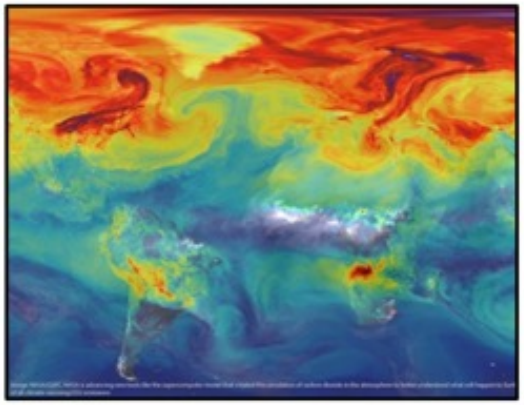


Geo Data Science with Python (GEOS-5984/4984)

Prof. Susanna Werth

Topic: Python Basics

Today’s music is from: Susanna

Please keep sending me your song suggestions through Canvas!

Notes/Reminders

- GitHub repositories
- Homework 1 - submit 2 files:
  - E01 (copy and rename L01_tutorial…)
  - E02 (as is)
- File handling for homework submission:
  - Copy the Exx_*ipyng file into your homework repository folder first, before you start working on homework file.
  - Copy the file and don’t create an empty notebook.
  - **Please don’t rename the file, it makes my work more difficult!**
  - **Instead: Put your name into the homework file at the top!**
- Syllabus Acknowledgement

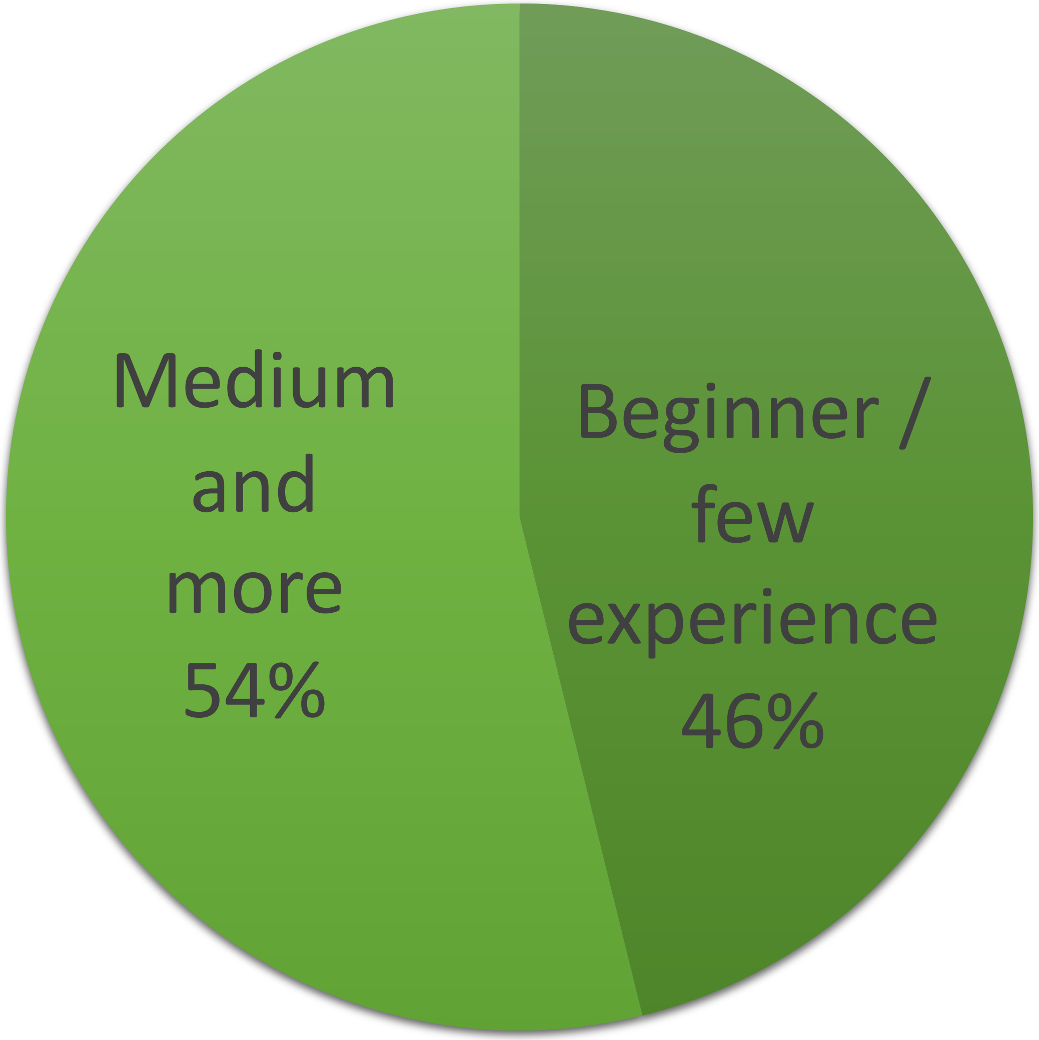
Today

- Survey
- Algorithms and Python Basics
- A taste of Python: Jupyter Notebook practice

Survey Results

Survey Results

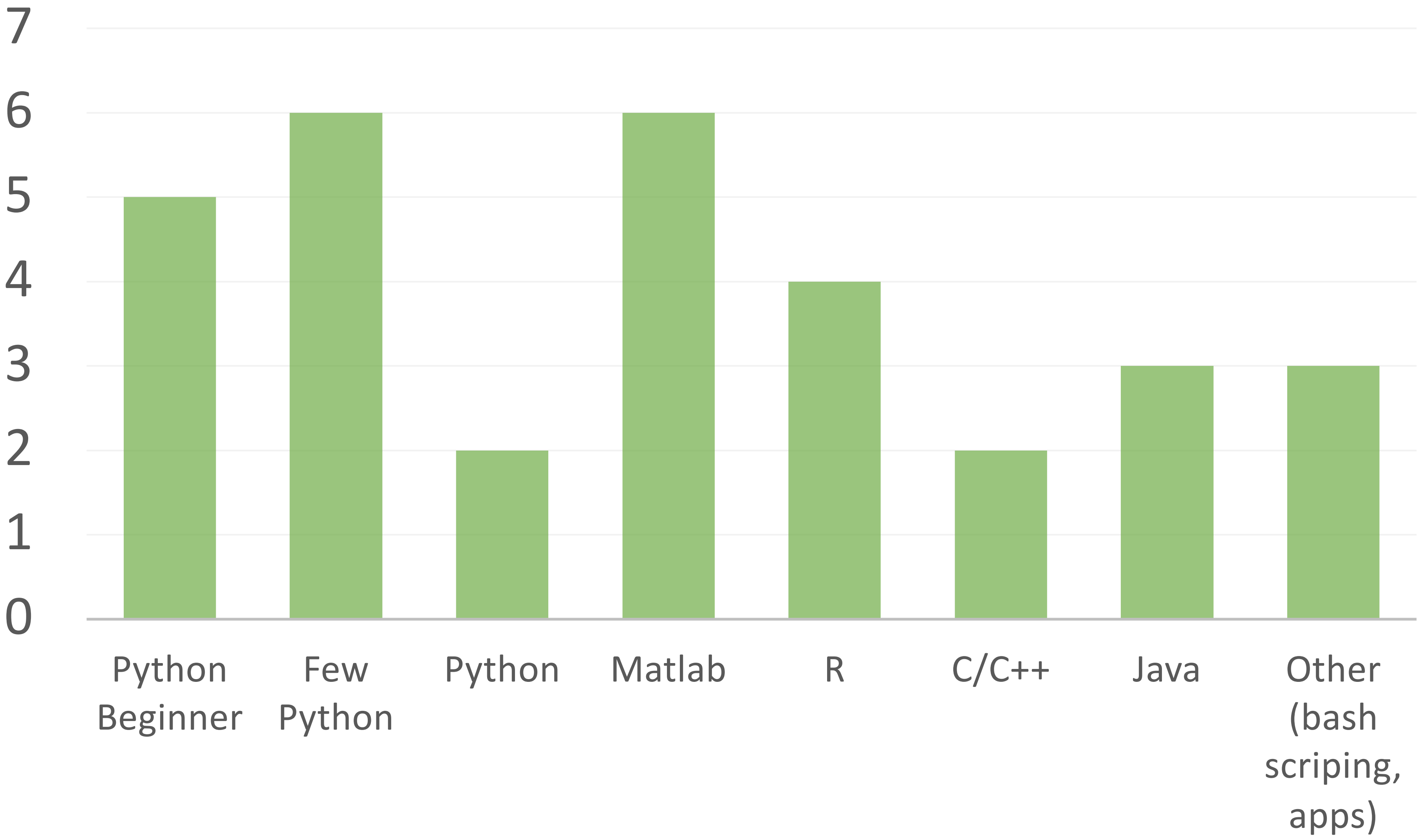
Question: **What is your previous coding experience ?** Total: 13



Survey Results

Question: **Which programming languages?**

Total: 13



Survey Results

What is your expectation for the course?

	Topic	Count
Programming	Build own programs/algorithms & improve experience	3
	Gain Python proficiency: syntax, concept, libraries	7
	Improve processing efficiency of my code	2
	Publish Notebooks online	1
Data Science	Apply coding for research	7
	Process (large and different) data sets	5
	Big data visualization	1
	Machine Learning	0

Algorithms

Terminology: Programming

- **Programming**
  ... process that professionals use to write **code** that instructs how a computer, application or software program performs.

  Computer Programming is the **art of making a computer do what you want it to do**.

- **Code**:
  ... “a **system of rules to convert information** —such as a letter, word, sound, image, or gesture—into another form or representation, sometimes shortened or secret. [...]
  An early example is **the invention of language** which enabled a person, through speech, to communicate what he or she saw, heard, felt, or thought to others.”

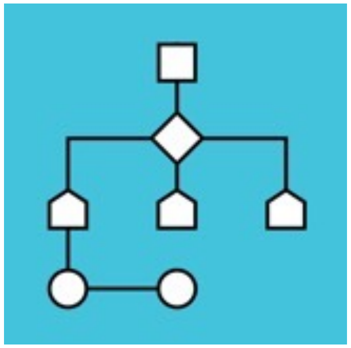
Terminology: Coding

- **Programming Language**
  ... a set of syntax rules that define how **code** should be written and formatted (so that the computer understands it).

- **Coding**:
  ... writing “**program instructions**” (comp. sci.), following syntax (form) & semantics (meaning) of a certain computer language

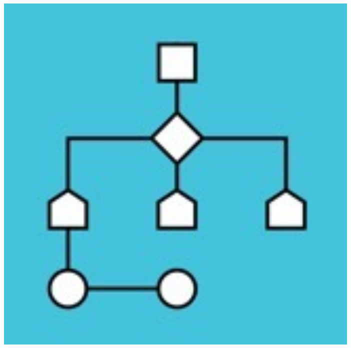
  Coding is the process of **writing an algorithm in a programming language.**

What is an Algorithm?



- Definition: Procedure for solving a (mathematical) problem in a finite number of steps that frequently involves repetition of an operation
  - Example: Mathematics: finding the greatest common divisor of two numbers.
- Broadly: a step-by-step procedure for solving a problem or accomplishing some end (especially by a computer)
- Computer program consist of many algorithms. (Programming is more complex than coding)

Algorithms: Solving Problems



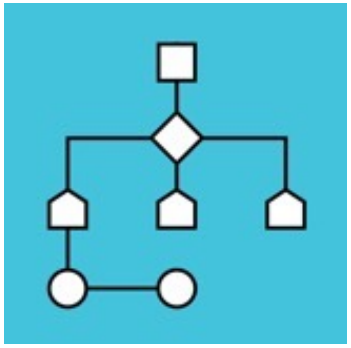
You have a friend arriving at the airport and your friend needs to get from there to your house. Which algorithms could you formulate to instruct your friend?

Some suggestions ?

Take a moment and note down a step-by-step instruction for your friend!

Step-by-step instructions: breaking down a big problem into **small** ones!

Algorithms: Solving Problems



A Friend arrives at the airport

You have a friend arriving at the airport and your friend needs to get from there to your house. Which algorithms could you formulate to instruct your friend?

1) The taxi algorithm:

- Go to the taxi stand.
- Get in a taxi.
- Give the driver my address.

2) The pick-up algorithm:

- When your plane arrives, call my cell phone.
- Meet me outside baggage claim.

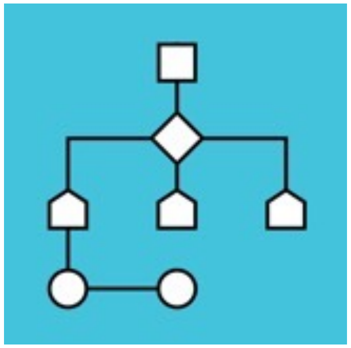
3) The rent-a-car algorithm:

- Take the shuttle to the rental car place.
- Rent a car.
- Follow the directions to get to my house.

4) The bus algorithm:

- Outside baggage claim, catch bus number 70.
- Transfer to bus 14 on Main Street.
- Get off on Elm street.
- Walk two blocks north to my house.

Algorithms: Solving Problems



All four algorithms **accomplish the same goal**, but each in a completely different way

However:

- Each algorithm has different cost and travel time
- Algorithm must be chosen based on circumstance, for example:
  - Priorities
  - Resources available
  - Safety

Essential Algorithms in Coding

Sequential searching: Takes in an array and returns the index of the value we are searching for

5	**44**	**12**	**304**	**2**	**99**	**10**	**1**	**89**	**4**
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10

Any ideas how to do that ???

Sequential Search Algorithm

5	**44**	**12**	**304**	**2**	**99**	**10**	**1**	**89**	**4**
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10

Task: Where is number 44? (find index)

1. Looking at each value in turn (i.e., start with the value in array[0], then array[1], etc).
2. The algorithm quits and returns true if the current value is 44.

Answer: i=2

Advantage: Works on unsorted algorithms.

Disadvantage: Search duration depends on location of searched value.

Can take very long for long arrays.

Binary Search Algorithm

1	**2**	**4**	**5**	**10**	**12**	**44**	**89**	**99**	**304**
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10

Binary Search Task: Where is number 44?

1. Find the midpoint of the sorted array.
2. Compare the midpoint to the value of interest.
3. If the midpoint is larger than the value, perform binary search on right half of the array.
4. If the midpoint is smaller than the value, perform binary search on left half of the array.
5. Repeat these steps until the midpoint value is equal to the value of interest or we know the value is not in the array.

Answer: i=7

Binary Search

1	**2**	**4**	**5**	**10**	**12**	**44**	**89**	**99**	**304**
i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10

Advantage: Requires usually fewer comparisons.
Duration depends only on length of array.
Doubling length of array, requires only one more search step.

Disadvantage: Requires sorted array and needs efficient sort algorithm.

Essential Coding Algorithms

- Searching:
  - Sequential Search
  - Binary Search
- Sorting:
  - Merge sort
  - Bin sort

Find example code in Python for some of these algorithms here:

https://www.codementor.io/learn-programming/3-essential-algorithm-examples-you-should-know

Solving Problems Efficiently

- Identification of advantages and disadvantages in different situations.
- Know the strengths and weaknesses of different algorithms.
- **Run-time is very important for large datasets, so efficient algorithms are essential.**
- Pick the best one for the task at hand.

What is an algorithm?

Video: What is an algorithm, David J. Malan

https://youtu.be/6hfOvs8pY1k

Explore influence of algorithms on daily life:

- TED-talk: www.ted.com/playlists/323/the_influence_of_algorithms
- Press: https://www.theguardian.com/science/2013/jul/01/how-algorithms-rule-world-nsa

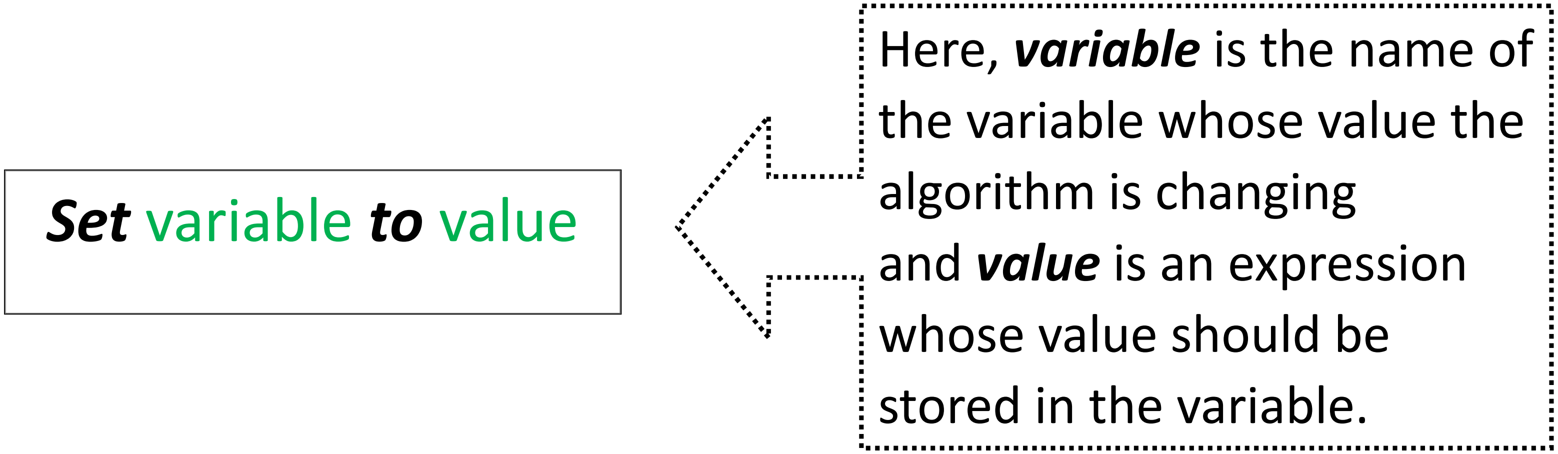
Formulating Algorithms: Pseudo Code

- Pseudo code is a kind of “structured English” for describing algorithms
- No syntax requirements as a programming language would have.
- It allows to focus on the logic of the algorithm.

Four Pillars of Writing Algorithms

1. Assignment
2. Sequence
3. Selection
4. Repetition

1. Assignment



Set variable ***to*** value

Here, ***variable*** is the name of the variable whose value the algorithm is changing and ***value*** is an expression whose value should be stored in the variable.

- Assign or change the value of **variables**
- variables = **data** containers
- In pseudo-code: **set-to**, equals operator **‘=‘** or an ‘**->’**
- Examples:
  - total = 10
  - set total to 0

2. Sequence

- Sequence is a linear progression where one task is performed sequentially after another.
- Set of operations & actions (e.g, mathematical, logical expressions) performed in the sequence (top to bottom)
- Listed one after the other, one on each line (and all having the same indent)

- Examples in pseudo-code: a set of assignments
  - set x to 5
  - set total to 0
  - total -> total + 10
  - X = len(S)
  - i = 10

3. Selection

- Useful when:
  - a sequence should be performed under a certain condition.
  - to make a choice between two alternative courses of action.

```
IF condition THEN
    Sequence 1
ELSE
    Sequence 2
ENDIF
```

Here, ***ELSE*** and **sequence 2** are optional. If the *condition* is true, sequence 1 is performed, otherwise sequence 2 is performed. If *condition* is false and sequence 2 is missing, nothing will execute.

- Example in pseudo-code:
  - **IF-THEN**,
  - **IF-THEN-ELSE**

4. Repetition

- Backbone of automated calculations
- Loops, iterations

```
WHILE condition
        Sequence
ENDWHILE
```

The loop is entered only if the condition is true. The "sequence" is performed for each iteration. After each iteration, the condition is evaluated and the loop continues as long as the condition is true.

- Can be big time consumers and for large datasets it should be applied only when necessary!
- Examples:
  - **WHILE** is a loop with a simple condition testing at its beginning.
  - **FOR**
  - **REPEAT-UNTIL** that are helpful in certain cases.

Summary (in pseudo-code)

Assignment	Variables = data containers
	SET variable to value
	variable = value
	variable -> value

Sequence	writing one action after another, each action on a line by itself, and all actions aligned with the same indent
Selection	IF...THEN...ELSE or IF...THEN...ELSE...ENDIF
Repetition	WHILE...ENDWHILE REPEAT..UNTIL FOR...ENDFOR
Also: Parallelism (making things happen at the same time)	

Python Basics

What kind of programming language is Python?

- High-level (less close to binary code, understood by human)
- Interpreted: no compiler and linker (line-by-line interpretation)
- Relatively old (Created 1991 by Guido van Rossum)
- Open-source
- Philosophy: easy to read code and syntax, flexible (packages, see “import this”)
- Supports multiple programming paradigms & features
  - Paradigms: object-oriented, imperative, functional and procedural
  - Array syntax (Fortran90)
  - Statistical analyses and visualization of data (MATLAB)
  - File management (shell scripting)
  - Scripting language for applications, e.g. Esri’s ArcGIS
- Disadvantages:
  - Slower than compiled code (e.g. Fortran, C++)
  - Sparse software support

4 Pillars of Programming Algorithms

1. Assignment
2. Sequence
3. Selection
4. Repetition

```
total = 0
count = 0
while numbers remain
    total = total + next number
if count > 0
      avg = total/count
else
      error
```

Python:

Tools of traditional pillars contained in Python, but ...

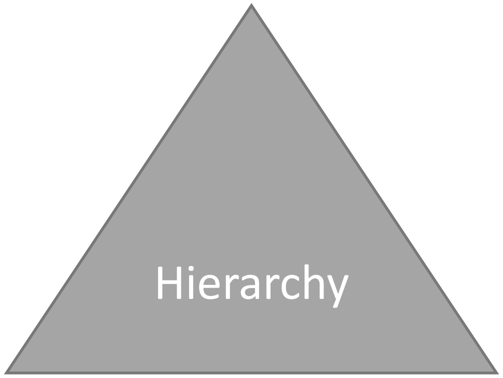
“In Python we do things with stuff”: focused on **objects** and what we can do with them.

In Python, **data** take the form of **objects**, which are most fundamental components in Python programming.

Python Conceptual Hierarchy

Python program components

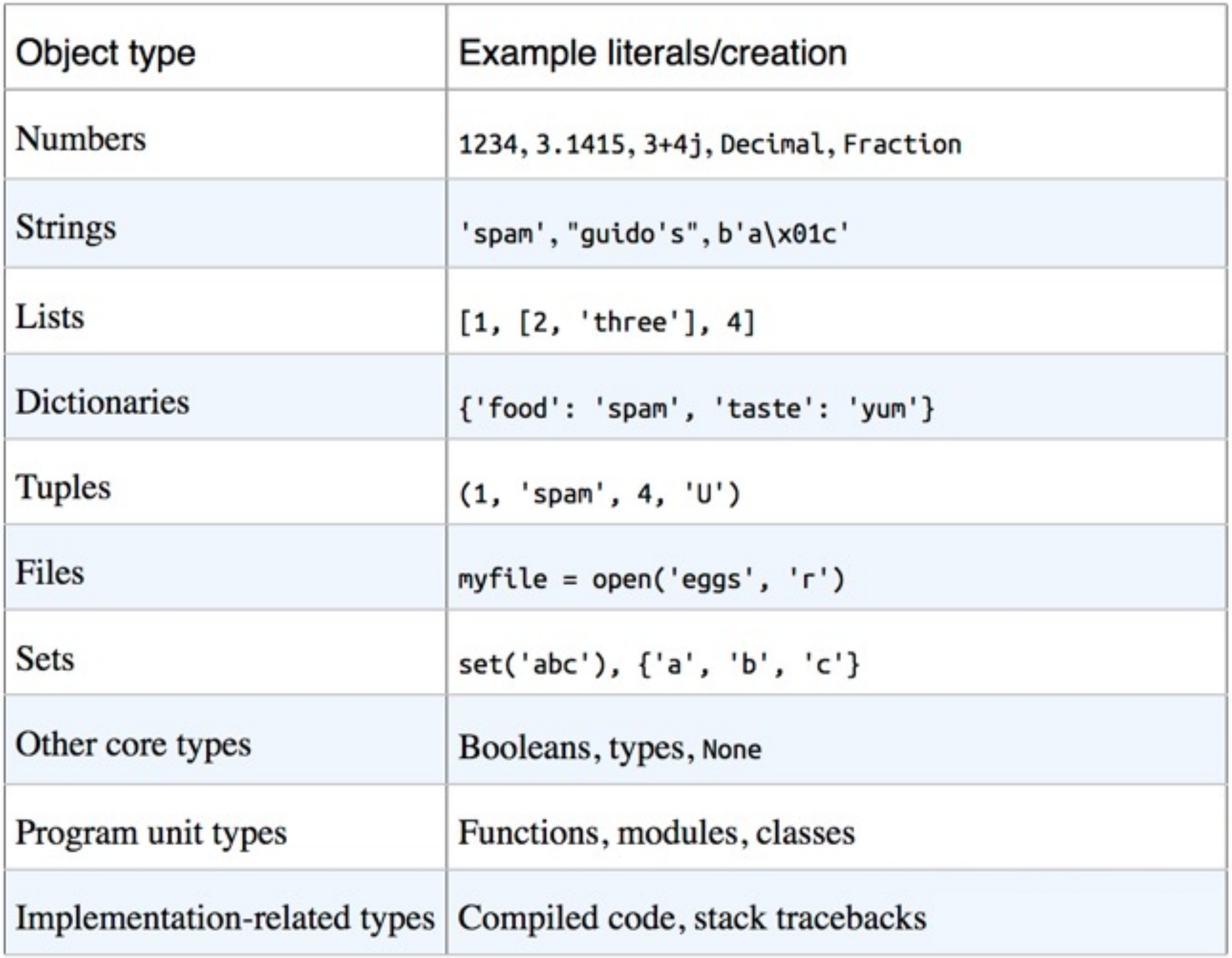
- Programs are composed of ***modules***
- Modules contain ***statements***
- Statements contain ***expressions***
- Expressions create and process ***objects***



- *Objects* are data elements (e.g. variables, functions, ...)
- *Expression* is a **combination of one or more objects** that the programming language interprets and computes to **produce another object**. They are embedded in statements.
- *Statements* code the larger logic of a program (e.g. assignments, repetitions, selections, ...)
- *Modules* are highest-level organization unit, packages code for reuse

Core Object Types

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.

General Python Concepts

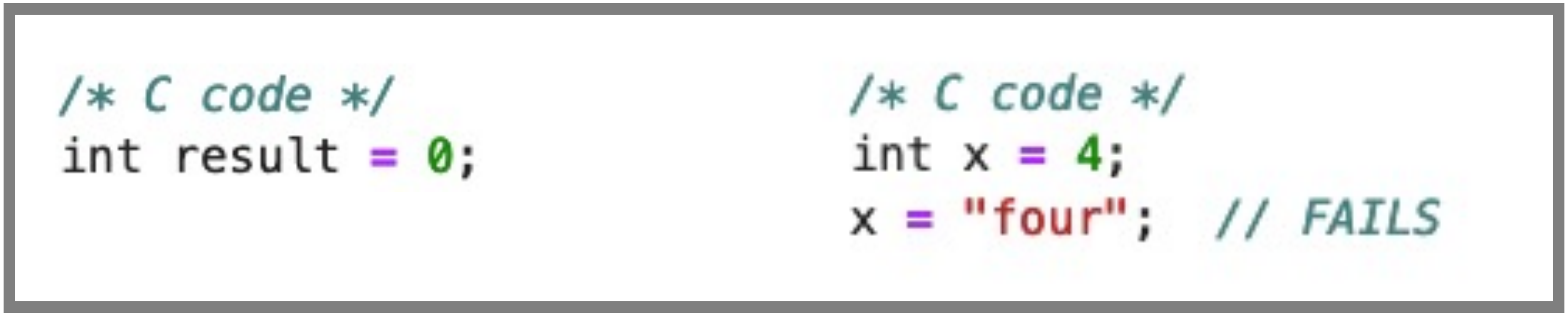
A. Dynamically type

B. Strongly type

C. Mutability

Python Assignment

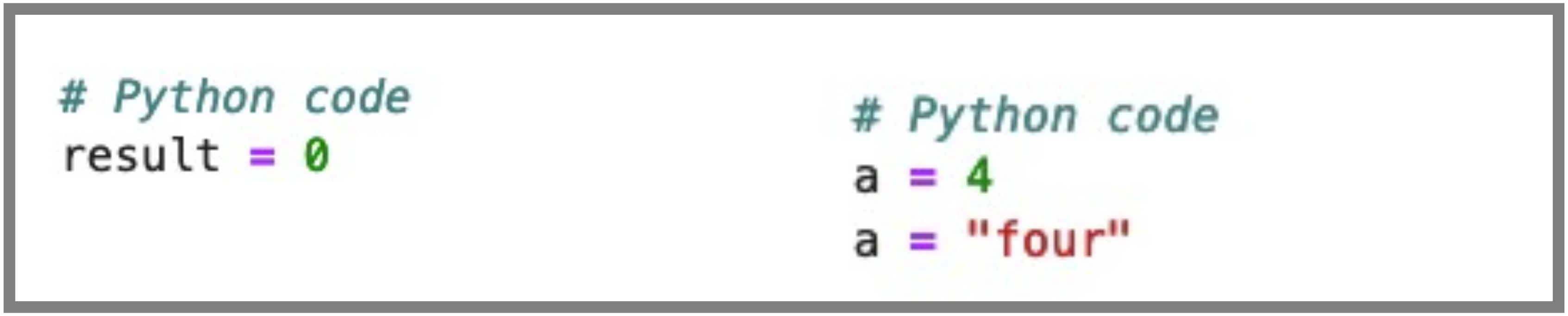
C



```c
/* C code */
int result = 0;
```

```c
/* C code */
int x = 4;
x = "four";  // FAILS
```

Python



```python
# Python code
result = 0
```

```python
# Python code
a = 4
a = "four"
```

General Python Concepts

A. Dynamically typed (vs. statically typed)

- Python automatically figures out the type of a variable whet you first assign it a value (no need to declare)
- Types is associated with object, not with variable name

- **Python Assignment**:

  creates object > creates variable > links them

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.

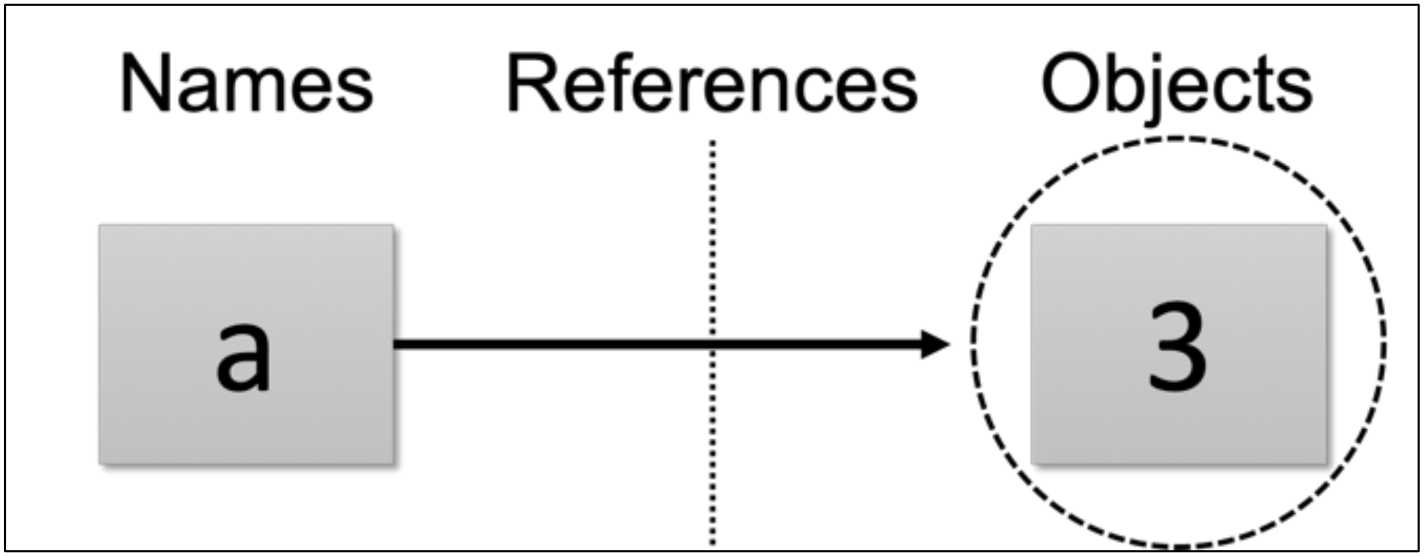


Figure 6-1. Names and objects after running the assignment a = 3. Variable a becomes a reference to the object 3. Internally, the variable is really a pointer to the object's memory space created by running the literal expression 3.

Lutz (2013)

General Python Concepts

B. Strongly Typed (vs. Weakly Typed)

- Type of a given object does not change
- You can only perform operations that are valid for the object’s type.
- Type specific operations:
  - Numerous built-in tools: functions, methods
  - **Methods** are a set of **type associated operations** (more precisely & later: method is a function that takes a class instance as its first parameter, methods are members of classes)

General Python Concepts

C. Mutable vs. Immutable

- Python objects can **mutate**, at least some of them
- Mutability describes the possibility of **in-place** modification of objects by calling functions or methods
- These functions typically may not have any output, but change the input object.

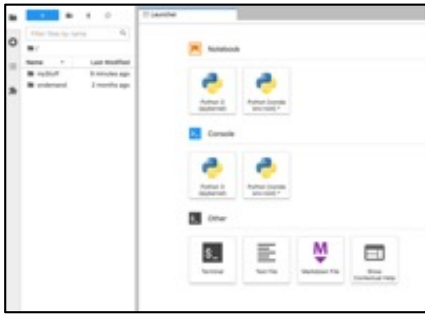
- numbers, strings, and tuples are *immutable*
- lists, dictionaries, and sets are *mutable*

A Taste of Python

Let’s look at some examples

Working with tutorial Notebooks

- Open the tutorial book: L03_tutorial-empty_ATasteOfPython.ipynb
- Copy the file to a separate folder (not your homework repository), to work on it
- Let’s start studying it together



Note: if you want to work on a tutorial notebook, it is better to copy it to a separate folder (not your homework repository)

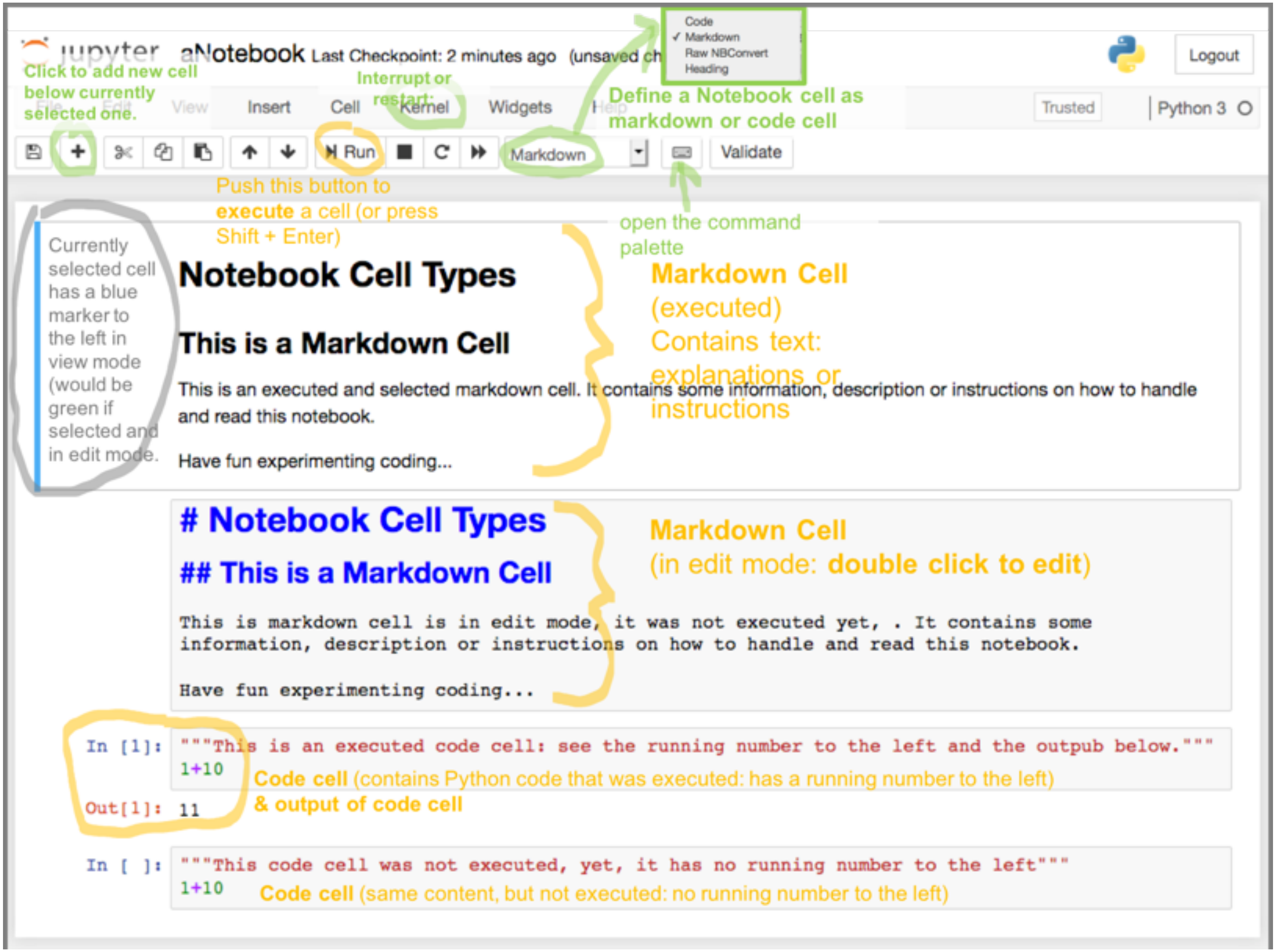
Working with homework Notebooks

- Open the tutorial book: E02_ATasteOfPython.ipynb

- **Note: also if you work on a homework notebook, it is better to copy it to your homework repository folder first**
- Let's start working on it together

- If you already worked on that, study the notebook L03_reading_PythonConcepts.ipynb

Jupyter Notebooks



Click to add new cell below currently selected one.

Interrupt or restart:

Define a Notebook cell as markdown or code cell

Push this button to **execute** a cell (or press Shift + Enter)

open the command palette

Currently selected cell has a blue marker to the left in view mode (would be green if selected and in edit mode.

Notebook Cell Types

This is a Markdown Cell

This is an executed and selected markdown cell. It contains some information, description or instructions on how to handle and read this notebook.

Have fun experimenting coding...

Markdown Cell (executed) Contains text: explanations or instructions

```
# Notebook Cell Types
## This is a Markdown Cell

This is markdown cell is in edit mode, it was not executed yet, . It contains some
information, description or instructions on how to handle and read this notebook.

Have fun experimenting coding...
```

Markdown Cell (in edit mode: **double click to edit**)

```
In [1]: """This is an executed code cell: see the running number to the left and the outpub below."""
        1+10
Out[1]: 11
```

Code cell (contains Python code that was executed: has a running number to the left)
& output of code cell

```
In [ ]: """This code cell was not executed, yet, it has no running number to the left"""
        1+10
```

Code cell (same content, but not executed: no running number to the left)

Practice

- Make sure you are fully able to **submit homework**
- Submit/update the fully completed E02_ATasteOfPython.ipynb in your repository *geosf22_<yourPID>*

- Optional:
  - Revise today's content with the notebook: L03_reading_PythonConcepts.ipynb
  - Video and reading material about Algorithms (listed on previous slides)