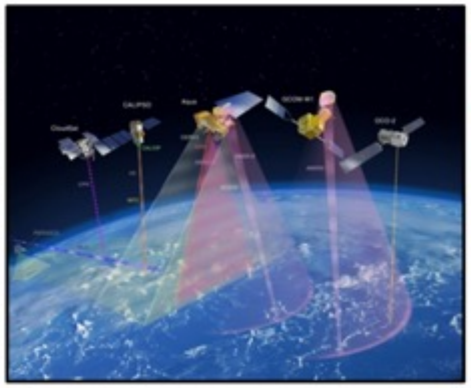
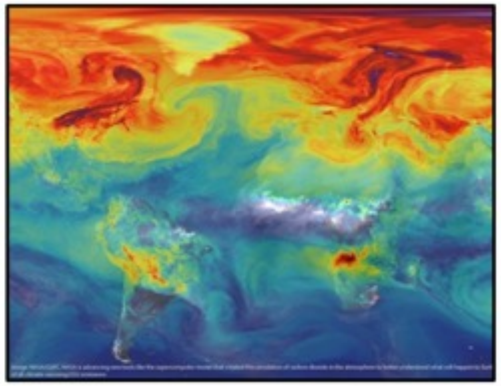



Geo Data Science with Python (GEOS-5984/4984)

Prof. Susanna Werth

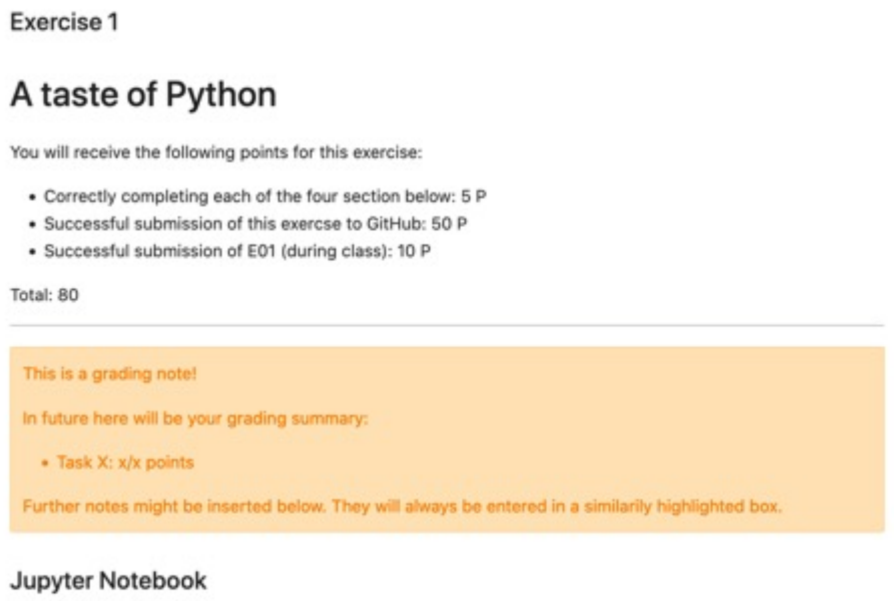
Topic: Python Lists And Dictionaries

Today's music is from: Carmen and Abdullah

Please keep sending me your song suggestions through Canvas!

Notes/Reminders

- More music welcome!
- Graded homework test:
  Your repository should now have a file E02_ATasteOfPython_copy_graded.ipynb, with an example grading note:



Exercise 1

A taste of Python

You will receive the following points for this exercise:

- Correctly completing each of the four section below: 5 P
- Successful submission of this exercse to GitHub: 50 P
- Successful submission of E01 (during class): 10 P

Total: 80

This is a grading note!

In future here will be your grading summary:

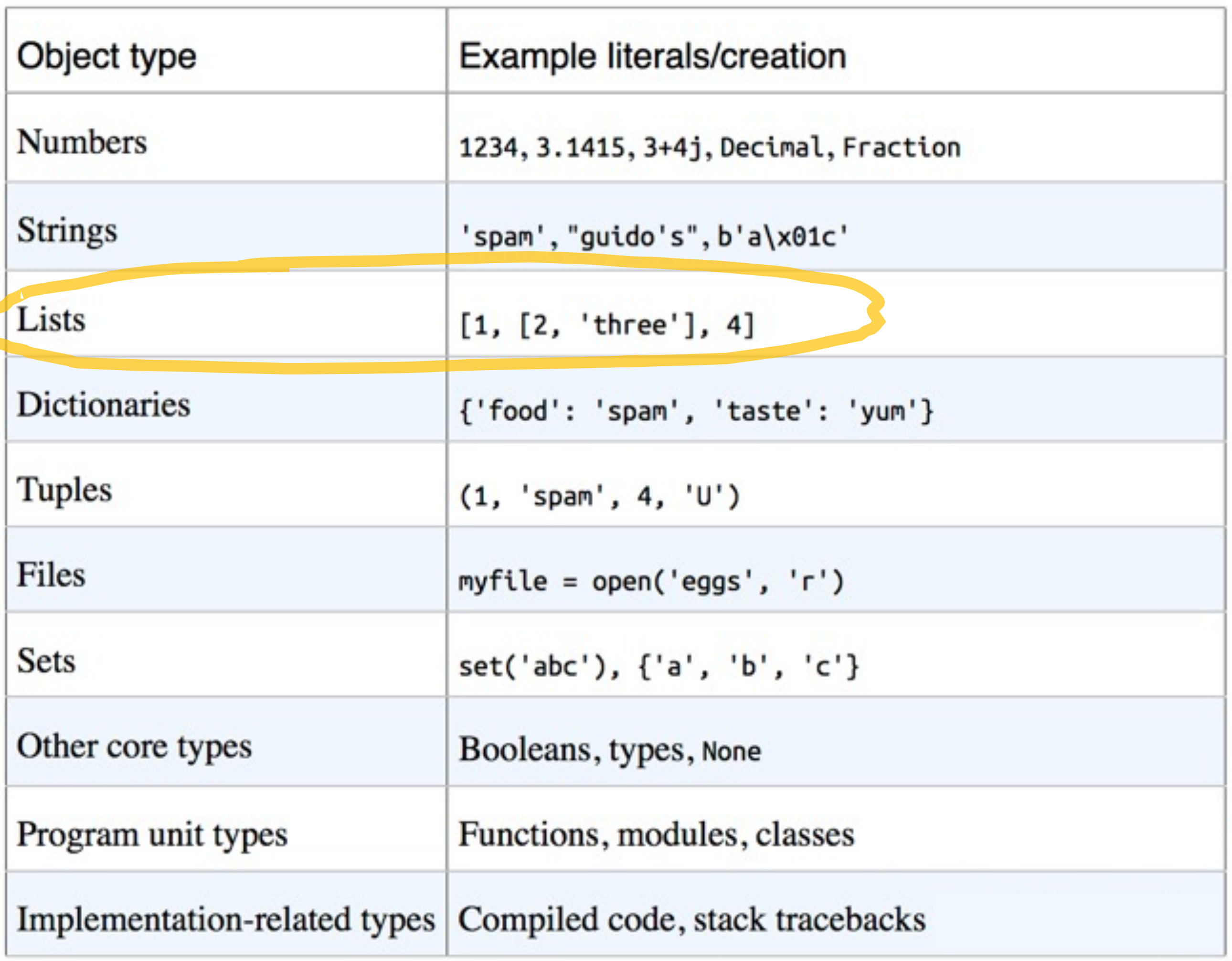
- Task X: x/x points

Further notes might be inserted below. They will always be entered in a similarily highlighted box.

Jupyter Notebook

Python Lists

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

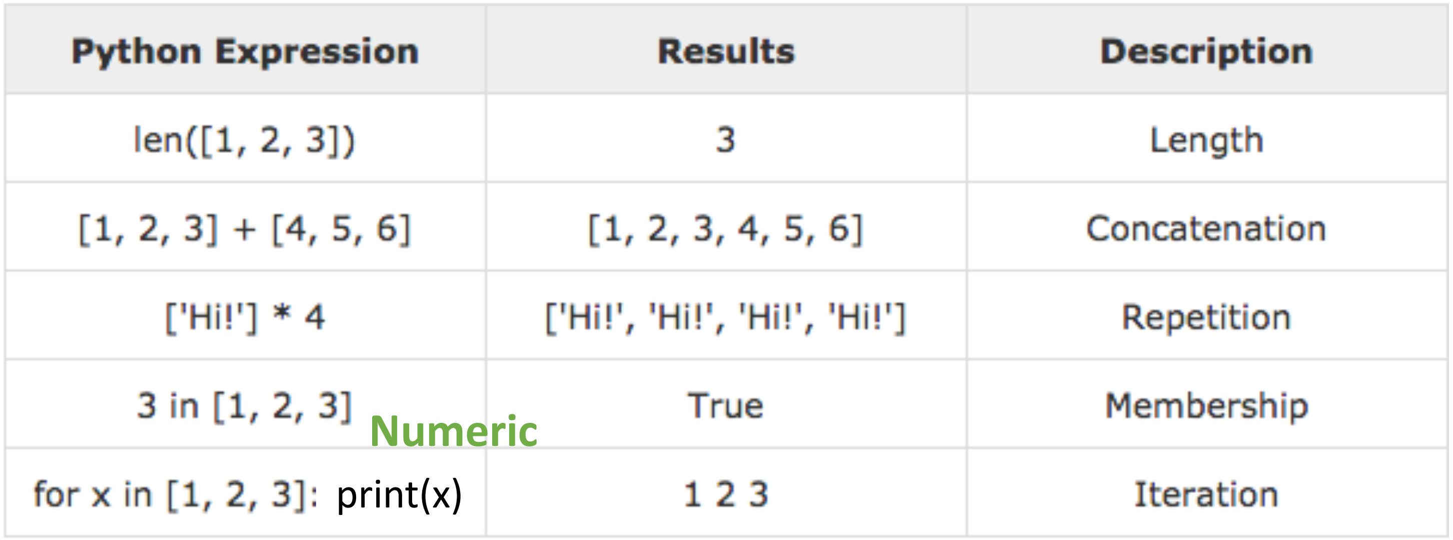
Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media, Inc.

Lists

- "Most general **sequence** provided by the language"
  - Ordered collection of *arbitrary* objects
  - Allows for all sequence operations (index, slice, concatenate, repetition, etc.)

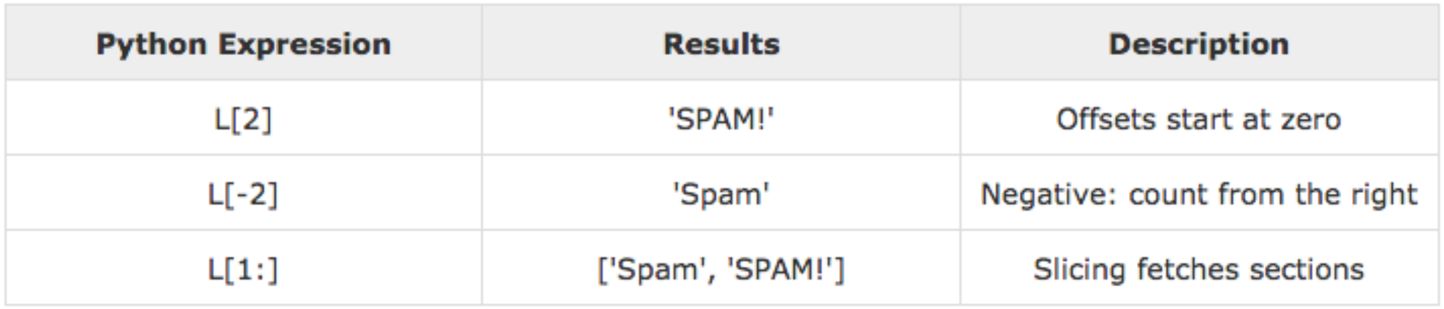
- **Mutable** (in contrast to string sequences)

Basic Sequence Operations



Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print(x)	1 2 3	Iteration

Numeric



Python Expression	Results	Description
L[2]	'SPAM!'	Offsets start at zero
L[-2]	'Spam'	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

https://www.tutorialspoint.com/python/python_lists.htm
More in e.g., Lutz (2013), Table 8-1

Tutorial

Examples

- Literal for assignment
- Sequence Operations
  - Indexing
  - Slicing
  - Concatenation
  - Multiplication
- Nested Lists

Mutability of Objects

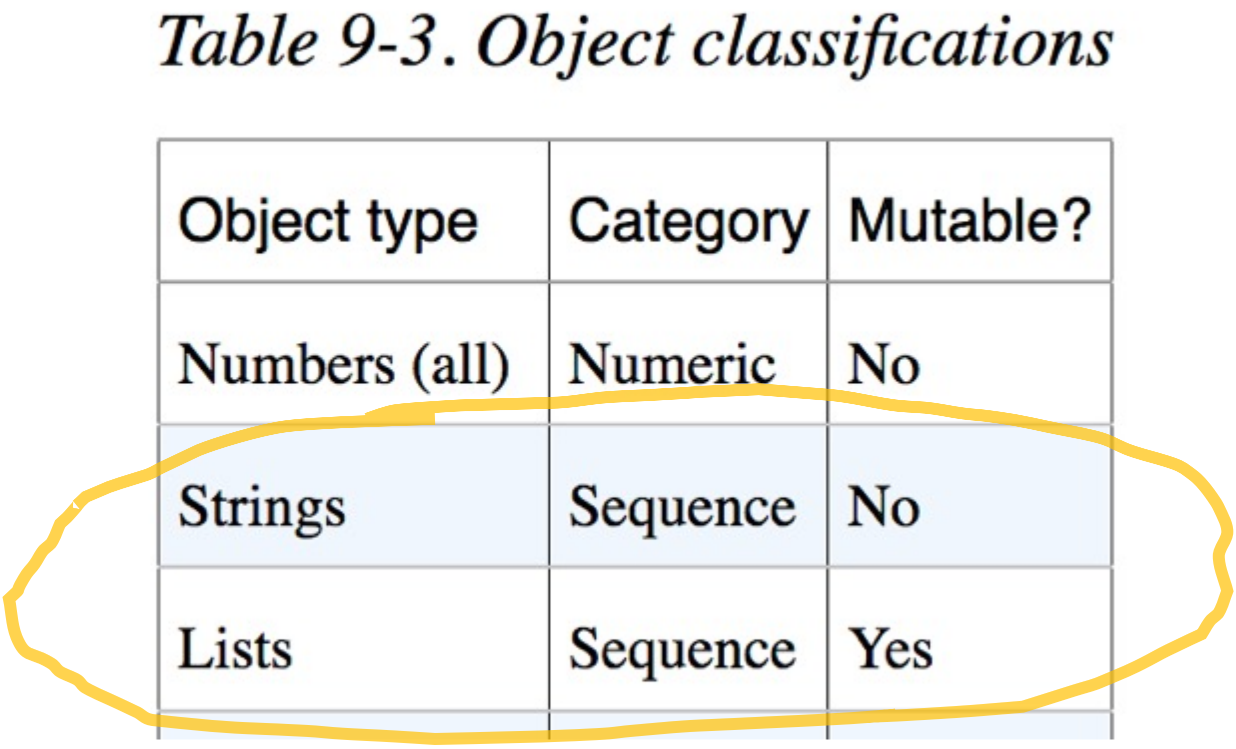


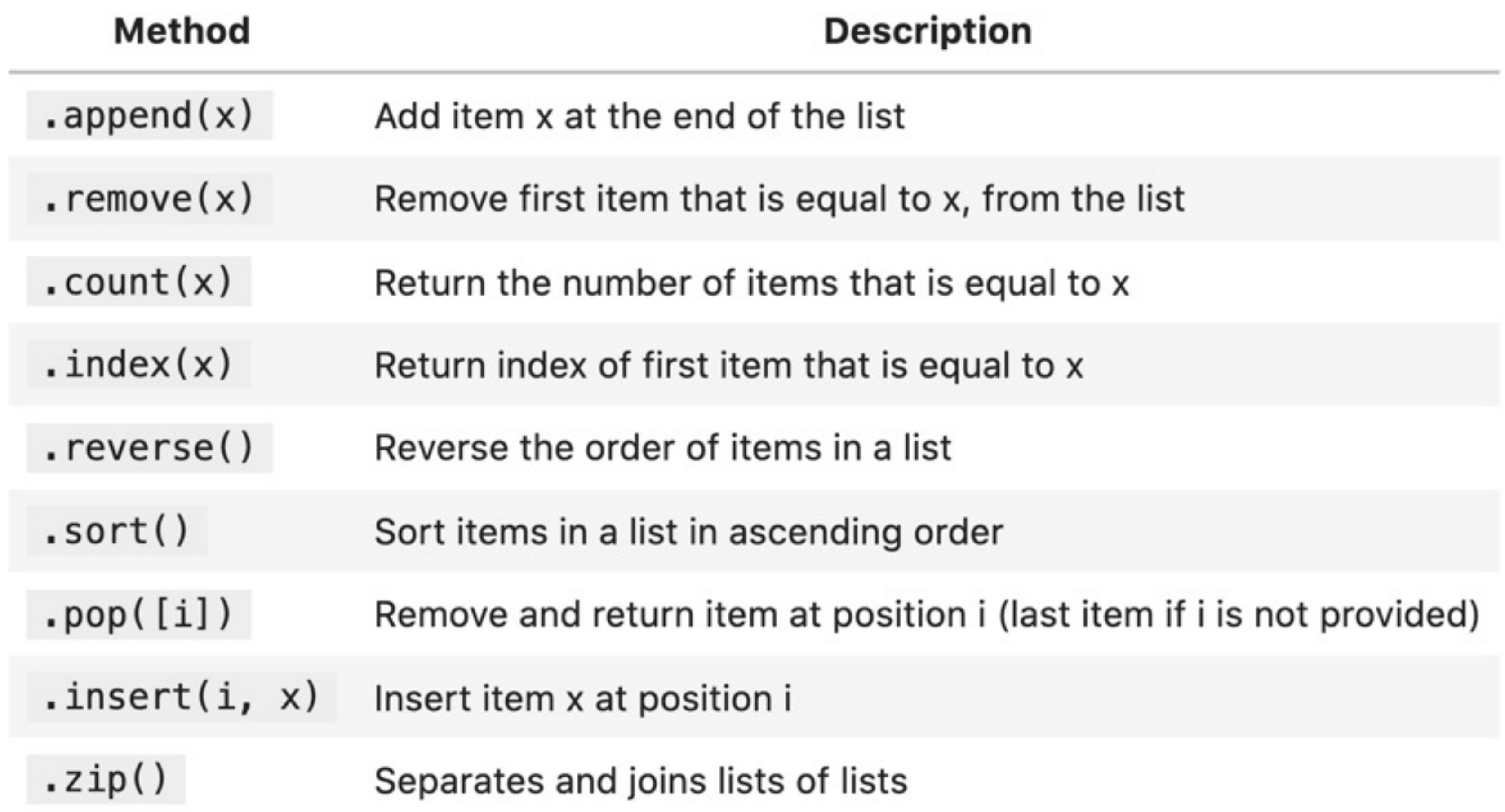
Table 9-3. Object classifications

Object type	Category	Mutable?
Numbers (all)	Numeric	No
Strings	Sequence	No
Lists	Sequence	Yes

Lutz (2013)

Built-in List Methods

Table 1: *Important List Methods*



Method	Description
`.append(x)`	Add item x at the end of the list
`.remove(x)`	Remove first item that is equal to x, from the list
`.count(x)`	Return the number of items that is equal to x
`.index(x)`	Return index of first item that is equal to x
`.reverse()`	Reverse the order of items in a list
`.sort()`	Sort items in a list in ascending order
`.pop([i])`	Remove and return item at position i (last item if i is not provided)
`.insert(i, x)`	Insert item x at position i
`.zip()`	Separates and joins lists of lists

More Examples: https://www.tutorialspoint.com/python/python_lists.htm

Python Assignment

- **creates object > creates variable > links them**
  1. *Variables (Names)* are entries in a system table, with spaces for links to objects;
  2. *Objects* are pieces of allocated memory, with enough space to represent the values for which they stand; and
  3. *References* are automatically followed *pointers* from variables to objects.

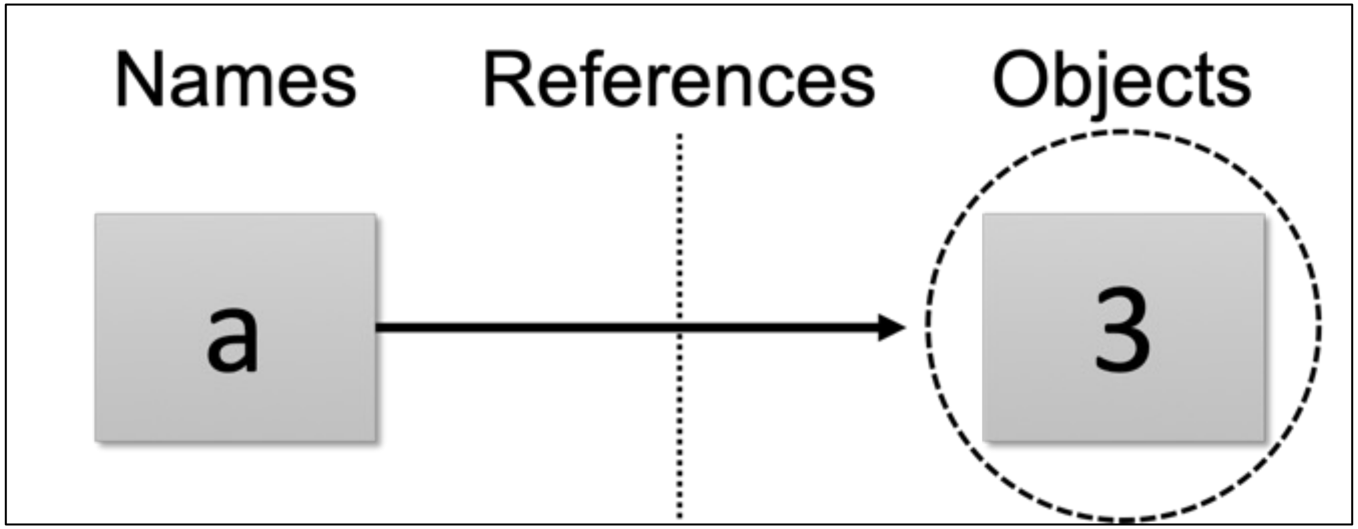


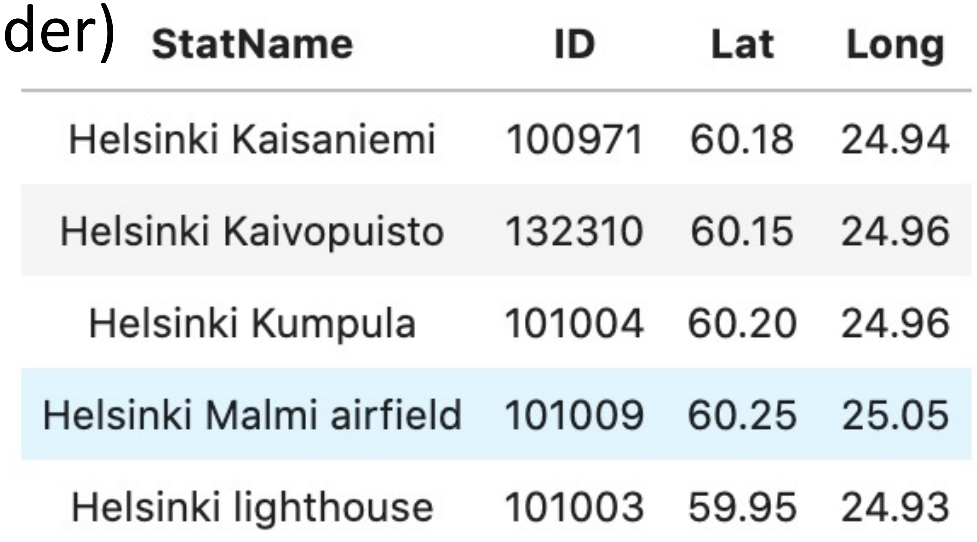
Figure 6-1. Names and objects after running the assignment a = 3. Variable a becomes a reference to the object 3. Internally, the variable is really a pointer to the object's memory space created by running the literal expression 3.

Lutz (2013)

Tutorial Save Data in Python *Lists*

The table provides information about weather stations in Finland (copy from L05_reading_Dict).

1. Store in a list of lists ***meteoStations*** (rows as inner list, ignore table header)



StatName	ID	Lat	Long
Helsinki Kaisaniemi	100971	60.18	24.94
Helsinki Kaivopuisto	132310	60.15	24.96
Helsinki Kumpula	101004	60.20	24.96
Helsinki Malmi airfield	101009	60.25	25.05
Helsinki lighthouse	101003	59.95	24.93

2. Use remove() and insert() method to remove the last item and insert it as second, append() any new field
3. What is the data type of your list elements?

Summary List Comprehensions

$$[\ \underbrace{\texttt{x**2}}_{\substack{\text{Output}\\ \text{Expression}}}\quad \texttt{for}\ \underbrace{\texttt{x}}_{\substack{\text{Variable}\\ \text{(local)}}}\ \texttt{in}\ \underbrace{\texttt{num}}_{\substack{\text{Input}\\ \text{Sequence}}}\quad \underbrace{\texttt{if x > 0}}_{\substack{\text{Optional}\\ \text{Predicate}}}\ ]$$

4 3 1 (2)

Reading order

Tutorial

Lists & Comprehensions

1. Create a list from 0 to 10 using the function range().
2. Write a list comprehension that returns only the even numbers from 0 to 10.
   Tip: Operator % returns remainder of a division of two numbers

3. Write a list comprehension that extracts only station ID's and names from your data set **meteoStations**.
4. Store the results in the variables **meteoIDs** and **meteoStatNames**.

Tutorial

Comprehensions on nested Lists

We have the following nested list (list of three lists):

m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Which is like a matrix:

$$m = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Which code will retrieve:

- 2nd row of the matrix (4,5,6)
- 2nd column of the matrix (2,5,8)

Python Dictionaries

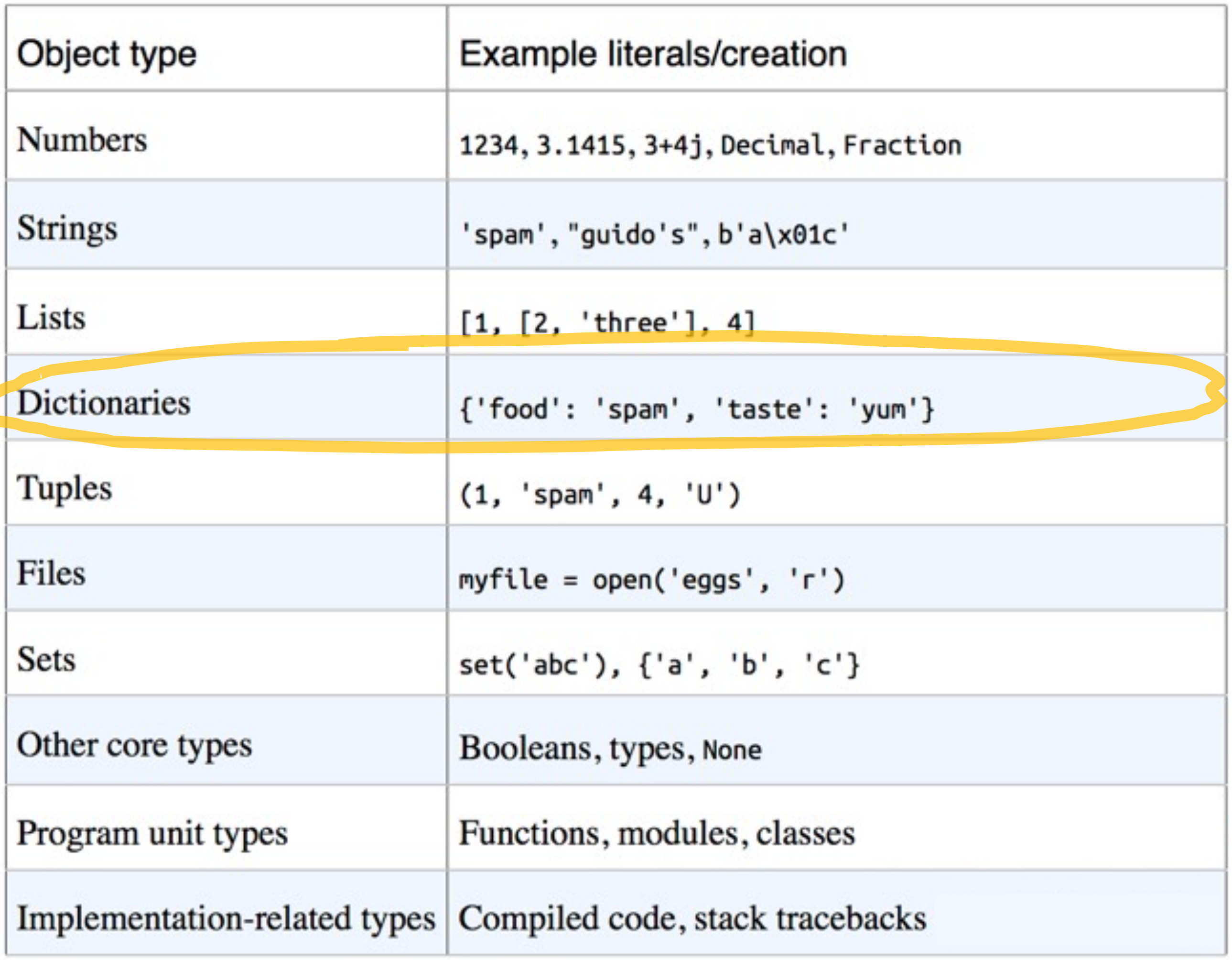


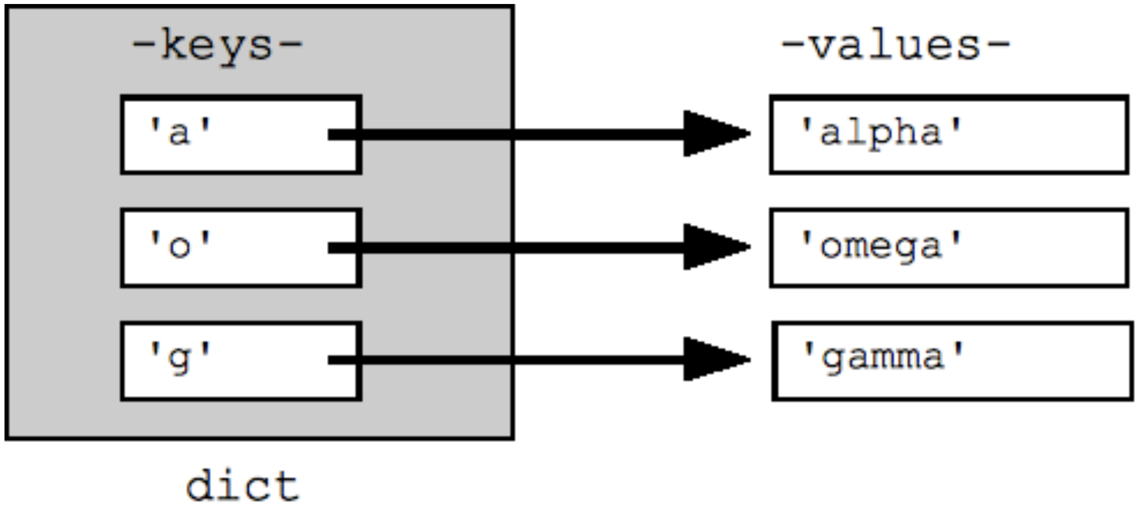
Table 4-1. Built-in objects preview

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 **Dictionaries**

- **Unordered** collection of *arbitrary* objects (no sequence operations) = ***Mutable*** *mapping*
- Are like lists, but more general: can have any type of index!
- *key* - *value* pairs: accessing content by **key**
    - Objects **map** keys to values (no fixed positions for items)



- **Type specific operations** (methods: pop, keys, values, items, get, update, …

Tutorial

Examples

- Literal for assignment of dictionary
- Accessing Dictionaries
- Dictionary Methods
  - Keys
  - Values
  - items
  - Get
  - *Please study both sections of the reading book on dictionaries L05_reading_Dictionaries.ipynb to learn about methods for modifying and deleting dictionary elements!*
- Nested Dictionaries

Dictionaries Keys

- Dictionaries Map keys to values.
- Values of Dictionaries are directly accessible by keys.
- **Keys have to be of immutable object type**, e.g. numbers or strings. Lists cannot be used as keys
- A list comprehension performed on a dictionary iterates over its keys by default. The keys will appear in an arbitrary order.

```
>>> [e for e in dictname]
['key1', 'key2',...]
```

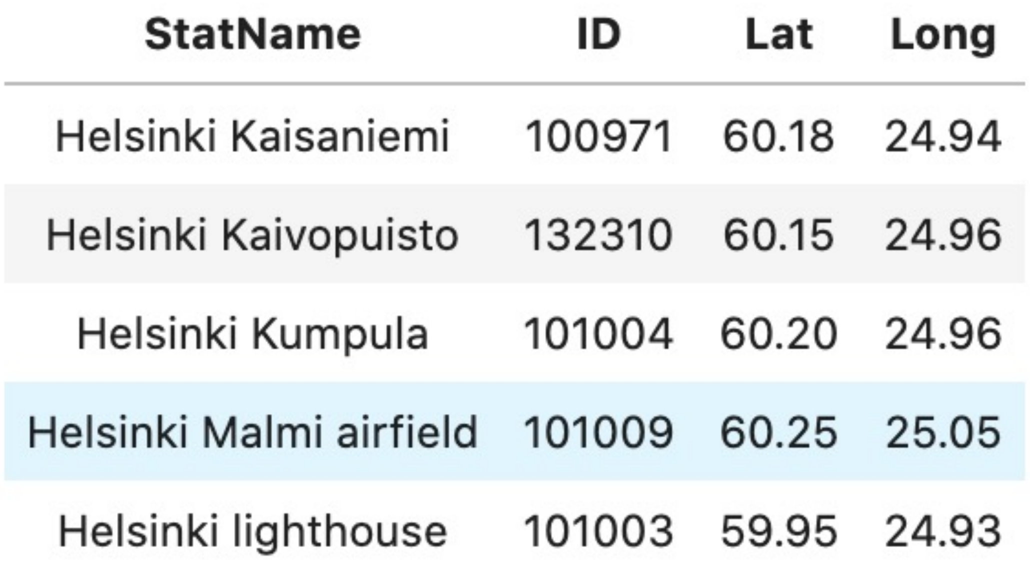
- Lists are mutable, they can't be used as keys, but they can be used as nested values (nesting dicts and list)

Tutorial

Save Data in Python *Dict*

The table provides information about weather stations in Finland (copy from L05_reading_Diction…).

1. Store in a dictionary ***meteoStatDict: dict of 5 dicts***



StatName	ID	Lat	Long
Helsinki Kaisaniemi	100971	60.18	24.94
Helsinki Kaivopuisto	132310	60.15	24.96
Helsinki Kumpula	101004	60.20	24.96
Helsinki Malmi airfield	101009	60.25	25.05
Helsinki lighthouse	101003	59.95	24.93

2. Write code to retrieve specific **row** from the dataset.
3. Write list comprehension to retrieve specific **column** from the datasets.

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.

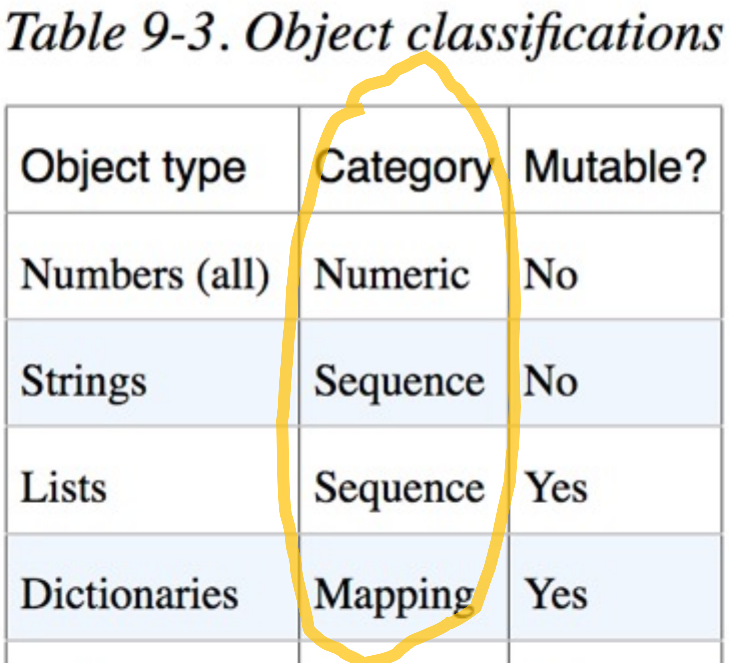
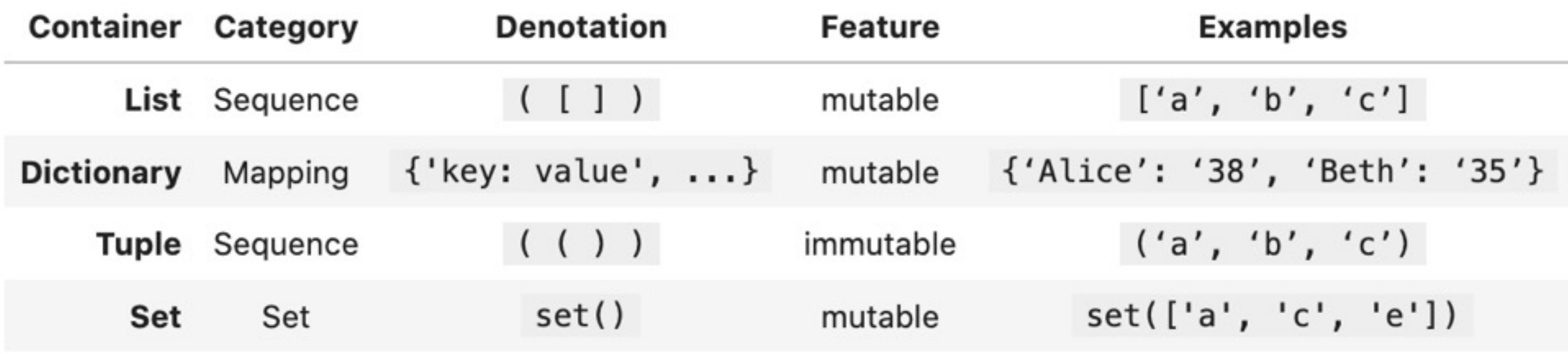


Table 9-3. Object classifications

Object type	Category	Mutable?
Numbers (all)	Numeric	No
Strings	Sequence	No
Lists	Sequence	Yes
Dictionaries	Mapping	Yes

Literals of Complex Data Containers



Container	Category	Denotation	Feature	Examples
List	Sequence	`( [ ] )`	mutable	`[‘a’, ‘b’, ‘c’]`
Dictionary	Mapping	`{'key: value', ...}`	mutable	`{‘Alice’: ‘38’, ‘Beth’: ‘35’}`
Tuple	Sequence	`( ( ) )`	immutable	`(‘a’, ‘b’, ‘c’)`
Set	Set	`set()`	mutable	`set(['a', 'c', 'e'])`

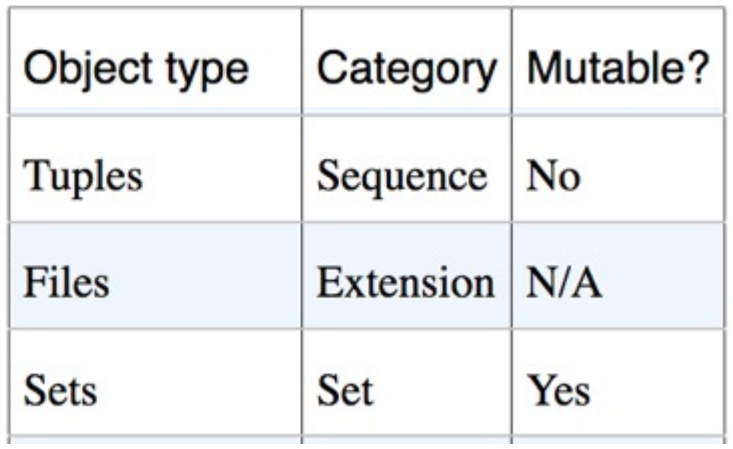


Table 9-3. Object classifications

Object type	Category	Mutable?
Tuples	Sequence	No
Files	Extension	N/A
Sets	Set	Yes

Practice

- **Revise Dictionaries** via the reading material L05_reading_Dictionaries.ipynb (especially section about methods)
- Perform Exercise E03_ListsDictionaries.ipynb (practice on lists and dictionaries)
- Please don’t rename the exercise file, fill out the name & collaborator section at the top of the notebook