Lecture 2- Crash Course in Python

ERT 429/529

Geological Data Analysis

Jan 27th 2026

Python

- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - Statistical analysis
  - Visualize

Python

- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - **Access Data**
    - Time series dataset
    - Geospatial datasets
  - Statistical analysis
  - Visualize

Python

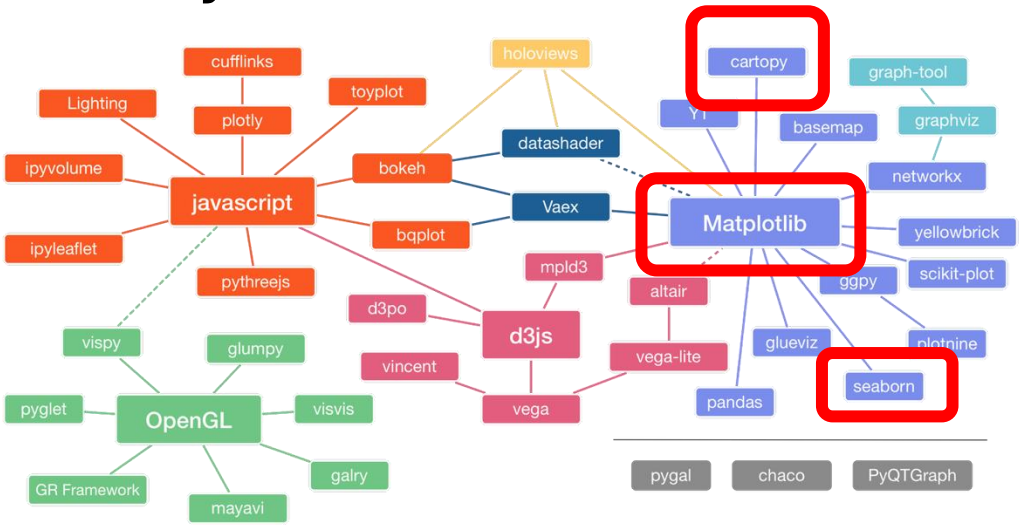
- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - **Statistical analysis**
    - Universal statistics (confidence intervals, hypothesis testing, Bayes theorem, distributions, resampling techniques, regression models, etc.)
    - Time series analysis (seasonality, decomposition)
    - Geospatial data analysis
  - Visualize

Python

- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - Statistical analysis
  - **Visualize**

Python

- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - Statistical analysis
  - **Visualize**
    - Matplotlib
    - Cartopy (Maps)
    - Seaborn (high-level interface)



Source: https://pyviz.org/overviews/index.html

An example practice

Please write a function replicate_strings

This function will duplicate the string X times.

For example, list1 = [‘a’,’b’,’c’], list2 = [1,2,3],

it will output a third list ['a','bb','ccc'].

An example practice

Please write a function **replicate_strings**

This function will duplicate the string X times.

For example, list1 = [‘a’,’b’,’c’], list2 = [1,2,3],

it will output a third list ['a','bb','ccc'].

How do I turn a real-world idea into something a computer can understand step-by-step?

An example practice

Please write a function **replicate_strings**

This function will duplicate the string X times.

For example, list1 = [‘a’,’b’,’c’], list2 = [1,2,3],

it will output a third list ['a','bb','ccc'].

1. What is this function supposed to do?
2. If Python were a machine, what raw materials does it need?
3. What do we do for each string-number pair?

Programming is about finding the repeated rule.

An example practice

Please write a function **replicate_strings**

This function will duplicate the string X times.

For example, list1 = [‘a’,’b’,’c’], list2 = [1,2,3],

it will output a third list ['a','bb','ccc'].

Step-by-Step Operation

1. Start with an empty `result` list
2. Look at the first pair: 'a' and 1
3. Repeat 'a' → 'a'
4. Append it to the `result` list
5. Move to the next pair
6. Repeat until the end of the lists

An example practice

Please write a function **replicate_strings**

This function will duplicate the string X times.

For example, list1 = [‘a’,’b’,’c’], list2 = [1,2,3],

it will output a third string ['a','bb','ccc'].

```python
def replicate_strings(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] * list2[i])
    return result

# Example usage:
list1 = ['a', 'b', 'c']
list2 = [1, 2, 3]
print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']

```

Python

- Variables
- Data Types
- Operators
- Control Flow Functions
- Indentation
- Basic Data Structures
- Comments

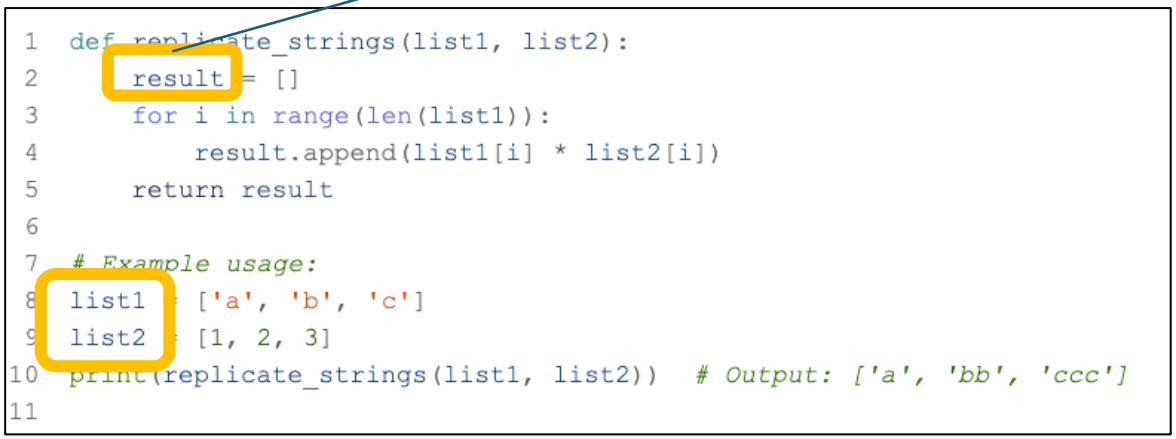
```python
def replicate_strings(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] * list2[i])
    return result

# Example usage:
list1 = ['a', 'b', 'c']
list2 = [1, 2, 3]
print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']

```

Python

- **Variables**
- Data Types
- Operators
- Control Flow Functions
- Indentation
- Basic Data Structures
- Comments

Local Variables

- Defined **inside** a function.
- Only exist **while the function runs**.
- **Not accessible** outside the function.



```
1  def replicate_strings(list1, list2):
2      result = []
3      for i in range(len(list1)):
4          result.append(list1[i] * list2[i])
5      return result
6
7  # Example usage:
8  list1 = ['a', 'b', 'c']
9  list2 = [1, 2, 3]
10 print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
11
```

Python

- Variables
- **Data Types**
- Operators
- Control Flow Functions
- Indentation
- Basic Data Structures
- Comments

```python
def replicate_strings(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] * list2[i])
    return result

# Example usage:
list1 = ['a', 'b', 'c']
list2 = [1, 2, 3]
print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
```

String

Integer

Python

- Variables
- **Data Types**
- Operators
- Control Flow Functions
- Indentation
- Basic Data Structures
- Comments

Type	Name	Example	Description
int	Integer	5, -3, 100	Whole numbers
float	Floating point	3.14, -0.5, 2.0	Decimal numbers
str	**String**	**"hello", 'abc'**	**Text data**
bool	Boolean	True, False	Logical values
list	**List**	**[1, 2, 3], ['a', 'b']**	**Ordered, changeable collection**
tuple	**Tuple**	**(1, 2), ('x', 'y')**	**Ordered, unchangeable collection**
dict	**Dictionary**	**{'a': 1, 'b': 2}**	**Key-value pairs**
set	**Set**	**{1, 2, 3}**	**Unordered collection of unique items**
NoneType	None	None	Represents absence of a value

***string, list, tuple, dict, and set will be discussed in data structures as well**

Python

- Variables
- Data Types
- **Operators**
- Control Flow Functions
- Indentation
- Basic Data Structures
- Comments

```
1  def replicate_strings(list1, list2):
2      result = []
3      for i in range(len(list1)):
4          result.append(list1[i] * list2[i])
5      return result
6
7  # Example usage:
8  list1 = ['a', 'b', 'c']
9  list2 = [1, 2, 3]
10 print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
11
```

Arithmetic operator

Assignment operator

- Comparison operators
  - Examples: `==, !=, >, <, >=, <=`
- Logical operators
  - Examples: `and, or, not`

Practice

- What is x?
  - `x = True or False`
  - `x = True and False`
  - `x = 5 > 4`

Operator Type	Examples	Precedence Level
Parentheses	( )	Highest
Exponentiation	**	
Unary operators	+x, -x, ~x	
Multiplication/Division	*, /, //, %	
Addition/Subtraction	+, -	
Comparison	==, <, >	
Logical	not, and, or	
Assignment	=, +=, -=	Lowest

Python

- Variables
- Data Types
- Operators
- **Control Flow**
  Functions
- Indentation
- Basic Data Structures
- Comments

```python
def replicate_strings(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] * list2[i])
    return result

# Example usage:
list1 = ['a', 'b', 'c']
list2 = [1, 2, 3]
print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']

```

- if statement
  - Example: `if,elif,else`
- Loops
  - `for` loop
  - `while` loop

Python

- Variables
- Data Types
- Operators
- **Control Flow**
  Functions
- Indentation
- Basic Data Structures
- Comments

Example (if statement)

```
if x > 3:
    print(“x is larger than 3”)
elif x == 3:
    print(“x equals to 3”)
else:
    print(”x is smaller than 3”)
```

Python

- Variables
- Data Types
- Operators
- **Control Flow**
  Functions
- Indentation
- Basic Data Structure
- Comments

Examples (for loop)

```
for i in range(5):
    print(i)
```

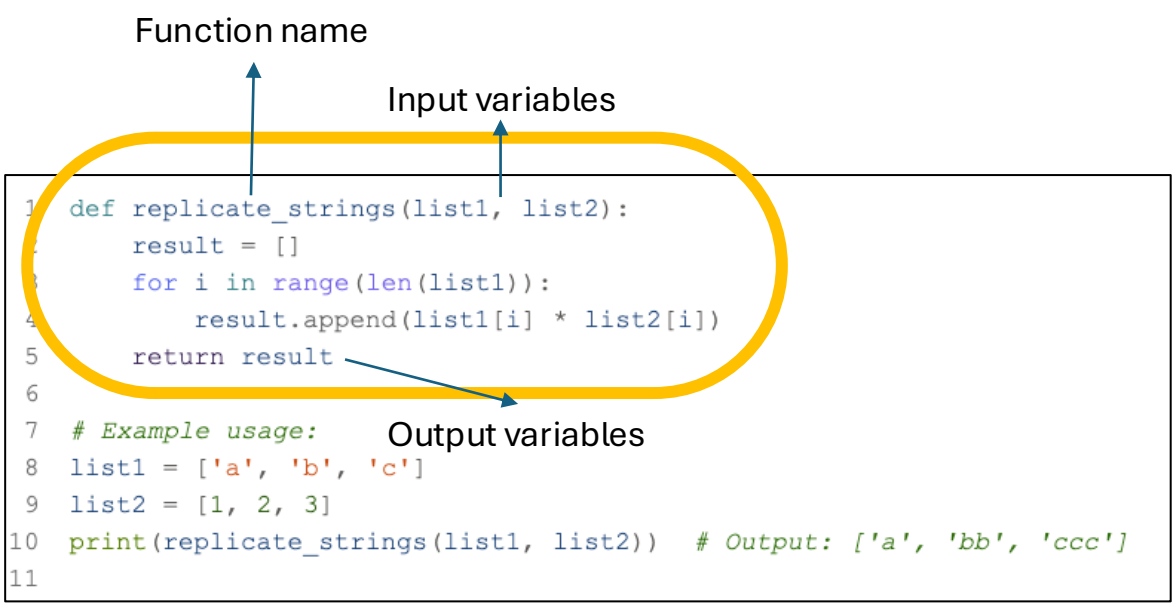
```
namelist = ['Tom','Lisa','Jim']
for v in namelist:
    print(i)
```

```
namelist  = ['Tom','Lisa','Jim']
scorelist = [88,93,91]
for rank,name in enumerate(namelist):
    print(f"The score for {name} is
{scorelist[rank]}")
```

```
for name,score in zip(namelist,scorelist):
    print(f"The score for {name} is {score}")
```

Python

- Variables
- Data Types
- Operators
- Control Flow
  Functions
- Indentation
- Basic Data Structures
- Comments



Python

- Variables
- Data Types
- Operators
- Control Flow Functions
- **Indentation**
- Basic Data Structures
- Comments

"If it belongs to a block, it must be indented."

```
1  def replicate_strings(list1, list2):
2      result = []
3      for i in range(len(list1)):
4          result.append(list1[i] * list2[i])
5      return result
6
7  # Example usage:
8  list1 = ['a', 'b', 'c']
9  list2 = [1, 2, 3]
10 print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
11
```

What to indent:

- Code inside **functions**, **loops**, **conditionals**, and **classes**.
- Typically **4 spaces** (or 1 tab) per level.

Python

- Variables
- Data Types
- Operators
- Control Flow Functions
- Indentation
- **Basic Data Structures**
- Comments

```
1  def replicate_strings(list1, list2):
2      result = []
3      for i in range(len(list1)):
4          result.append(list1[i] * list2[i])
5      return result
6
7  # Example usage:
8  list1 = ['a', 'b', 'c']
9  list2 = [1, 2, 3]
10 print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
11
```

list

Python

- Variables
- Data Types
- Operators
- Control Flow Functions
- Indentation
- **Basic Data Structures**
- Comments

Data Structure	Example	Description
list	[1, 2, 3]	Ordered, changeable, allows duplicates
tuple	(1, 2, 3)	Ordered, unchangeable, allows duplicates
dict	{'a': 1, 'b': 2}	Key-value pairs, unordered (ordered since Python 3.7)
set	{1, 2, 3}	Unordered, no duplicates
str	"hello"	Sequence of characters (also behaves like a list)

Python

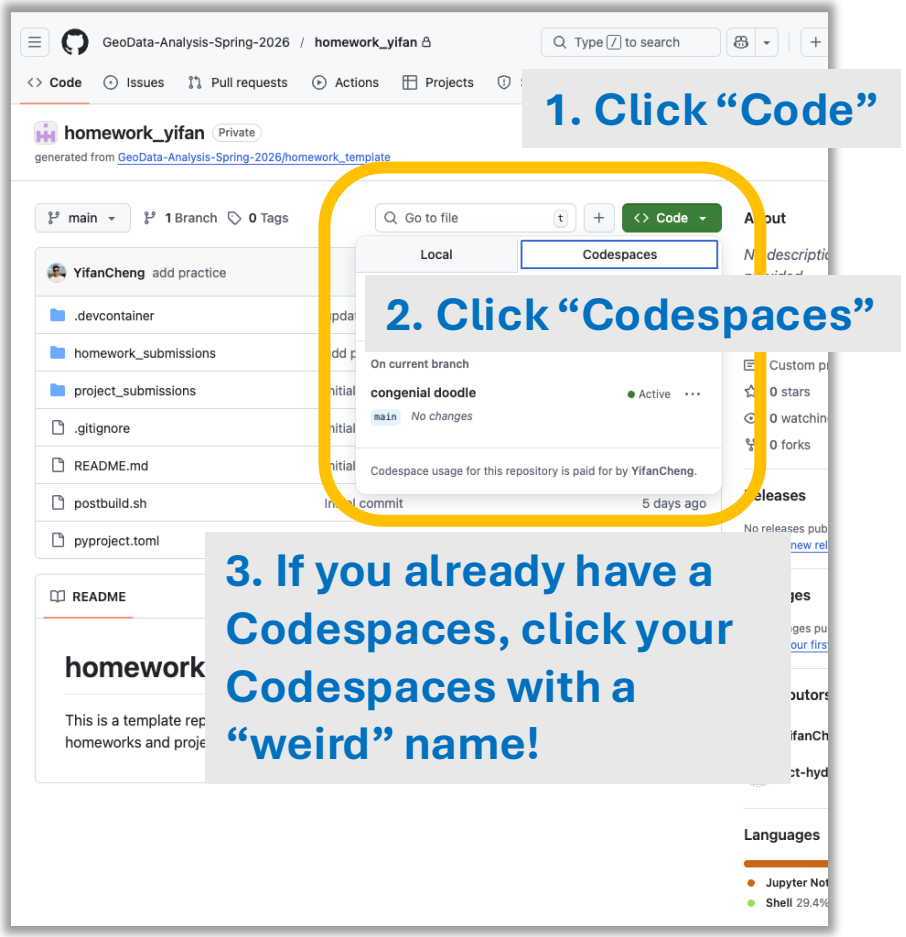
- Variables
- Data Types
- Operators
- Control Flow Functions
- Indentation
- Basic Data Structures
- **Comments**

```
1  def replicate_strings(list1, list2):
2      result = []
3      for i in range(len(list1)):
4          result.append(list1[i] * list2[i])
5      return result
6
7  # Example usage:
8  list1 = ['a', 'bb', 'ccc']
9  list2 = [1, 2, 3]
10 print(replicate_strings(list1, list2))  # Output: ['a', 'bb', 'ccc']
11
```

- All text after the “#” is a comment.
- “#” can be added in the beginning or in the middle of a line.
- A comment can be an explanation to a specific line or a chunk of codes, which greatly increases the readability of a code

Let’s practice Python in GitHub CodeSpace!

- Let’s go to our course organization https://github.com/orgs/GeoData-Analysis-Spring-2026
- And click your homework repo!



26