You've forgotten to import a module

> You've forgotten to define a variable

Your code uses a variable outside the scope where it's defined

Your code calls a function before it's defined

You're trying to print a single word and have forgotten the quotes

#### **IOError**

You're trying to open a file that doesn't exist

#### KeyError

You're trying to look up a key that doesn't exist in a dict

http://pythonforbiologists.com

#### A variable that should contain a value does not

You are storing the return value of a function which You are printing an object changes the variable itself (e.g. sort)

#### A number which should be a fraction is coming out as zero in Python 2

You are dividing integers rather than floats. Convert the numbers to floats or from \_\_future\_\_ import division

#### I'm trying to print a value but getting a weirdlooking string

(e.g. a FileObject) when you want the result of calling a method on the object

#### A regular expression is not matching when I expect it to

You have forgotten to use raw strings or escape backslash characters

#### I am reading a file but getting no input

You have already read the contents of the file earlier in the code, so the cursor is at the end.

neithei

loops

#### A list which should have a value for every iteration only has a single value

You have defined the list inside the loop: move it outside

#### A loop which uses the range function misses out the last value

The range function is exclusive at the finish: increase it by one.

#### I am trying to loop over a collection of strings, but am getting individual characters

You are iterating over a string by mistake

I am trying to write multiple lines to a file but only getting a single one You have opened the file inside the loop: move it

also check.





### Weeks 8-15

### weeks I-7 topics +

- data types
  - dictionaries
  - tuples
- modules
  - CSV
  - json (encode/decode)
  - pathlib
  - requests
- functions
  - lambdas (anonymous functions)
- lists
  - list comprehensions
- classes

- local dev environment
  - Python install
  - source code editor/IDE
  - command line
- debugging
- file types
  - \*.csv
  - \*.json
- data structures
  - structured data
  - semi-structured data
- RESTful APIs
  - HTTP request/response
  - JSON

final individual project assignment





### Slide deck revisions

errata: corrections and other changes

Slide no(s). Fix ver. Description
v1p1



