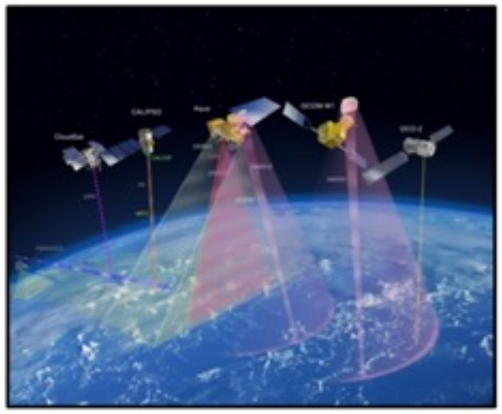
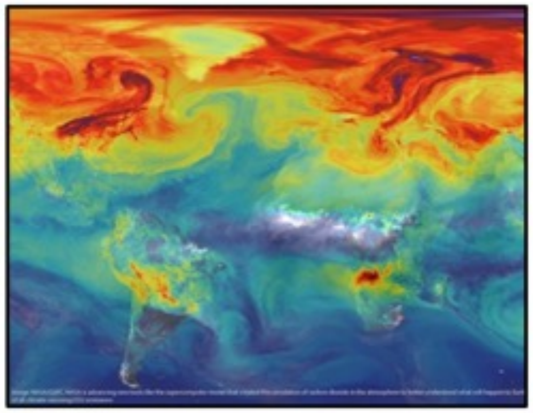



Geo Data Science with Python (GEOS-5984/4984)

Prof. Susanna Werth

Topic: Python Tuples, Sets, Files

Today's music is from: Becca

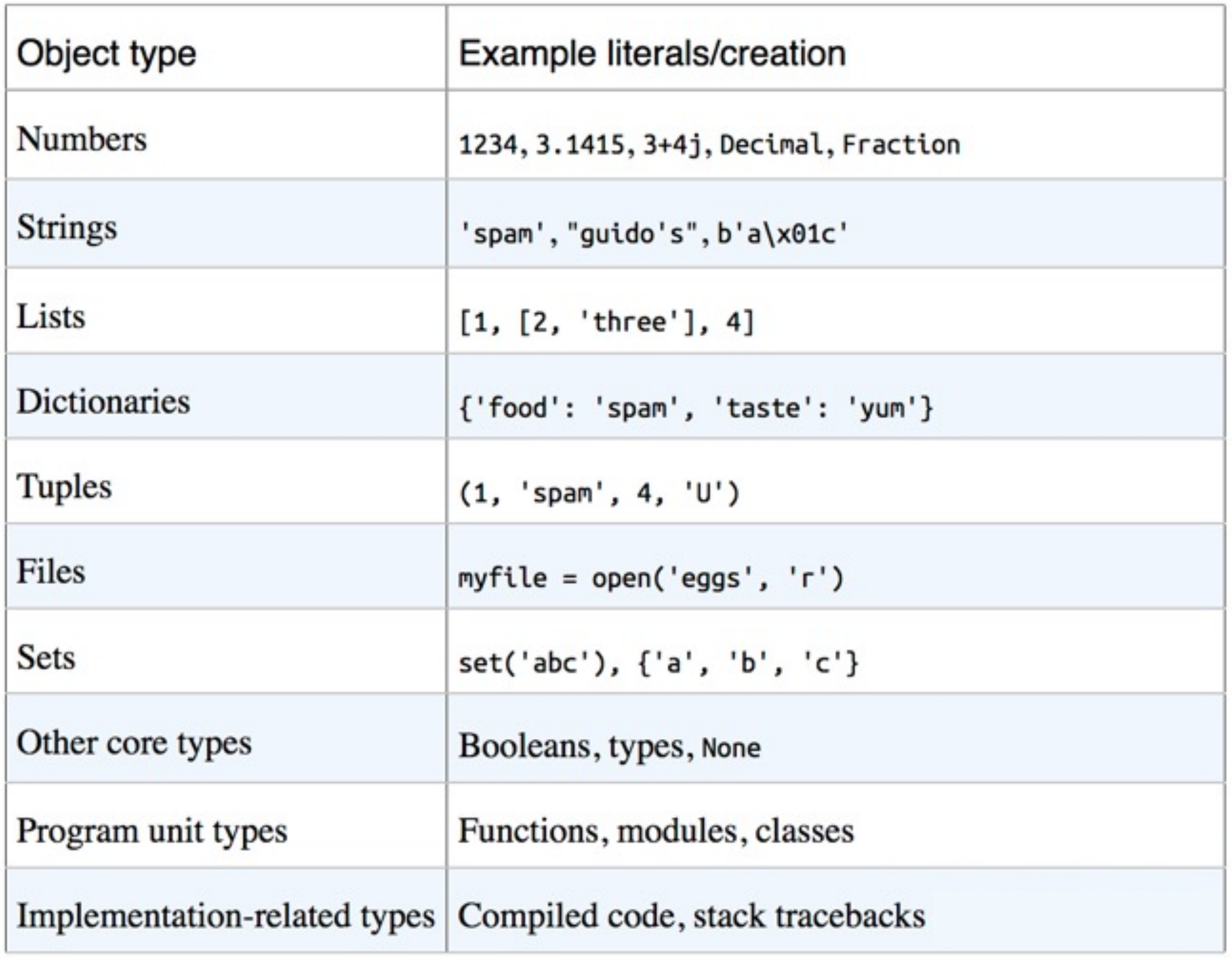
Please keep sending me your song suggestions through Canvas!

Notes/Reminders

- More music welcome!
- E03 - Task C.3 is updated, please update your repository

Printout: Python Built-in Objects

Table 4-1. Built-in objects preview

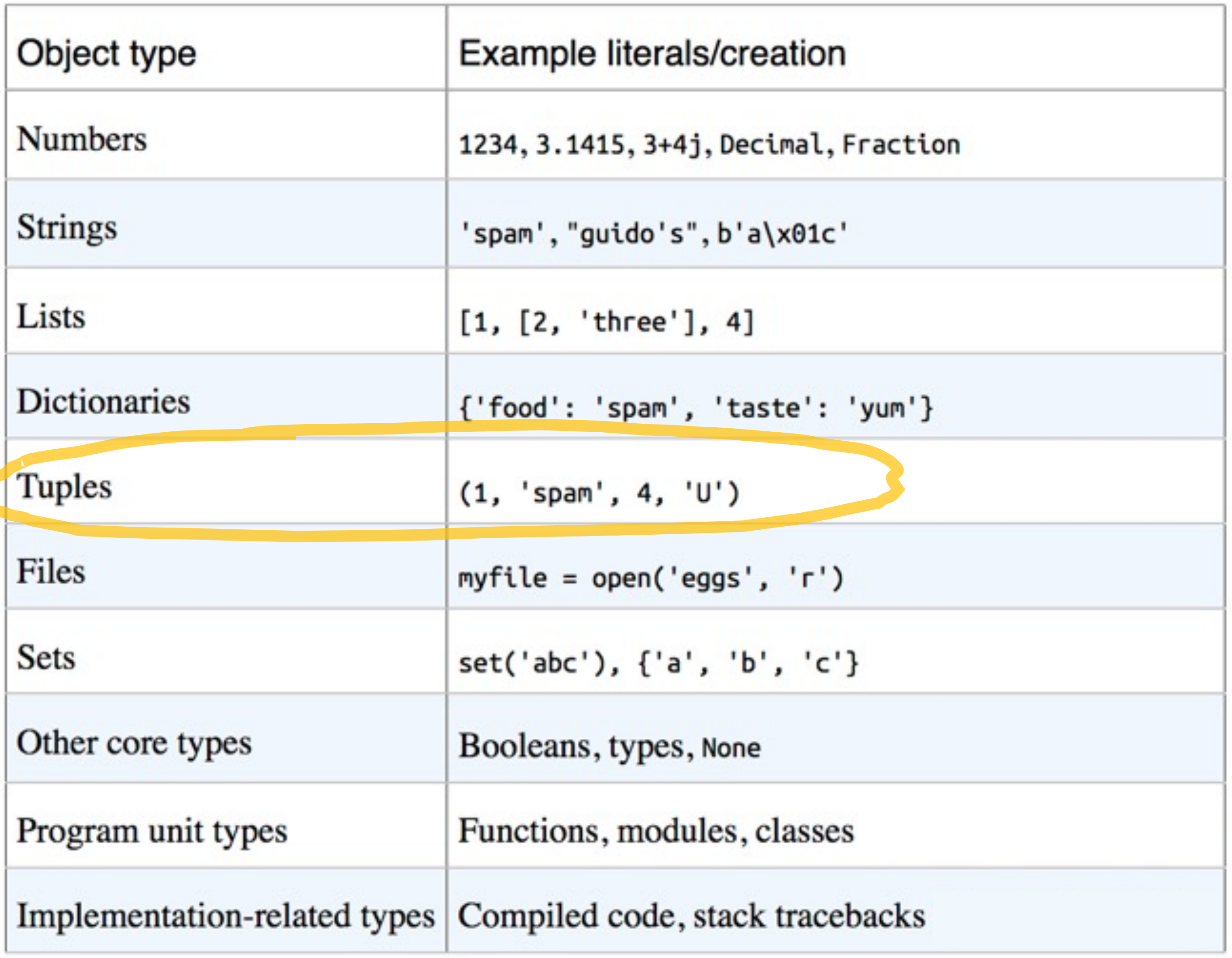


Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `3+4j`, `Decimal`, `Fraction`
Strings	`'spam'`, `"guido's"`, `b'a\x01c'`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1, 'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Sets	`set('abc')`, `{'a', 'b', 'c'}`
Other core types	Booleans, types, `None`
Program unit types	Functions, modules, classes
Implementation-related types	Compiled code, stack tracebacks

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media, Inc.

Python **Tuples**

Table 4-1. Built-in objects preview



Object type	Example literals/creation
Numbers	`1234, 3.1415, 3+4j, Decimal, Fraction`
Strings	`'spam', "guido's", b'a\x01c'`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1, 'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Sets	`set('abc'), {'a', 'b', 'c'}`
Other core types	Booleans, types, `None`
Program unit types	Functions, modules, classes
Implementation-related types	Compiled code, stack tracebacks

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media, Inc.

Tuples

- Sequence of values of any type
  - Ordered collection of *arbitrary* objects
  - **Immutable**, fixed-length
- Sequence operations and most list operations work on tuples:
  - List of values is indexed by integers
  - index, slice, concatenate, repetition, etc.
  - List comprehensions are applicable
  - Nesting

Tutorial

Tuple Examples

- Literal for assignment
- Sequence Operations
- Comparison to lists
- List Comprehension
  - Write a list comprehension for a tuple of numbers, returning the double of each

    Example: a_tuple = (2,3,4,5,6)
  - What object type does the list comprehension return and why?
- Range and Stride (also for other sequences)

Some Built-in Functions & Methods

Built-in Functions & Methods	Description
len()	Gives the total length of the tuple.
min()	Returns item from the tuple with min value.
max()	Returns item from the tuple with max value.
sum()	Add items.
tuple(seq)	Converts a list into tuple.
.index()	Returns the position at the first occurrence.
.count()	Return the number of times a value appears.

Tuples

- Why using tuples ?
    - Faster processed by Interpreter, compared to lists
        - While most sequence operations working on strings and lists also work on tuples
    - Immutability: similar role as “constant” declaration
    - Most commonly used as dictionary keys , which must be immutable
        - In contrast to lists
        - Helpful for larger datasets: can use entire datasets as keys

Tutorial

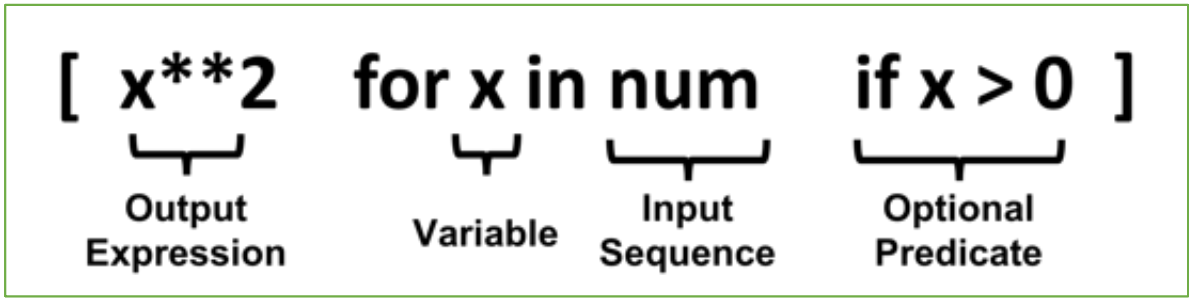
Work with Tuples

1. Create a tuple from 0 to 10 using the function range().
2. Write a list comprehension (including optional predicate) that returns only the even numbers from 0 to 10.

Note:

- *The list comprehension should work for any other range.*
- *Zero is an even number*
- *Tip: Operator % returns remainder of a division of two numbers.*

List Comprehension:



```
[ x**2 for x in num if x > 0 ]
```

Summary of Type Syntax

Container	Denotation	Feature	Examples
List	Delimited by ([ ])	mutable	[‘a’, ‘b’, ‘c’, ‘c’]
Dictionary	{key: value, key: value , …}	mutable	{‘Alice’: ‘7039931234’: ‘Beth’: ‘7033801235’}
Tuple	Denoted by parenthesis (( ))	immutable	(‘a’, ‘b’, ‘c’, ‘c’)
Set	set()	mutable	set([’a’, ‘b’, ‘c’])

Don’t know the data type of a variable? Type *type*!

```
>>> type(number)
<type ‘int’>
```

General Type Categories

Lutz (2013)

- **Numbers**
  - ✓**integer, floating-point, …**
  - Supports addition, multiplication, etc.
- **Sequences**
  - ✓**strings, lists, tuples**
  - Support indexing, slicing, concatenation, etc.
- **Mappings**
  - ✓**dictionaries**
  - Support indexing by key, etc.

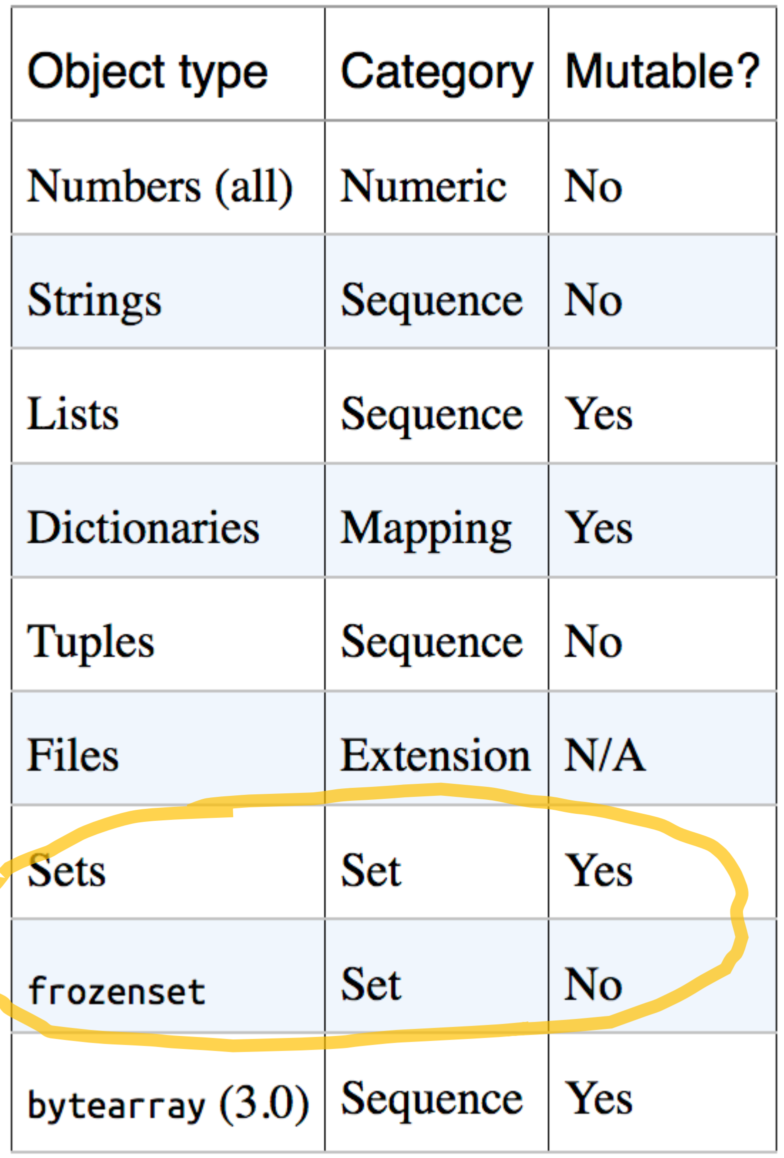
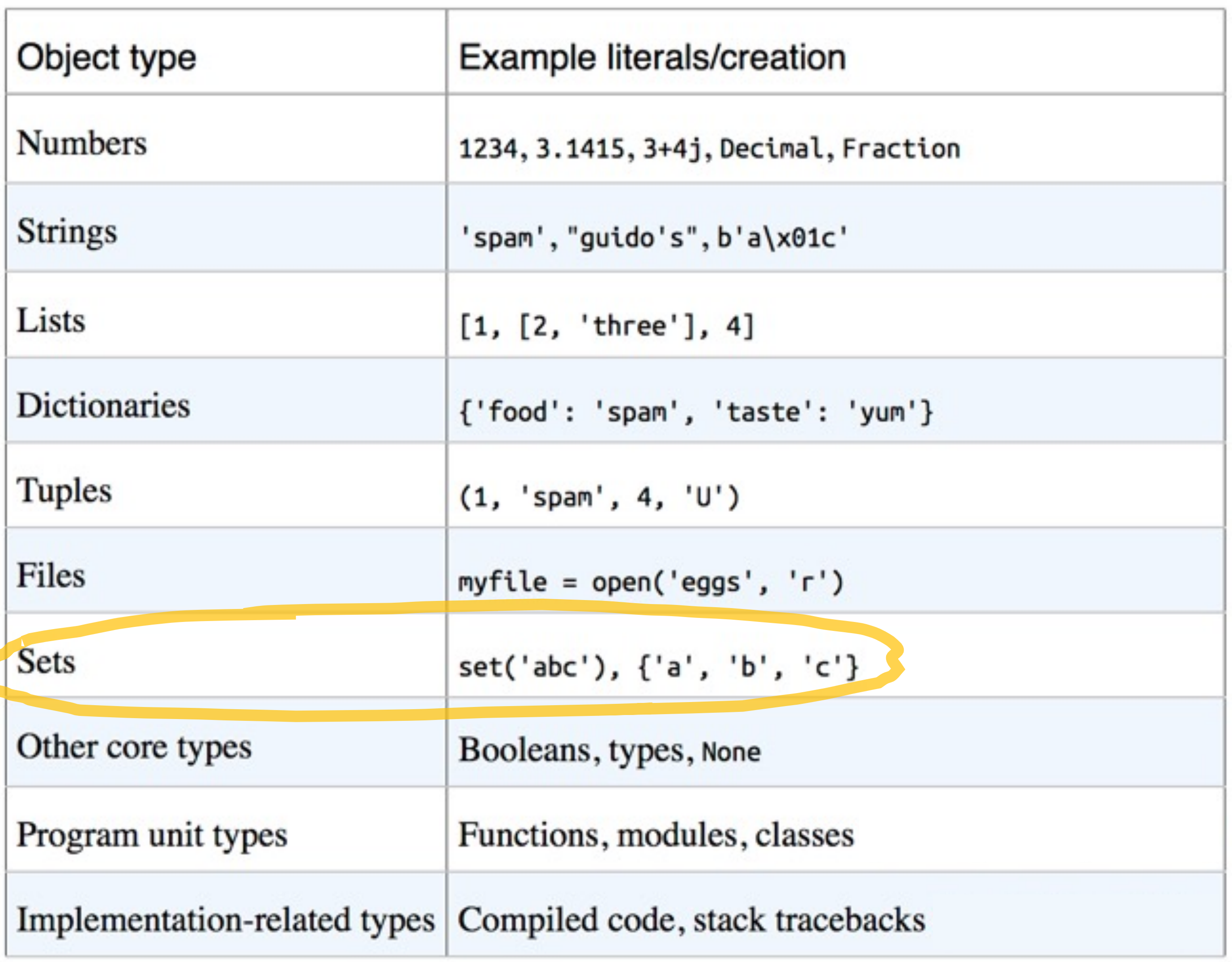


Table 9-3. Object classifications

Object type	Category	Mutable?
Numbers (all)	Numeric	No
Strings	Sequence	No
Lists	Sequence	Yes
Dictionaries	Mapping	Yes
Tuples	Sequence	No
Files	Extension	N/A
Sets	Set	Yes
`frozenset`	Set	No
`bytearray` (3.0)	Sequence	Yes

Python Sets

Table 4-1. Built-in objects preview

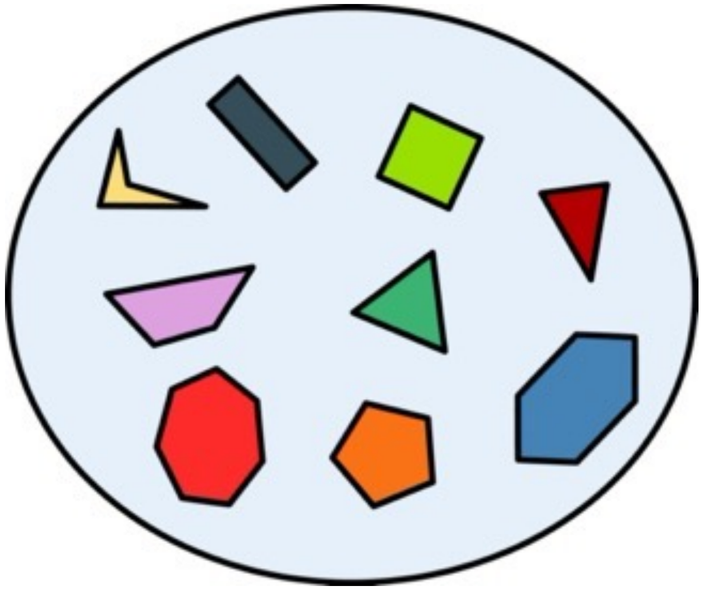


Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `3+4j`, `Decimal`, `Fraction`
Strings	`'spam'`, `"guido's"`, `b'a\x01c'`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1, 'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Sets	`set('abc')`, `{'a', 'b', 'c'}`
Other core types	Booleans, types, `None`
Program unit types	Functions, modules, classes
Implementation-related types	Compiled code, stack tracebacks

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media, Inc.

Mathematical Set Theory

- A *set* is an “unordered collection of unique and immutable objects that supports operations corresponding to **mathematical set theory**.”
- **set theory** branch of mathematical logic that studies sets
- **set** is a collection of distinct objects, considered as an object in its own right
- one of the most fundamental concepts in mathematics



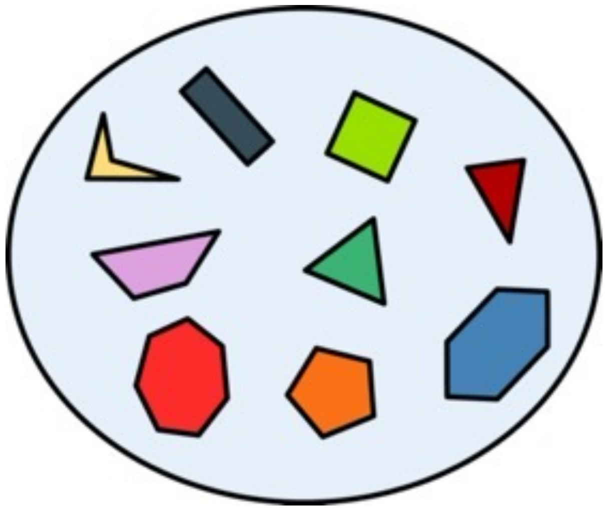
Tutorial

Set Examples

- Literal for assignment
- Create sets from lists

Sets in Python

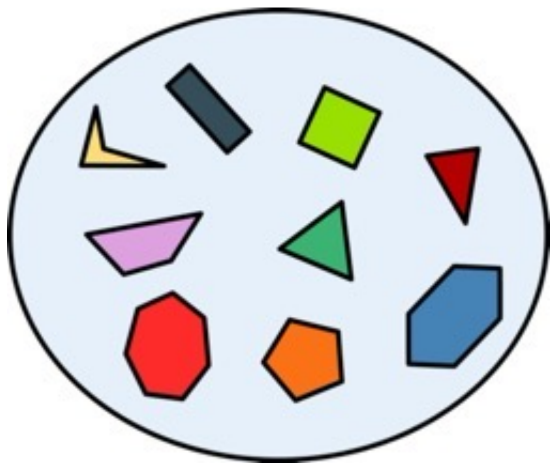
- Collection of unique **immutable objects**
- Unordered, no duplicates, not maps or sequences
- Sets themselves **mutable**
  - can embed tuples but cannot embed lists, dictionaries



List	Set
0	1
1	0
2	4
3	2
0	3
4	

List	Set
dog	cat
cat	dog
mouse	mouse
cat	duck
duck	
mouse	

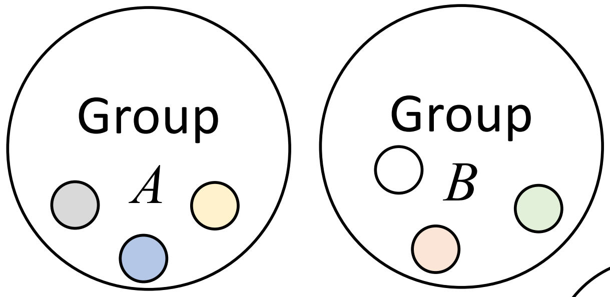
Purpose of Sets

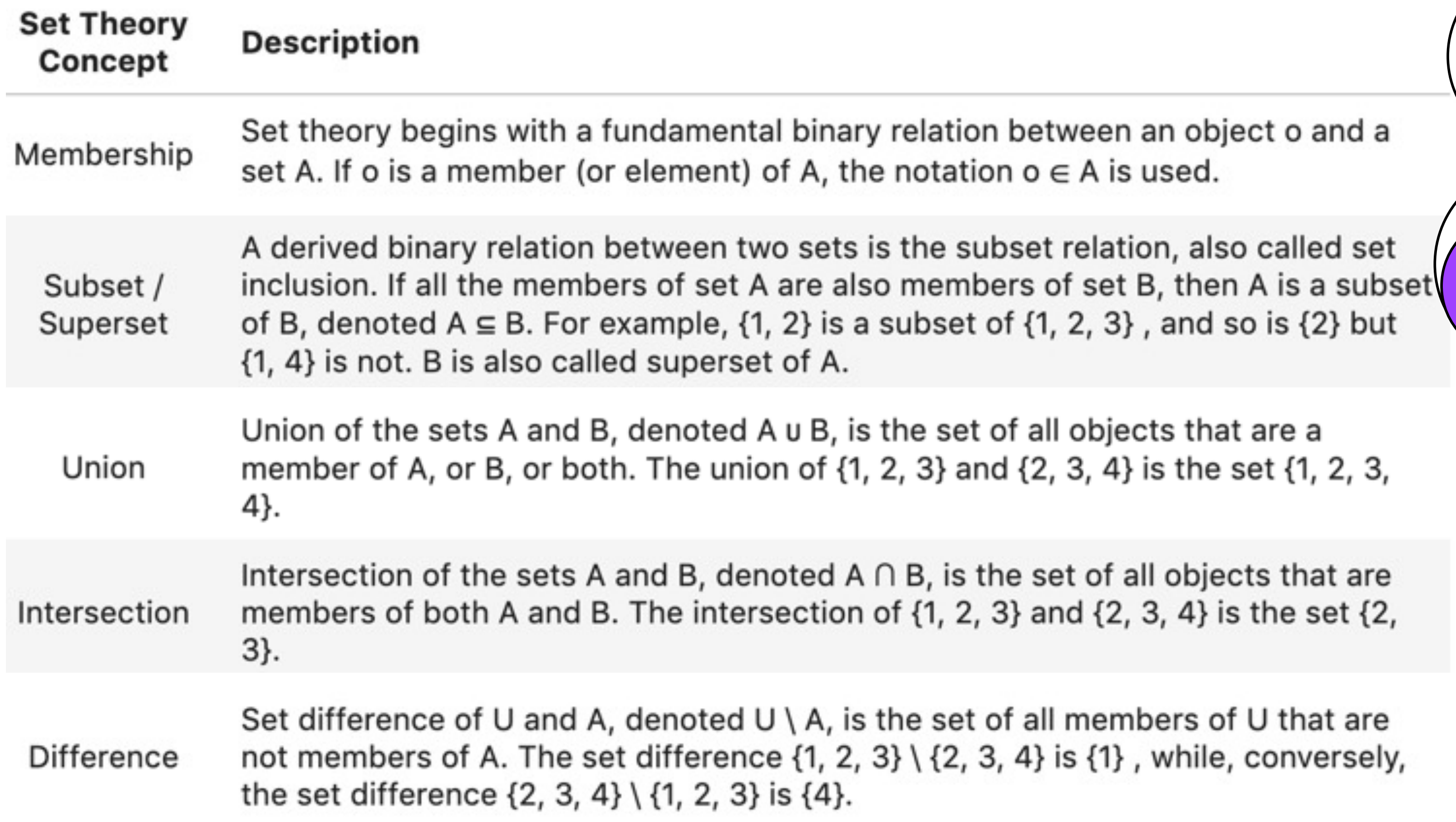


- Set theory studies
- Constructing and manipulating unsorted collections of unique elements
- Analyzing and comparing complex data structures
- Geo examples: getting spatial objects with two conditions matched
    - *Combining two regions' animal species to get collection*
    - *Get coordinates of police stations in a county*
    - *Finding French style restaurants near parks*

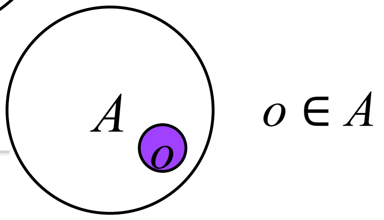
Set Operation Concepts

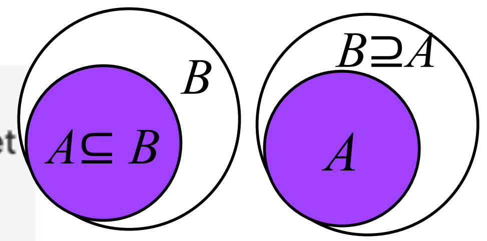
Groups of unique items

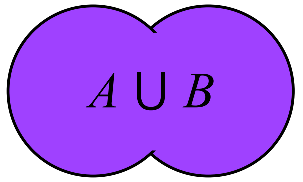


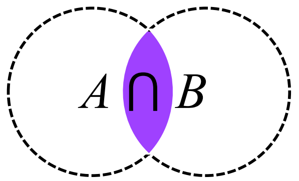


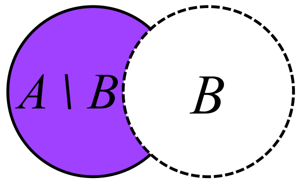
Set Theory Concept	Description
Membership	Set theory begins with a fundamental binary relation between an object o and a set A. If o is a member (or element) of A, the notation o ∈ A is used.
Subset / Superset	A derived binary relation between two sets is the subset relation, also called set inclusion. If all the members of set A are also members of set B, then A is a subset of B, denoted A ⊆ B. For example, {1, 2} is a subset of {1, 2, 3} , and so is {2} but {1, 4} is not. B is also called superset of A.
Union	Union of the sets A and B, denoted A ∪ B, is the set of all objects that are a member of A, or B, or both. The union of {1, 2, 3} and {2, 3, 4} is the set {1, 2, 3, 4}.
Intersection	Intersection of the sets A and B, denoted A ∩ B, is the set of all objects that are members of both A and B. The intersection of {1, 2, 3} and {2, 3, 4} is the set {2, 3}.
Difference	Set difference of U and A, denoted U \ A, is the set of all members of U that are not members of A. The set difference {1, 2, 3} \ {2, 3, 4} is {1} , while, conversely, the set difference {2, 3, 4} \ {1, 2, 3} is {4}.











Tutorial

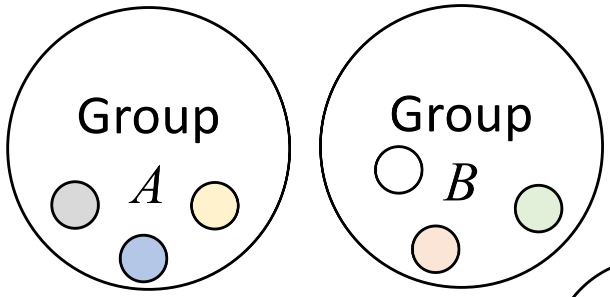
Set Examples

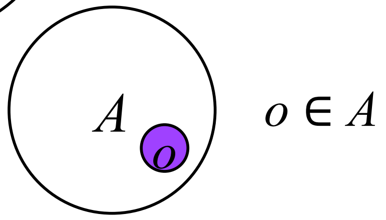
- Set Operations

Set Operation in Python

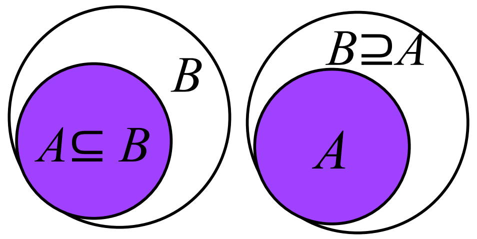
Groups of unique items

```
>>> A = set('abcde’)
>>> B = set('bdxyz')
```

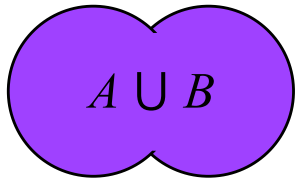




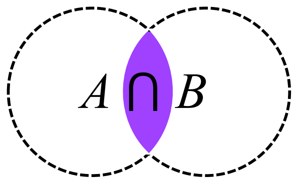
```
>>> ‘o' in A                    # Membership
```



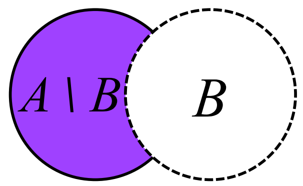
```
>>> A < B, B > A                # Subset, Superset
```



```
>>> A | B                       # Union
```



```
>>> A & B                       # Intersection
```



```
>>> A - B                       # Difference
```

Sets: Method Operations

Operation	Equivalent	Result
len(s)		number of elements in set s (cardinality)
s.issubset(t)	**s <= t**	test whether every element in s is in t
s.issuperset(t)	**s >= t**	test whether every element in t is in s
s.union(t)	**s | t**	new set with elements from both s and t
s.intersection(t)	**s & t**	new set with elements common to s and t
s.difference(t)	**s - t**	new set with elements in s but not in t
s.symmetric_difference(t)	**s ^ t**	new set with elements in either s or t but not both
s.copy()		new set with a shallow copy of s
s.add(x)		add element x to set s
s.remove(x)		remove x from set s
s.pop(x)		remove and return random element from s
s.clear()		remove all elements from set s

List comprehensions

```
>>> [e for e in x]      # list comprehensions working on sets
['a', 'c', 'b', 'e', 'd’]
```

Tutorial

Set Exercise

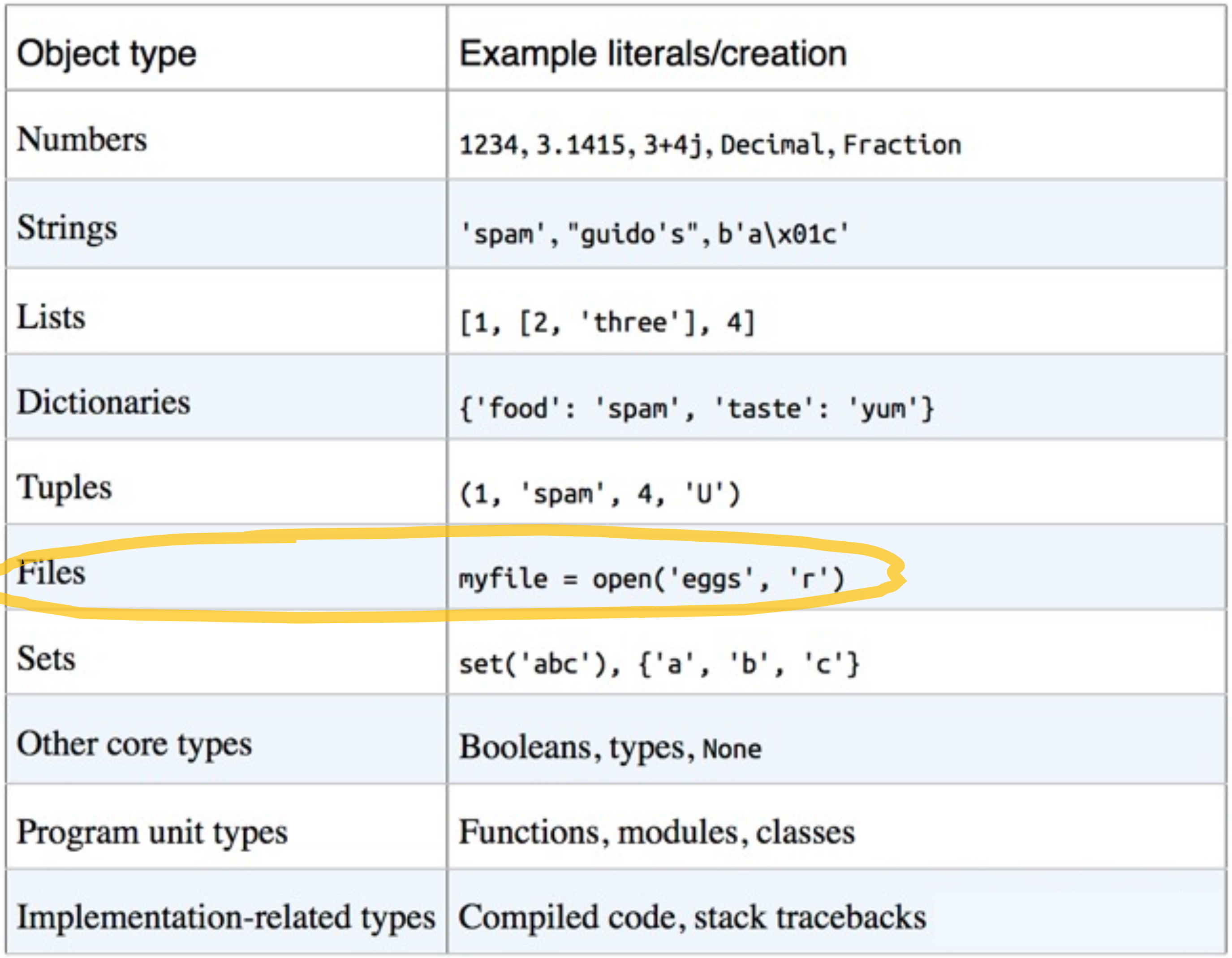
- Load the Python code snippet
  ~/geosf22_material/L06/SetSnippet.py

  with the Notebook magic command
  %load ~/geosf22_material/L06/SetSnippet.py

- Write Set operations to solve the questions.

Python **File Objects**

Table 4-1. Built-in objects preview



Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `3+4j`, `Decimal`, `Fraction`
Strings	`'spam'`, `"guido's"`, `b'a\x01c'`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1, 'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Sets	`set('abc')`, `{'a', 'b', 'c'}`
Other core types	Booleans, types, `None`
Program unit types	Functions, modules, classes
Implementation-related types	Compiled code, stack tracebacks

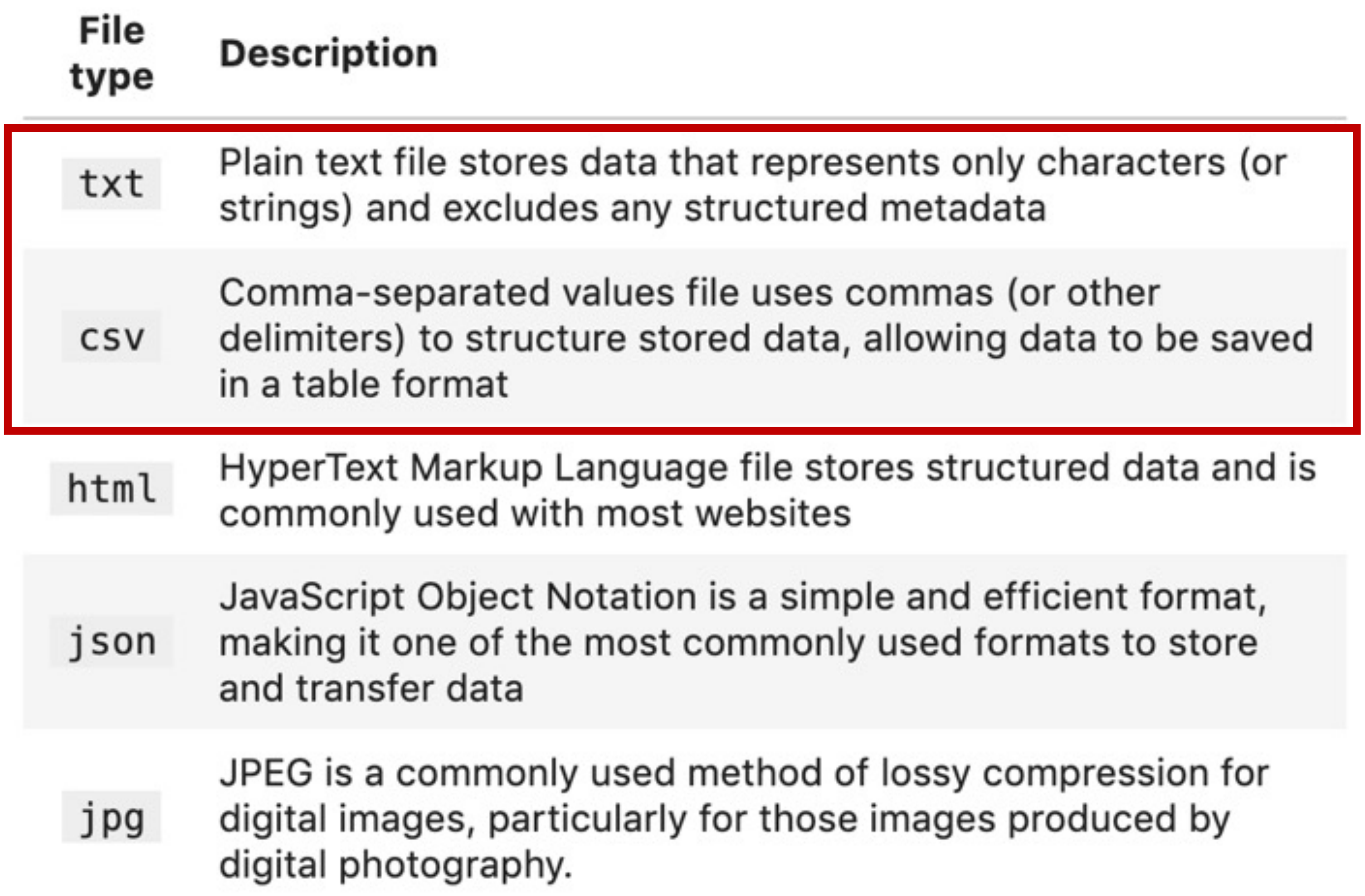
Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media, Inc.

Files Objects

- Main interface to external files on your computer
- Core object type
- No specific literal syntax for creating or reading
- **Built-in functions** for handling files

- Major file types:
    - text files: strings in unicode endocing
    - binary files: bytes strings type

Typical File Types to Read



File type	Description
`txt`	Plain text file stores data that represents only characters (or strings) and excludes any structured metadata
`csv`	Comma-separated values file uses commas (or other delimiters) to structure stored data, allowing data to be saved in a table format
`html`	HyperText Markup Language file stores structured data and is commonly used with most websites
`json`	JavaScript Object Notation is a simple and efficient format, making it one of the most commonly used formats to store and transfer data
`jpg`	JPEG is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.

Files: Opening & Closing

- File objects are created using the open() function.

```
>>> f = open('test.txt', 'w')    # opens file for writing
```

code	file opening mode
r	reading only, DEFAULT
w	writing (existing file will be overwritten)
a	appending data to end of file
r+	reading and writing

- File objects have to be closed, once done.

```
>>> f.close()              # closes the file
```

Files: Reading

```
>>> f = open('test.txt', 'r')
>>> text = f.read()        # Reads entire file into string
>>> text = f.readline()    # Reads line by line (incl. \n)
>>> text = f.readlines()   # Reads all lines to list of strings (incl. \n)

>>> text = [line for line in open('test.txt', 'r')]
          # List comprehension to read files line by line (incl. \n)
```

Files: Writing

```
>>> f.write(‘hello\n’)            # Writes string into file

>>> f.writelines([‘hello\n’,’whatever\n’])
                                   # Writes list of strings to file
```

Tutorial

Writing Files

- Read the file days.txt as a text file.
- We want to rewrite the file content to a days2.txt, that contains a title at the top: “Days of the Week” followed by the original content.

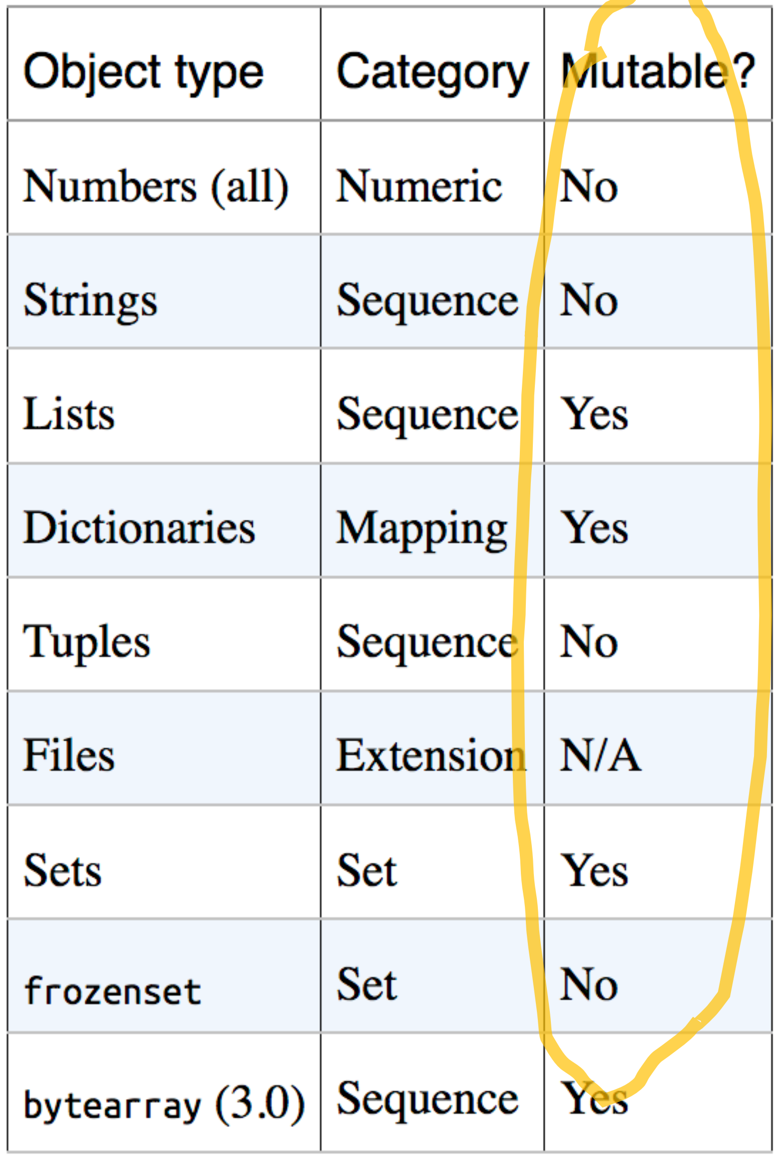
- User Input?

General Type Categories

Lutz (2013)

- **Numbers**
  - ✓**integer, floating-point, …**
  - Supports addition, multiplication, etc.
- **Sequences**
  - ✓**strings, lists, tuples**
  - Support indexing, slicing, concatenation, etc.
- **Mappings**
  - ✓**dictionaries**
  - Support indexing by key, etc.

Table 9-3. Object classifications



Object type	Category	Mutable?
Numbers (all)	Numeric	No
Strings	Sequence	No
Lists	Sequence	Yes
Dictionaries	Mapping	Yes
Tuples	Sequence	No
Files	Extension	N/A
Sets	Set	Yes
`frozenset`	Set	No
`bytearray` (3.0)	Sequence	Yes

Practice

- Optional: **Study Python Tuples**, **Sets** and **Files** via the following slides and reading material L06_reading_TuplesFilesSets.ipynb
- Find out how to read from user input