



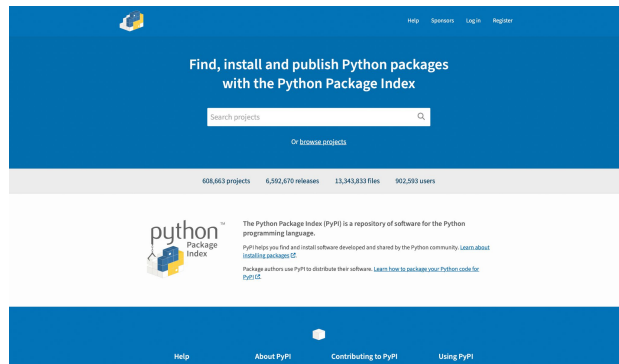
# Lab 4

# Python Software

COMS 2132 - Professor Bauer  
TA Session by Sophie Tsanang and Kavika Krishnan

# Introduction to Python Package Distribution

- Python Package Index (PyPI) is the official repository for Python packages, allowing developers to publish and distribute their software.
- Publishing packages to PyPI enables:
  - Modularity - Organizing code into reusable and shareable components
  - Open-Source Contribution - Making software accessible to the public and community
  - Professional Recognition - Demonstrating expertise and project





# Python Package Lifecycle – Step-by-Step

(end-to-end workflow when publishing a Python package)

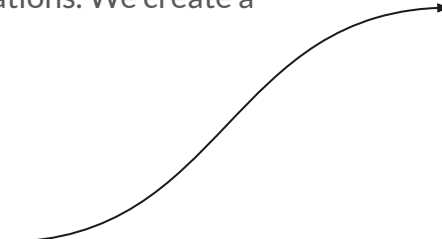
1. Write Python code and Create a Package
  - a. Develop/write your python module and package it with essential extra files:
    - i. README.md – A detailed description of the package.
    - ii. setup.py or pyproject.toml – Configuration files for installation.
    - iii. requirements.txt – Lists package dependencies.



## Example of Step 1

Let's say we want to create a package called mathlib, which provides basic math operations. We create a directory structure like this:

```
mathlib/  
├── src/  
│   ├── mathlib/  
│   │   ├── __init__.py  
│   │   └── operations.py  
├── pyproject.toml  
├── LICENSE  
└── README.md
```



- operations.py – Contains basic math functions.
- \_\_init\_\_.py – Marks it as a Python package.
- pyproject.toml – Defines metadata for PyPI.
- README.md – Explains how to use the package.

Inside operations.py:

```
def add(a, b):  
    return a + b
```

```
def subtract(a, b):  
    return a - b
```



## Python Package Lifecycle – Step-by-Step (cont.)

2. Choose a Version (Semantic Versioning)
  - Python packages use Semantic Versioning (Major.Minor.Patch):
    - Major – Breaking changes.
    - Minor – New features without breaking compatibility
    - Patch – Bug fixes and minor updates



## Example of Step 2

- Version 1.0.0 – First release with basic add() and subtract() functions
- Version 1.1.0 – Adds a multiply() functions
- Version 1.1.1 – Fixes a small bug in the add() function
- Version 2.0.0 – Major redesign (example: switching from functions to class)

**Can you guess the type of change? Major?  
Minor? Patch?**



## Example of Step 2

- Version 1.0.0 – First release with basic add() and subtract() functions
  - Type of change: Major because it's the first stable release
- Version 1.1.0 – Adds a multiply() functions
  - Type of change: Minor because it's adding a new function
- Version 1.1.1 – Fixes a small bug in the add() function
  - Type of change: Patch because it's fixing a bug
- Version 2.0.0 – Major redesign (example: switching from functions to class)
  - Type of change: Major because it's a redesign



## Python Package Lifecycle – Step-by-Step (cont.)

### 3. Build and Test the Package

- Ensure the package compiles correctly and passes all tests before distribution.

Example: Before publishing, we can test it using a simple Python script.

```
from mathlib.operations import add, subtract
```

```
print(add(5, 3)) # Should print 8
```

```
print(subtract(5, 3)) # Should print 2
```

If everything runs and works correctly, we proceed to packaging.





## Python Package Lifecycle – Step-by-Step (cont.)

### 4. Write and Update Documentation

- Clear documentation is crucial for user adoption and maintenance.

Example: Students should create a README.md file to describe how to use the package:

```
# MathLib
```

```
A simple math package for basic operations.
```

```
## Installation
```

```
pip install mathlib
```

```
## Usage
```

```
```python
```

```
from mathlib.operations import add
```

```
result = add(2, 3)
```

```
print(result) # Outputs: 5
```



## Python Package Lifecycle – Step-by-Step (cont.)

### 5. Publish to PyPI

- Upload your package to PyPI for public use.

Example: To publish the package:

```
python3 -m build
python3 -m twine upload --repository testpypi dist/*
```

Now students (or anyone) can install it using this link:

```
pip install --index-url https://test.pypi.org/simple/ mathlib
```



## Python Package Lifecycle – Step-by-Step (cont.)

### 6. Maintain and Update

- Keep improving the package by releasing updates, fixing bugs, and adding new features.

Example:

- Students can improve it by adding new functions (could be division, multiplication)
- Fixing bugs
- Adding features – If they add a new `square_root()` function, they release version 1.2.0.



# Software License

Why is a Software License Important?

- A software license is a legal agreement that defines how others can use modify, and share your code
- Without a license, no one can legally use your code, even if it's publicly available
- Adding a license allows you to control how your work is used while ensuring compliance with your intentions.



# Common Software Licenses

A permissive license means:

- You **can do almost anything with the code**—use it, modify it, share it, and even sell it.
- You **don't have to share your changes**—you can keep them private.
- The **only rule** is that you must give credit to the original creator (keep the copyright notice).

License	Type	Main Feature
MIT License	Permissive	<ul style="list-style-type: none"><li>- Allows users to do almost anything they want with your project under the condition that the copyright notice and license is included in the licensed material.</li><li>- Under this license, users can distribute and modify the material for both personal and commercial use.</li></ul>
GPL (General Public License)	Copyleft (if someone uses your code, they <b>MUST</b> share their changes with the public.)	<ul style="list-style-type: none"><li>- The material should remain free to use.</li><li>- This license requires that all the work done using this material is distributed under the same license (GPL) as well. Otherwise, the use and distribution terms are similar to MIT License.</li></ul>
Apache License	Permissive	<ul style="list-style-type: none"><li>- Allows users to use the material for any purpose as long as they include the copyright notice and license in the licensed material.</li><li>- It is very similar to MIT License with contributors additionally providing an express grant of patent rights.</li></ul>



# How to Choose the Right Software License

**1. Do you want your software to be open for anyone to use and modify, even for closed-source projects?**

- YES → Use a permissive license like MIT or Apache. (Companies and individuals can use your code however they want.)
- NO → Use a copyleft license like GPL. (Anyone who modifies your code must share their changes.)

**2. If using a copyleft license, do you want your code to be used ONLY in open-source projects?**

- YES → Use a strong copyleft license like GPL. (Any project using your code must also be open-source.)
- NO → Use a weak copyleft license like LGPL. (Companies can still use your code in their proprietary software, but modifications to your code must be open-source.)

**3. What license do other projects use?**

- If most of your dependencies use GPL, you may have to use GPL too.
- If you're contributing to an existing open-source project, check what license they use before choosing yours.



## Lab Exercise – Creating a String Manipulation Package

In this lab, we will create a Python package called **stringutils** that provides basic string manipulation functions

- Write Python functions for string operations.
  - Some operations could be converts a string to lowercase, uppercase, and etc
- Organize the project into a proper package structure.
- Create a README.md and pyproject.toml file.
- Build and publish the package to **Test PyPI**.
- Install and test your package.



# Data Structures Review - Arrays

1D Arrays:

```
arr1d = [1, 2, 3, 4, 5, 6]
```

What does `arr1d[0]` equal?

Multidimensional Arrays:

- Using numpy library - can even convert images to 2d arrays of pixels!

```
import numpy as np
```

```
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
```

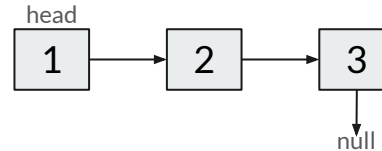
How can I use indexing to access the number 5 in this array?



# Data Structures Review - Linked Lists

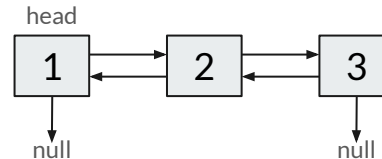
## Singly Linked List:

- each node contains data as well as the memory address of the next node in the list
- linked using pointers



## Doubly Linked List:

- same as singly linked list, except each node contains the memory address of the next node and the previous node



```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

```
class LinkedList:
    def __init__(self):
        self.head = None
```

```
def append(self, data):
    new_node = Node(data)
    if self.head is None:
        self.head = new_node
    return
    last_node = self.head
    while last_node.next:
        last_node = last_node.next
    last_node.next = new_node
```

## Data Structures Review - Stacks

- LIFO: Last In First Out
- Insertion “push” and deletion “pop” occur on the same end of the stack
- Can use an array for a simple implementation in Python:

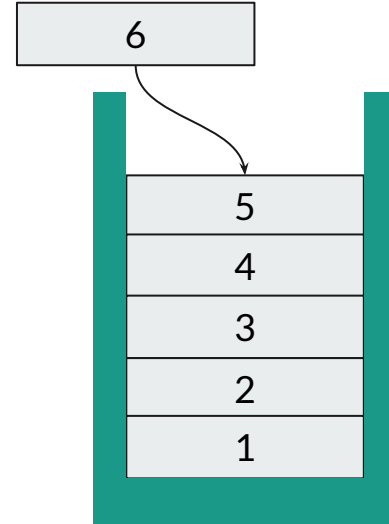
```
stack = []
```

```
stack.append(1)
```

```
stack.append(2)
```

```
stack.append(3)
```

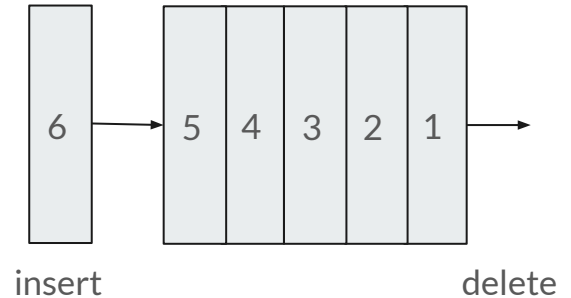
```
print(stack.pop()) # What will this print?
```



## Data Structures Review - Queues

- FIFO: First In First Out
- Insertion and deletion occur on opposite ends of the queue
- Can use an array for a simple implementation in Python:

```
queue = []  
queue.append(1)  
queue.append(2)  
queue.append(3)  
print(queue.pop(0)) # What will this print?
```





# Abstract Data Types (ADTs) vs. Data Structures

- An ADT is a concept that defines what operations can be performed but not how they are implemented. (Think of it as a blueprint for how data should behave.)
- A data structure is a specific way of organizing and storing data in memory.

Concept	What it Defines	Examples
ADT	What operations can be performed (concept)	Stack, Queue, List
Data Structures	How the data is actually stored and implemented	Array, Hash Table



## Queue using two Stacks

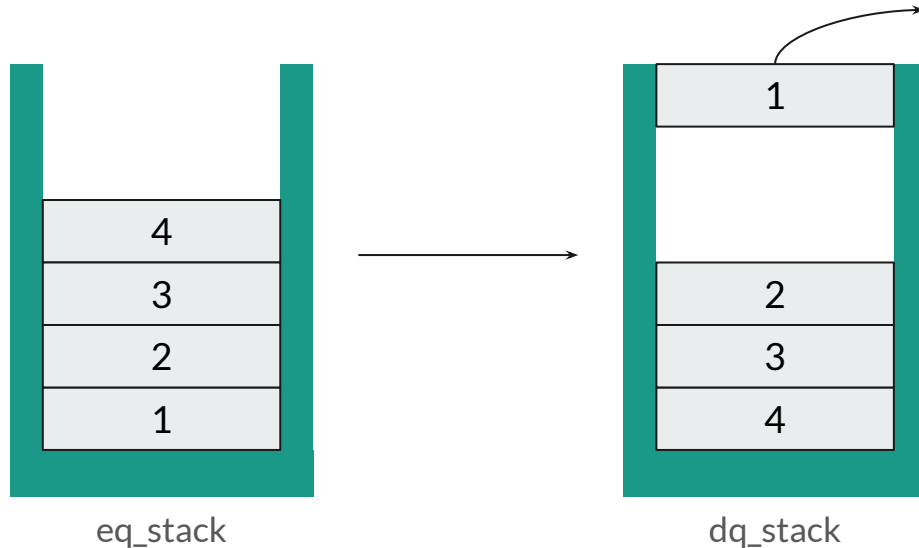
How could you implement a queue (FIFO) using two stacks (LIFO)?

elements  
= [1,2,3,4]



## Queue using two Stacks

How could you implement a queue (FIFO) using two stacks (LIFO)?





# Attendance

