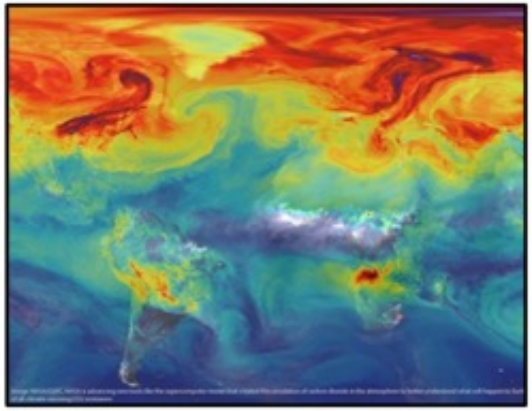





Geo Data Science with Python (GEOS-5984/4984)

Prof. Susanna Werth

Topic: Python Basic Object Types

Today's music is from: Carmen

Please keep sending me your song suggestions through Canvas!

Notes/Reminders

- I need some music from you!
- Let me know, if you are not fully set with computer setup by now!

- Literature

Programing

- **Development of a set of complex algorithms**
- **Coding of the all components of the complex algorithms**
- **Testing, Optimization and Documentation**

To program you have to code.

But to code you don’t have to program.

Building Blocks of a Program

- **Input** Algorithms
  - Get data from the the "outside world”
  - Reading data from a file, or some kind of sensor
  - User input on keyboard
  - Store data in variables
- **Data processing Algorithms**
  - Assignment
  - Sequence
  - Selection
  - Repetition
- **Output** Algorithms
  - Display the results of the program on a screen
  - Store results in a file or a device.

To learn how to write program in Python, we will start with data containers (= object types) & related basic concepts

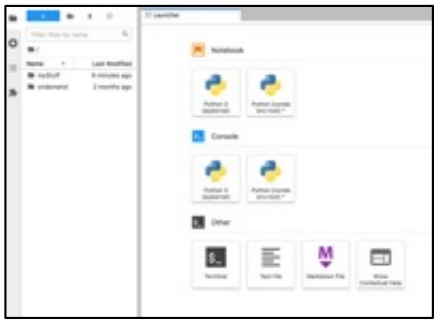
Reflection

“In Python we do things with stuff”:

… focused on **objects** and what we can do with them.

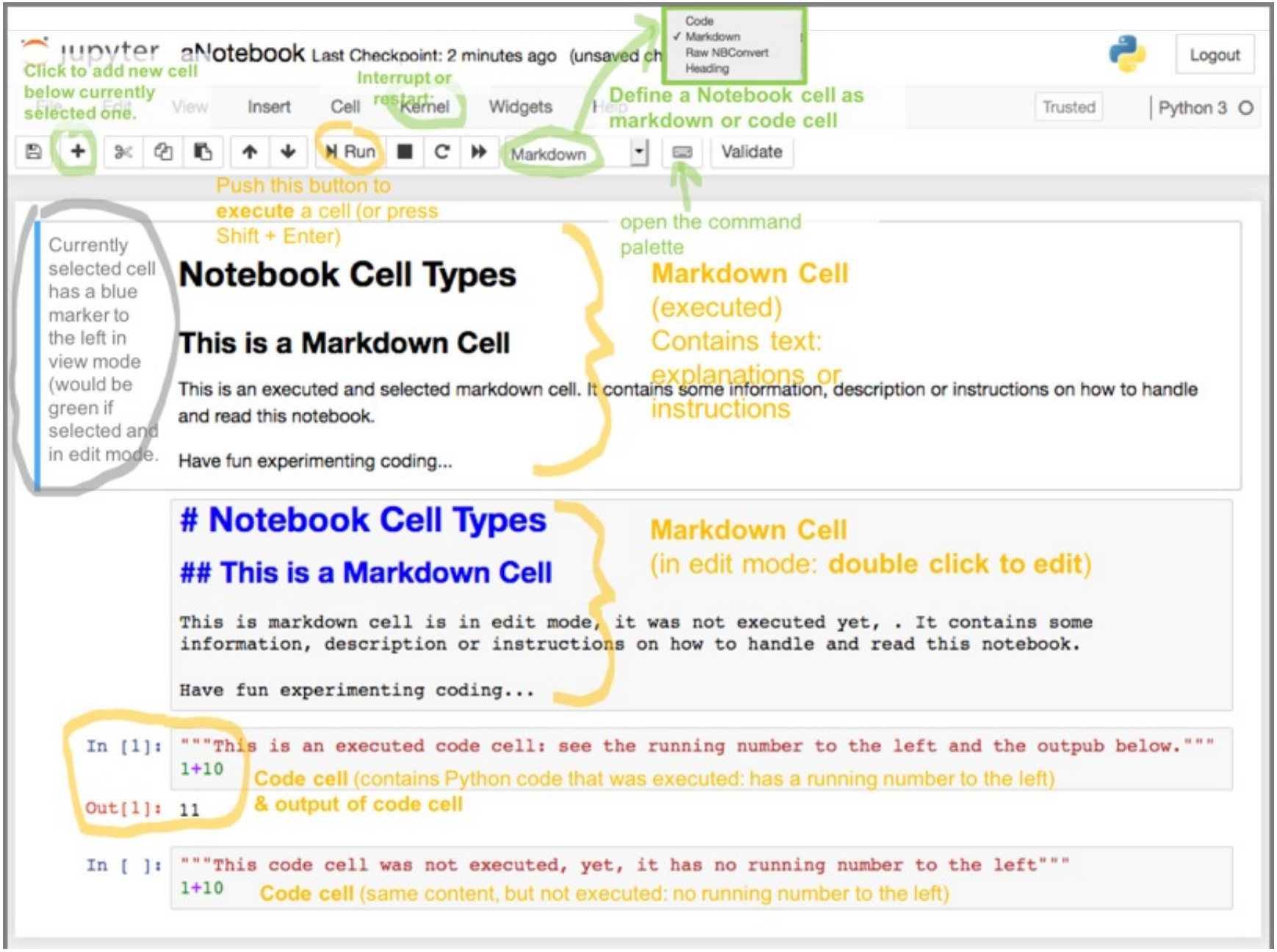
- Simplest objects are data containers in Python

Tutorials Today & Next Class



- Basic Object Types
  - Numbers
  - Boolean
  - Strings
  - Lists
- Advanced Object Types
  - Dictionaries
  - *Tuples*
  - *Sets*

Open a Jupyter Notebook !



Using the correct Python environment during exercises

- Anaconda: change from base to geosf22
- Command window / Terminal, type

```
conda activate geosf22
```

- Jupyter Notebook: change Kernel to geosf22

Installation commands for this environment

```
conda create --name geosf22

conda activate geosf22

conda install -c conda-forge boost python=3.9
nb_conda geopandas jupyterlab scipy cartopy
scikit-learn basemap statsmodels netcdf4 hdf5
pywget lxml pydap pywavelets seaborn xarray
```

Alternative option on GitHub

- Open a notebook (e.g. L03_...)
- Press the key "."
- Wait for magic to happen!

Opens **github.dev** web-based editor

- lightweight editing experience that runs entirely in your browser
- https://docs.github.com/en/codespaces/the-githubdev-web-based-editor

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.

Numeric Types in Python

Table 5-1. Numeric literals and constructors

Literal	Interpretation
`1234, -24, 0, 99999999999999`	Integers (unlimited size)
`1.23, 1., 3.14e-10, 4E210, 4.0e+210`	Floating-point numbers
`0o177, 0x9ff, 0b101010`	Octal, hex, and binary literals in 3.X
`0177, 0o177, 0x9ff, 0b101010`	Octal, octal, hex, and binary literals in 2.X
`3+4j, 3.0+4.0j, 3J`	Complex number literals
`set('spam'), {1, 2, 3, 4}`	Sets: 2.X and 3.X construction forms
`Decimal('1.0'), Fraction(1, 3)`	Decimal and fraction extension types
`bool(X), True, False`	Boolean type and constants

Lutz (2013). pg. 138, Ch. 5

Built-in Tools for Numbers

Expression Operators

- +, -, *, /, %, **, &, …
- Comprehensive overview: www.tutorialspoint.com/python/python_basic_operators.htm

Built-in mathematical functions

- int()
- pow()
- round()
- all functions listed at https://docs.python.org/3/library/functions.html

Utility modules

- math https://docs.python.org/3/library/math.html
  - math.ceil()
  - math.floor()
- random https://docs.python.org/3/library/random.html

Not built-in external packages are necessary for more numeric tools (NumPy, Pandas, …).

Python HELP ?!

If you ever need help on a function, use the help function for that:

>>> *help(int)*

>>> *help(math.ceil)*

Numbers

Go to the Python documentation pages & …

a) find at least one **built-in functions** that are accepting and **useful for numbers** in Python:
https://docs.python.org/3/library/functions.html

b) study and learn to use at least one **function of the math module**, for integer or floating-point numbers:
https://docs.python.org/3/library/math.html

Apply the Python help-function for your investigations.

Boolean

Table 5-1. Numeric literals and constructors

Literal	Interpretation
`1234, -24, 0, 99999999999999`	Integers (unlimited size) ✅
`1.23, 1., 3.14e-10, 4E210, 4.0e+210`	Floating-point numbers ✅
`0o177, 0x9ff, 0b101010`	Octal, hex, and binary literals in 3.X
`0177, 0o177, 0x9ff, 0b101010`	Octal, octal, hex, and binary literals in 2.X
`3+4j, 3.0+4.0j, 3J`	Complex number literals
`set('spam'), {1, 2, 3, 4}`	Sets: 2.X and 3.X construction forms
`Decimal('1.0'), Fraction(1, 3)`	Decimal and fraction extension types
`bool(X), True, False`	Boolean type and constants

Lutz (2013). pg. 138, Ch. 5

Numbers & Boolean

Boolean constants are resulting from:

- Comparisons and Equality Tests: **red**
- Boolean Tests: **yellow**, **orange**

Expression sheet:

- Numeric operators: blue

Python Expression Operators

Table 5-2. Python expression operators and precedence

Operators	Description
`yield x`	Generator function `send` protocol
`lambda args: expression`	Anonymous function generation
`x if y else z`	Ternary selection (`x` is evaluated only if `y` is true)
`x or y`	Logical OR (`y` is evaluated only if `x` is false)
`x and y`	Logical AND (`y` is evaluated only if `x` is true)
`not x`	Logical negation
`x in y, x not in y`	Membership (iterables, sets)
`x is y, x is not y`	Object identity tests
`x < y, x <= y, x > y, x >= y` `x == y, x != y`	Magnitude comparison, set subset and superset; Value equality operators
`x | y`	Bitwise OR, set union
`x ^ y`	Bitwise XOR, set symmetric difference
`x & y`	Bitwise AND, set intersection
`x << y, x >> y`	Shift `x` left or right by `y` bits
`x + y` `x - y`	Addition, concatenation; Subtraction, set difference

Operators	Description

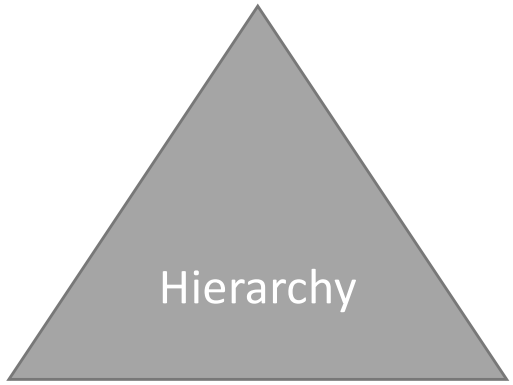
`x * y`
`x % y`
`x / y, x // y`
	Multiplication, repetition;
	Remainder, format;
	Division: true and floor

`-x, +x`	Negation, identity
`~x`	Bitwise NOT (inversion)
`x ** y`	Power (exponentiation)
`x[i]`	Indexing (sequence, mapping, others)
`x[i:j:k]`	Slicing
`x(...)`	Call (function, method, class, other callable)
`x.attr`	Attribute reference
`(...)`	Tuple, expression, generator expression
`[...]`	List, list comprehension
`{...}`	Dictionary, set, set and dictionary comprehensions

Logic & bitwise logic operators
Comparison operators
Assignment/numeric operators
Object access operators

Python Conceptual Hierarchy

- 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

Boolean

What is the Boolean value of the following expressions and why?

a = 0

b = 5

c = 10.0

- b or c
- a or c
- a and c
- a < c and c > a
- a and b or c
- bool(a and b or c)

Python **Strings**

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.

Formatted String Output and Type Codes

print('text %s' %(strVar))

Table 7-4. String formatting type codes

Code	Meaning
s	String (or any object's `str(X)` string)
r	`s`, but uses `repr`, not `str`
c	Character
d	Decimal (integer)
i	Integer
u	Same as `d` (obsolete: no longer unsigned)
o	Octal integer
x	Hex integer
X	`x`, but prints uppercase
e	Floating-point exponent, lowercase
E	Same as `e`, but prints uppercase
f	Floating-point decimal
F	Floating-point decimal
g	Floating-point `e` or `f`
G	Floating-point `E` or `F`
%	Literal `%`

Lutz (2013)

format() method

- string.*format()* creates a new string object and allows the user to control the formatting of the object.

- String formatting, not just for display
- **{}** functions as place holder
- See more examples in the reading notebook

String Methods & Functions

- String Methods Examples
  - .format() string formatting https://pyformat.info/
  - .find('pa') find substring in a string
  - .split(',') split on delimiter ','
  - .replace('n','m') …?
  - lower, upper, rstrip, …

- String built-in functions
  - print() print string to display
  - len() length of string
  - str() converts number to string
  - …

Strings: Summary

```
>>> S = 'spam'      # creates a string
```

- **Immutability** (no in-place modification)
- **Sequence**:
  - "A positionally ordered collection of other objects"
  - left-to-right order

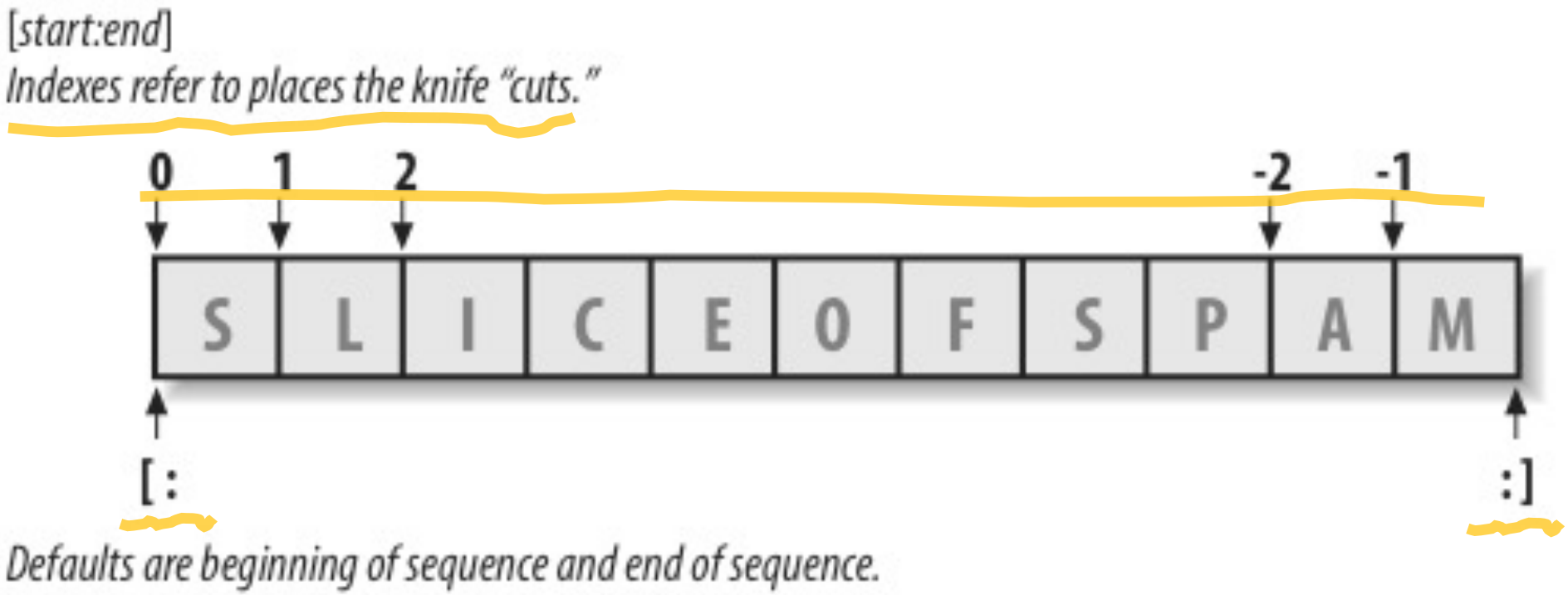
Sequence Operations

- Indexing
- Slicing
- Concatenation
- Repetition

Strings are Sequences

Sequence operands:

- **Indexing**: Fetches components, from front or end
- **Slicing**: Extracts a contiguous section



Slicing

What is the correct slice for the first and last name of the string “Tony McQuire”?

What is the correct slice for the last name sliced from the end only?

Strings are Sequences

Sequence operands:

- **Concatenation**: X + Y makes a new sequence object with the contents of both operands.

- **Repetition**: X * N makes a new sequence object with N copies of the sequence operand X.

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

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

Practice

- **Read part D of** L04_reading_BasicObjectTypes.ipynb (revising Python lists and sequences)
- Nothing to submit
  (practice on object types will follow next week)