INTRO TO PYTHON

ERT 474/574

Open-Source Hydro Data Analytics

Sep 8th 2025

University at Buffalo The State University of New York

Python

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

Python

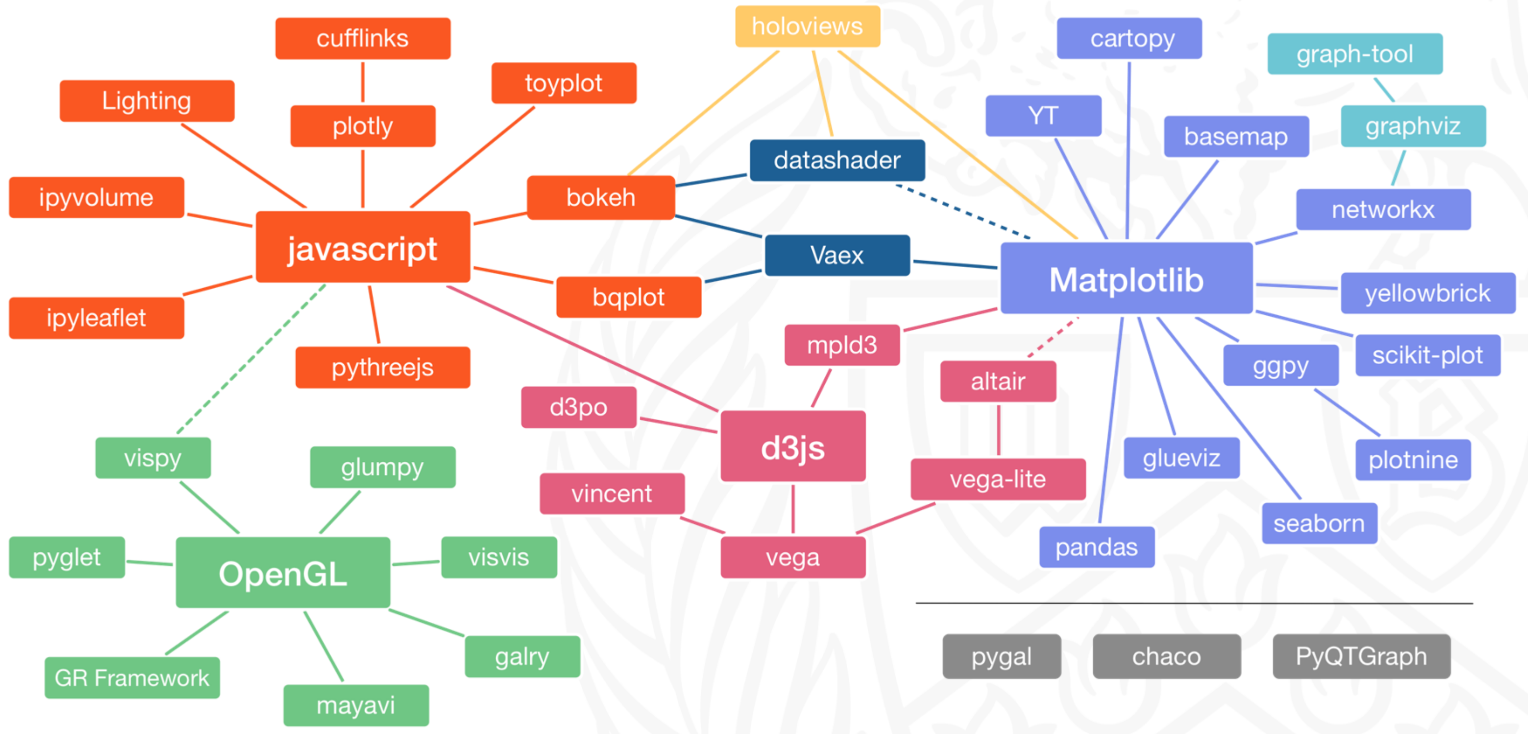
- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
    - **Access Data**
        - Time series dataset
        - Geospatial datasets
    - Data analysis
    - Visualize

Python

- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - **Data analysis**
    - Basic statistics (such as mean/max/min across monthly/daily/seasonal scales)
    - Hypothesis testing, confidence intervals, etc.
    - Time series analysis (seasonality, decomposition)
    - Geospatial data analysis
  - Visualize

Python

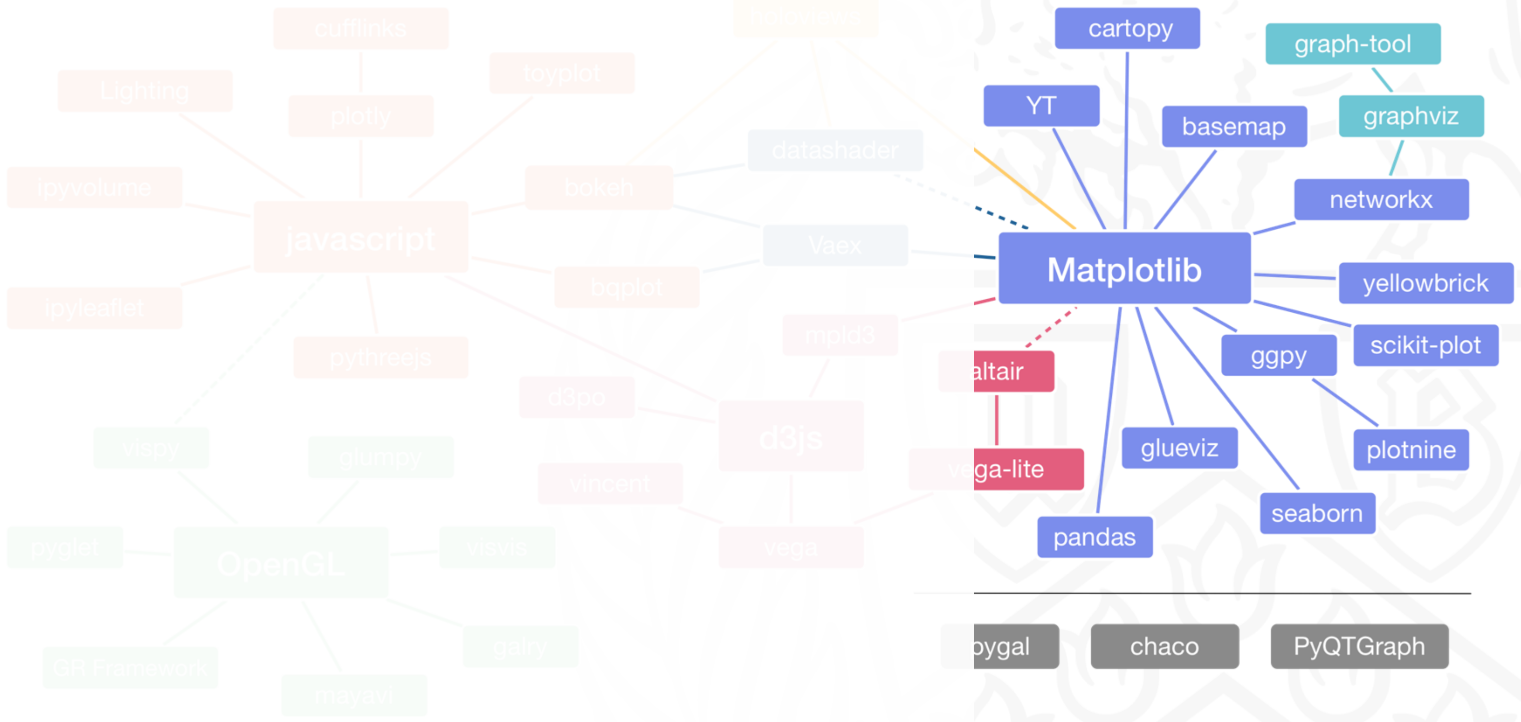
- It is a kind of language
- Every language has a grammar
- In this class, we will mostly use it for
  - Access Data
  - Data analysis
  - **Visualize**



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

Python

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



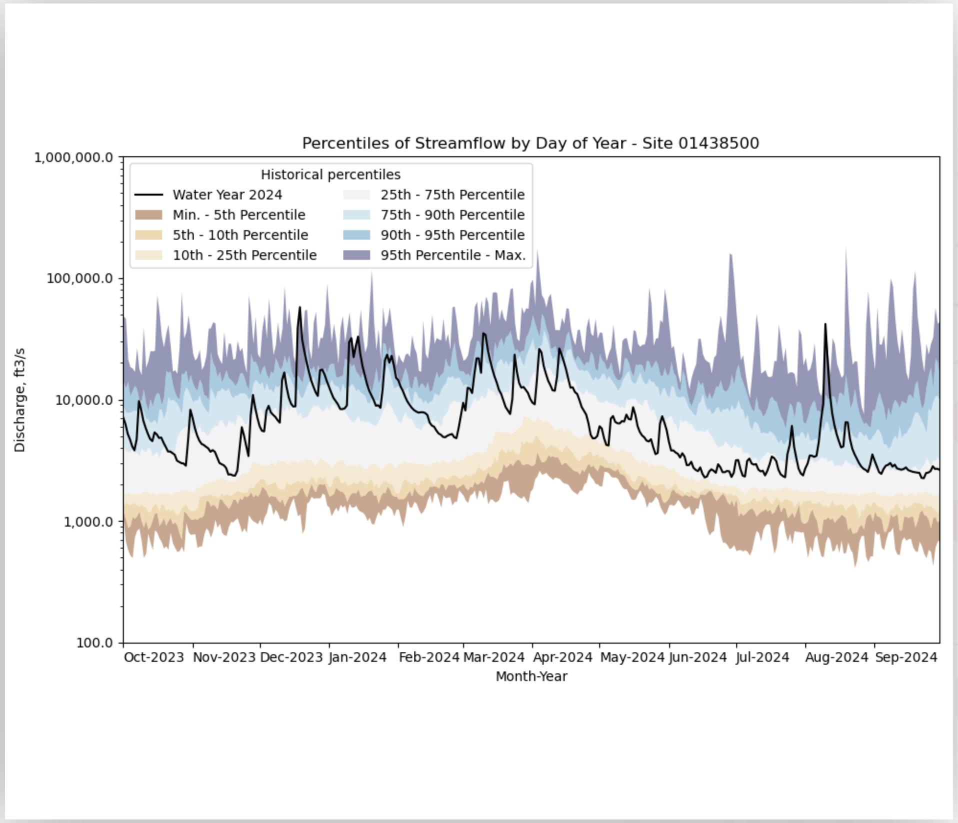
Source: https://pyviz.org/overviews/index.html

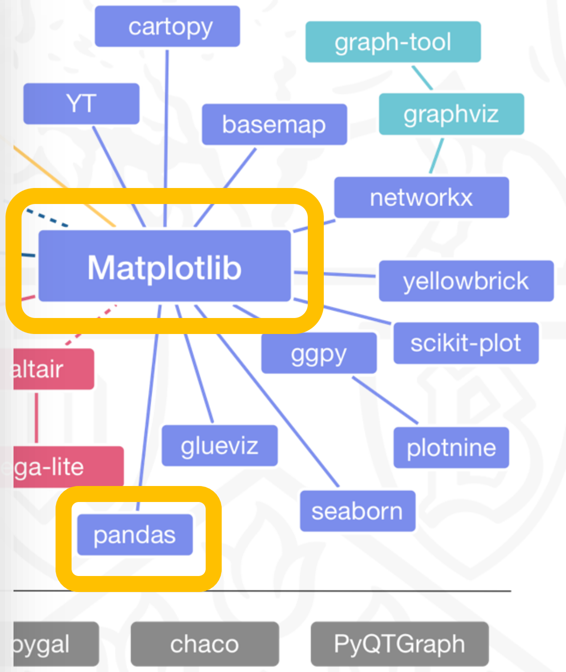
Python

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

Some examples – Time series plots

Daily streamflow percentiles from the 2024 water year compared to historical percentile values for the Delaware River at Montague, NJ.





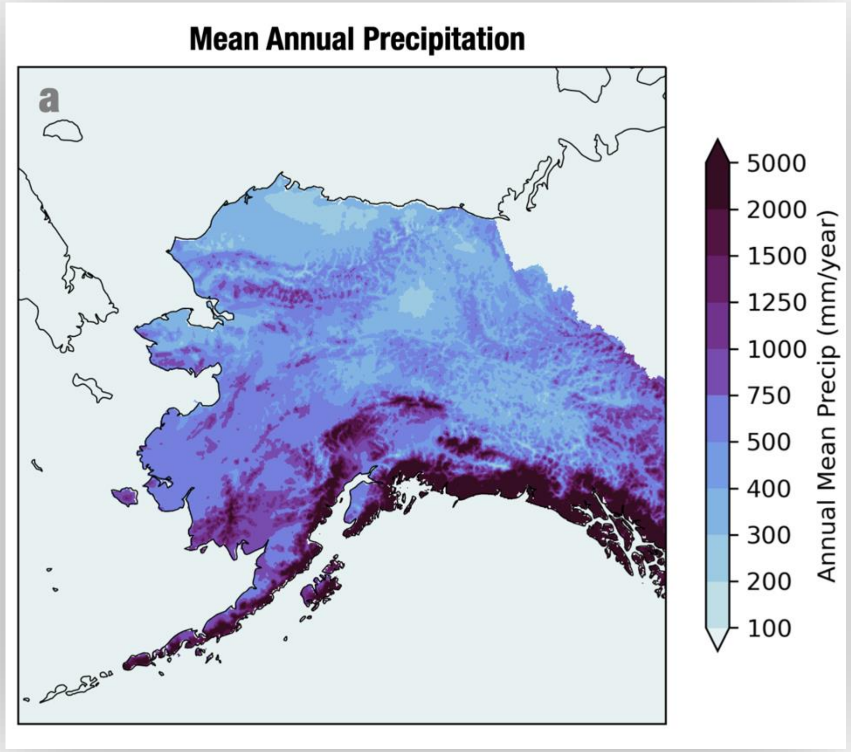
Source: https://waterdata.usgs.gov/blog/introducing-hyswap/

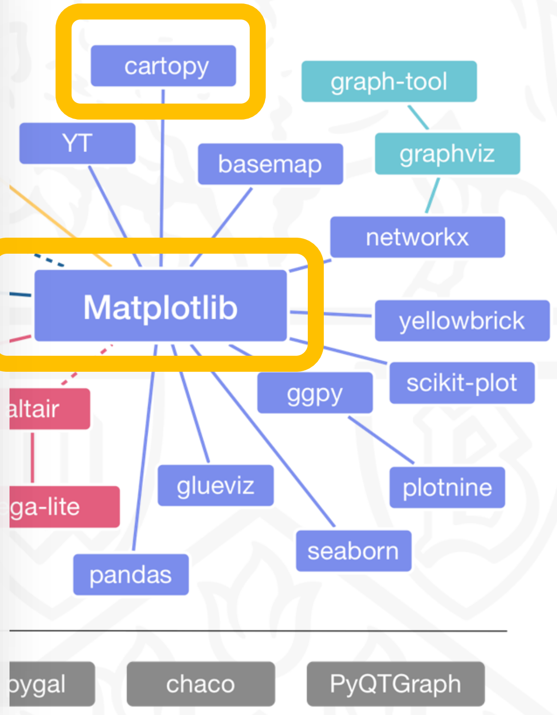
University at Buffalo The State University of New York

Python

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

Some examples – Gridded Dataset
Mean annual precipitation across Alaska and Yukon River Basin





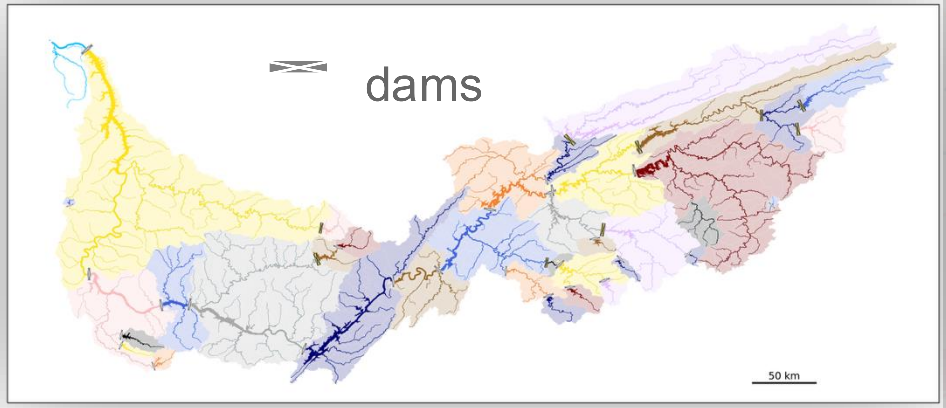
xarray

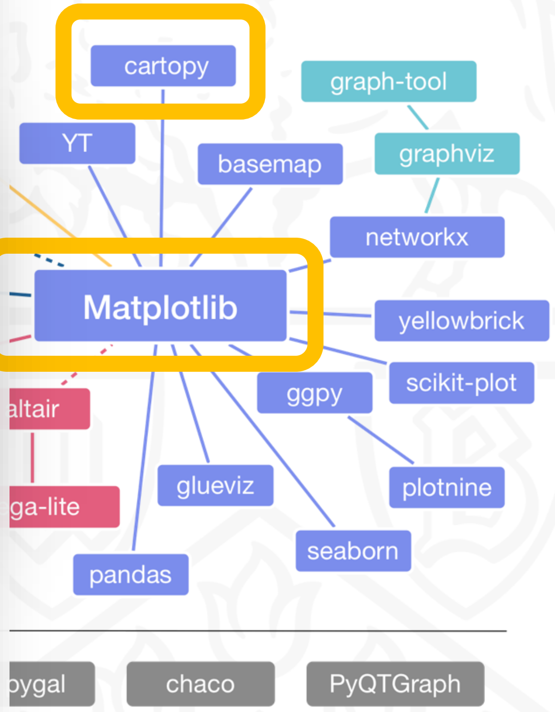
Source: https://pyviz.org/overviews/index.html

Python

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

Some examples – Vector-based Shapefiles
The fragmentation of river systems due to dam constructions





geopands

MS Powerpoint

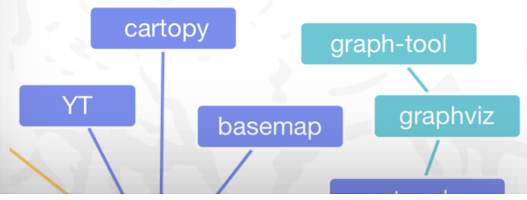
Source: https://pyviz.org/overviews/index.html

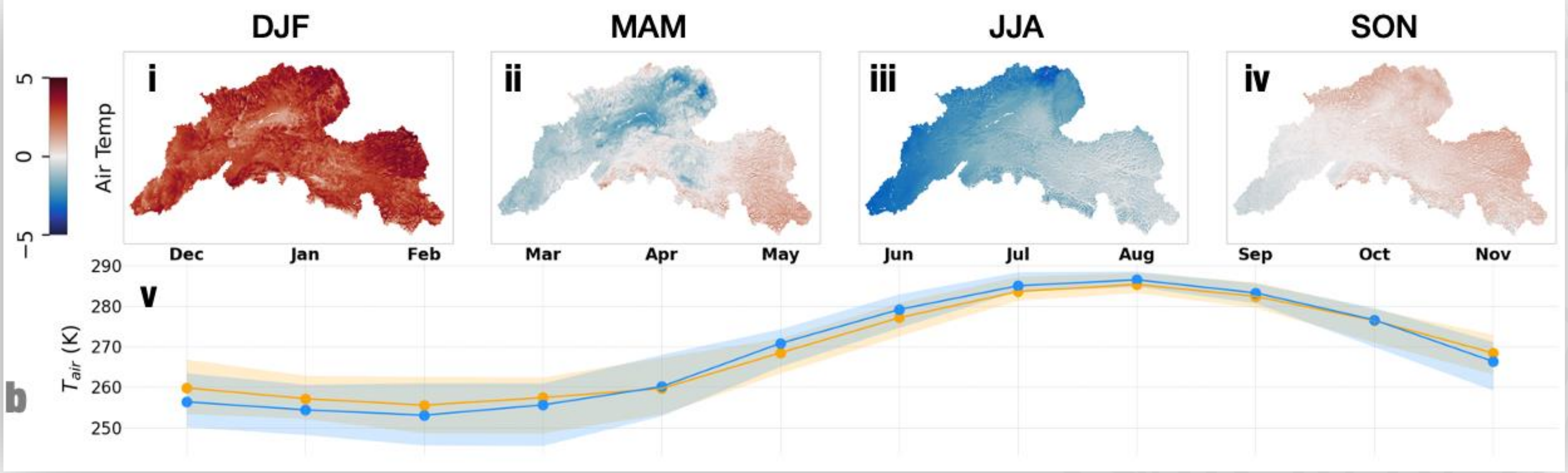
Combination of a variety of packages!!

Python

- It is a kind of language
- Every language has a grammar

Some examples:
Evaluating model simulation against observation
Seasonal differences (upper panels) and monthly time series (lower panel)





Python

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

Your creativity is the limit!

IN LAST WEEK’S LAB

You were asked to write a Python function

Python

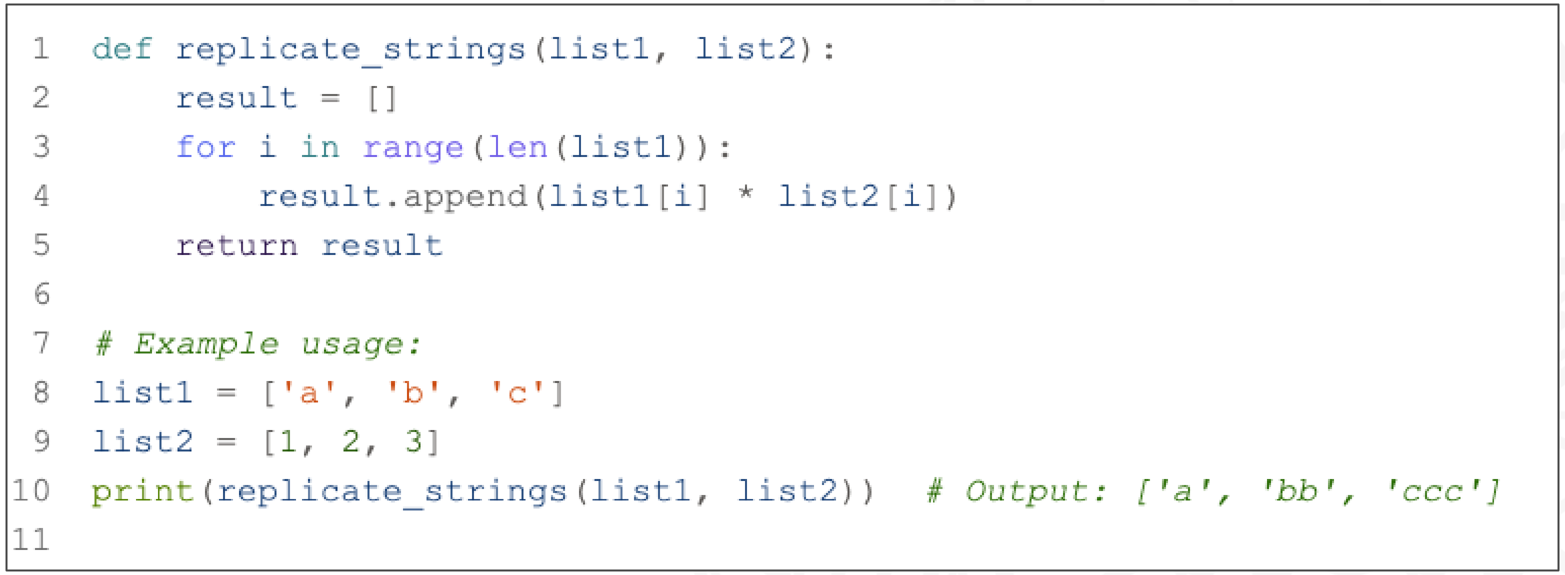
Practice 1.1

Please write a function `replicate_strings`

This function will duplicate the string X times.

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

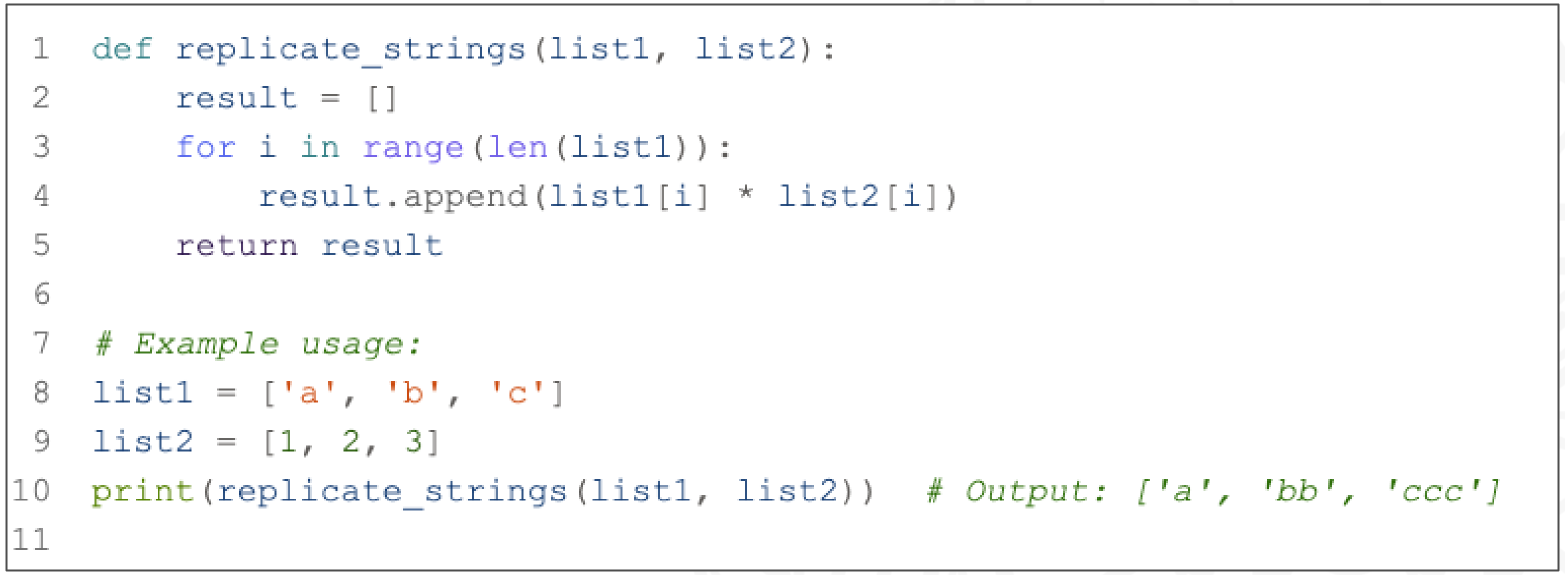
it will output a third string ['a', 'bb', 'ccc'].



```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



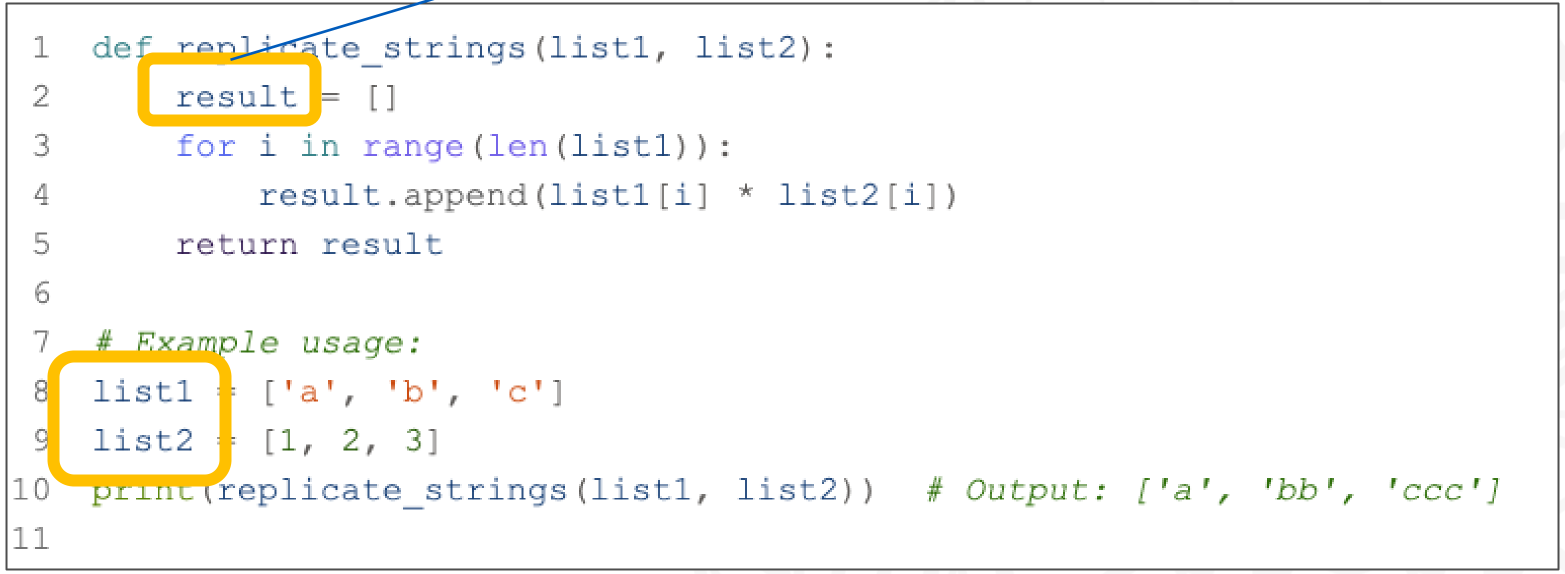
```python
def replicate_strings(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] * list2[i])
    return result

# 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

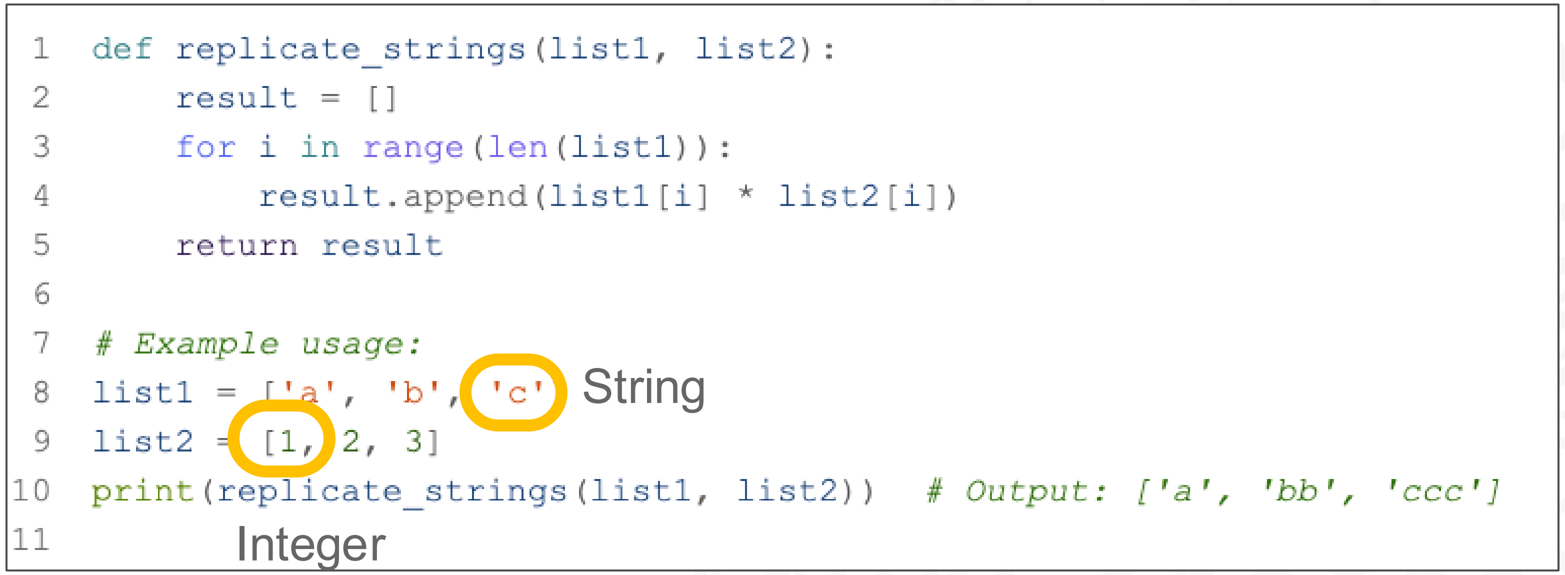
- Defined **inside** a function.
- Only exist **while the function runs**.
- **Not accessible** outside the function.



```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

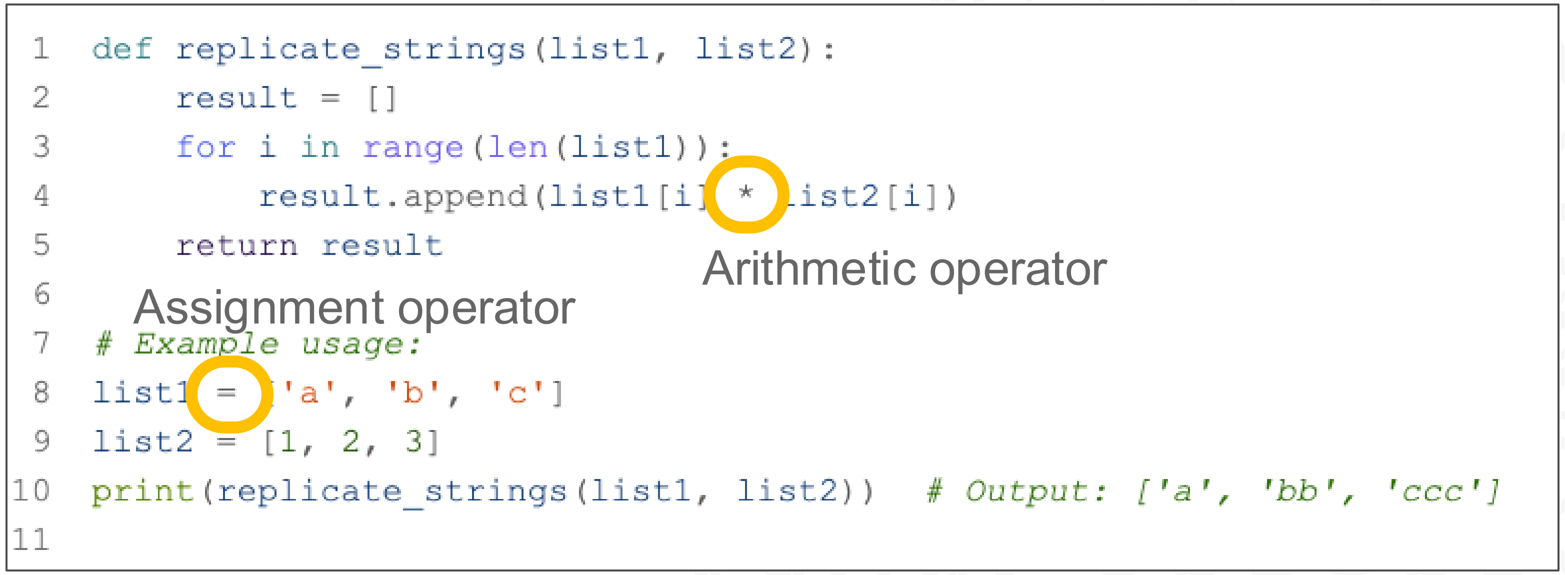
Type	Name	Example	Description
int	Integer	5, -3, 100	Whole numbers
float	Floating point	3.14, -0.5, 2.0	Decimal numbers
str	**String**	**"hello", 'abc'**	**Text data**
bool	Boolean	True, False	Logical values
list	**List**	**[1, 2, 3], ['a', 'b']**	**Ordered, changeable collection**
tuple	**Tuple**	**(1, 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



```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']
```

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

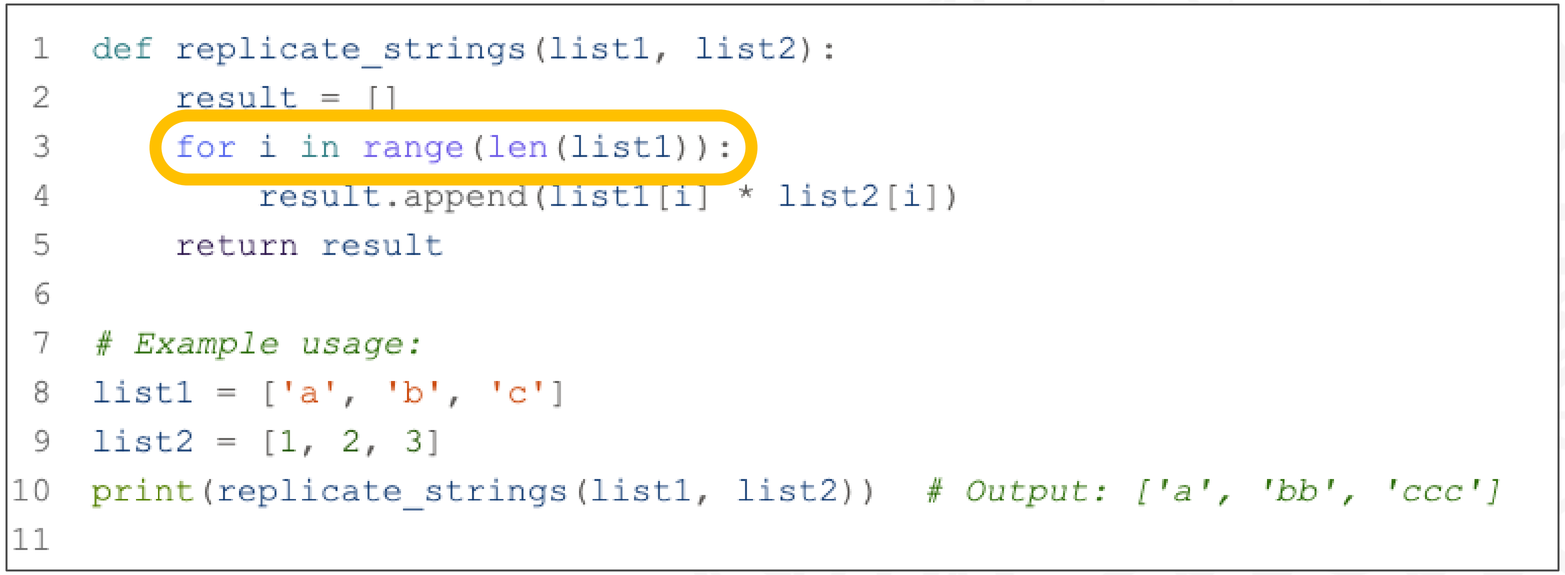
Practice

- 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 Structures
- Comments

Examples (for loop)

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

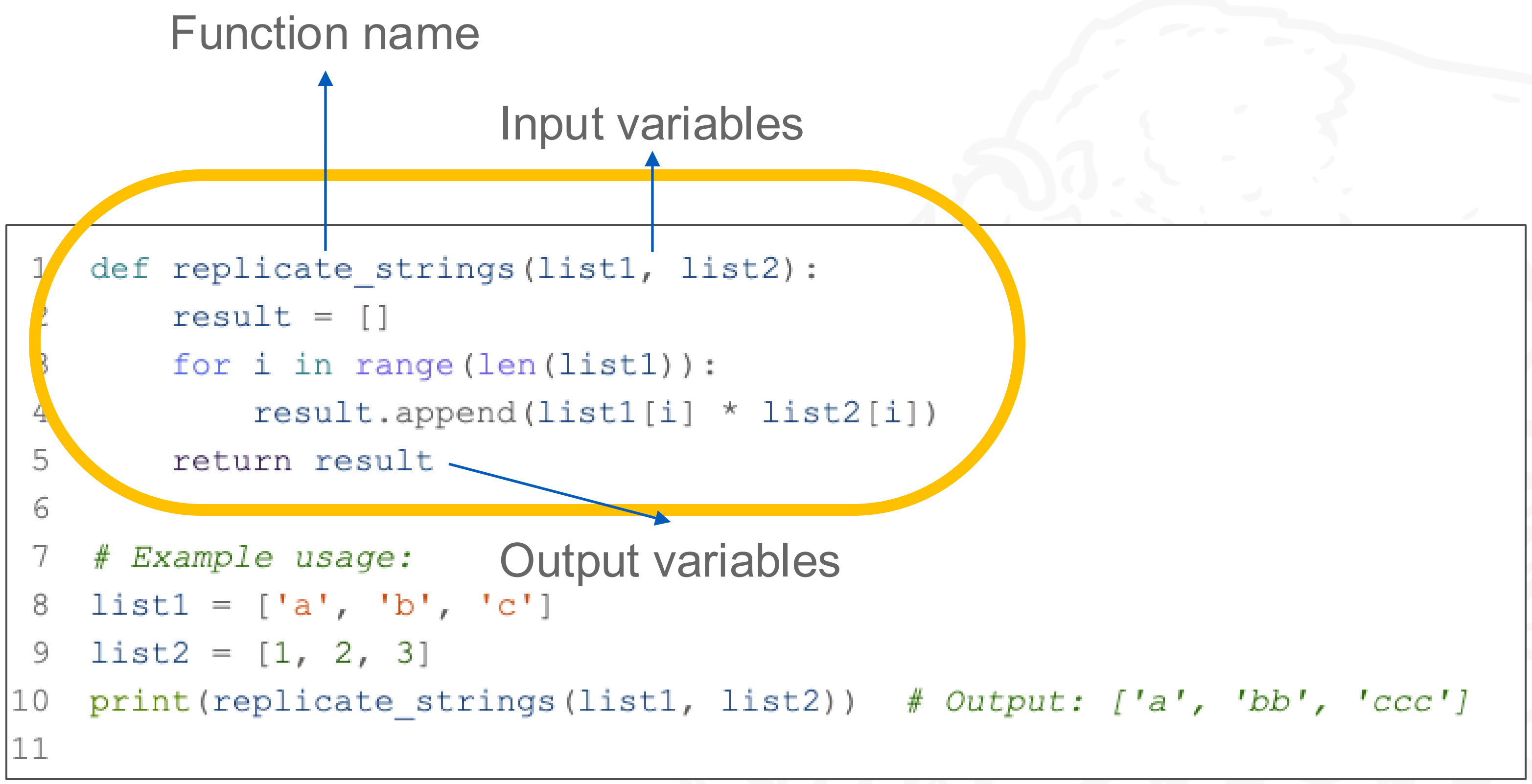
```
namelist = [‘Tom’,’Lisa’,’Jim’]
for v in namelist:
    print(i)
```

```
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
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

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

```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']

```

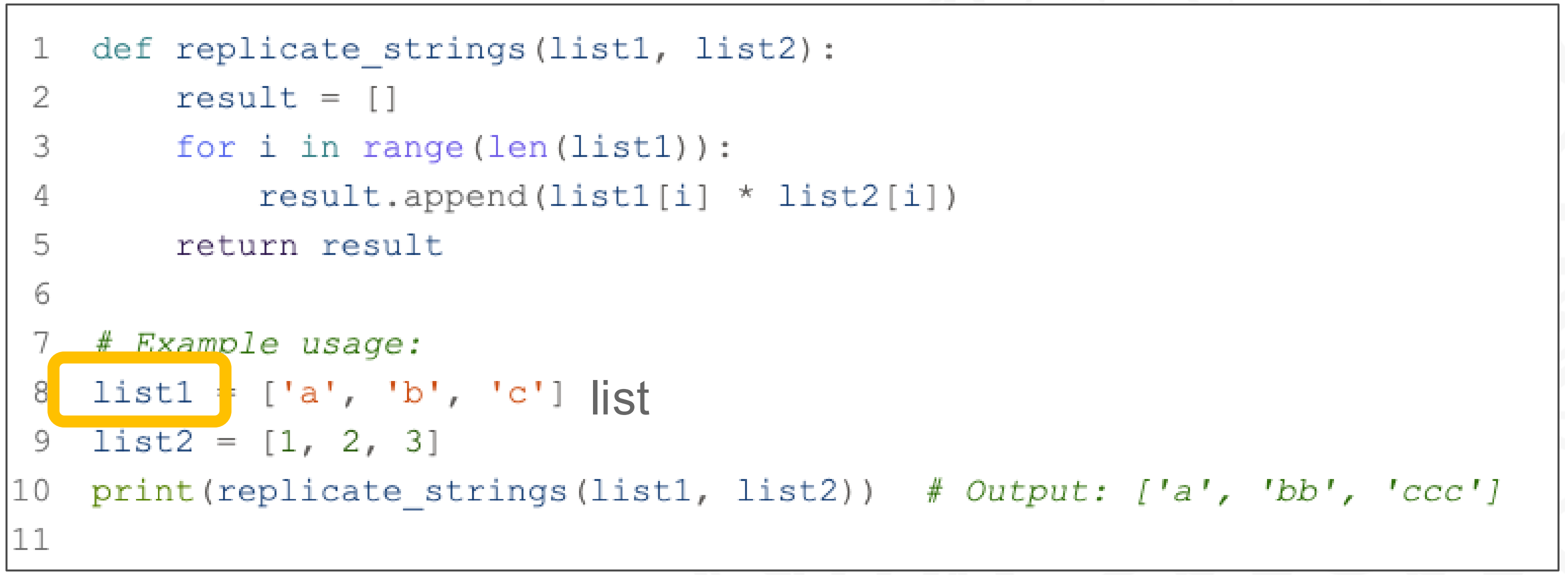
What to indent:

- 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



```
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']
```

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)

Practice

Write a Python program that:

- Stores a list of student names and their scores (provided).
- Uses a function to calculate the average score.
- Uses control flow to print:
    - Who passed (score ≥ 60)
    - Who failed
- Uses proper indentation and comments.

```
# Step 1: Define data structure
students = {
    "Alice": 85,
    "Bob": 58,
    "Charlie": 72,
    "Diana": 49
}
```