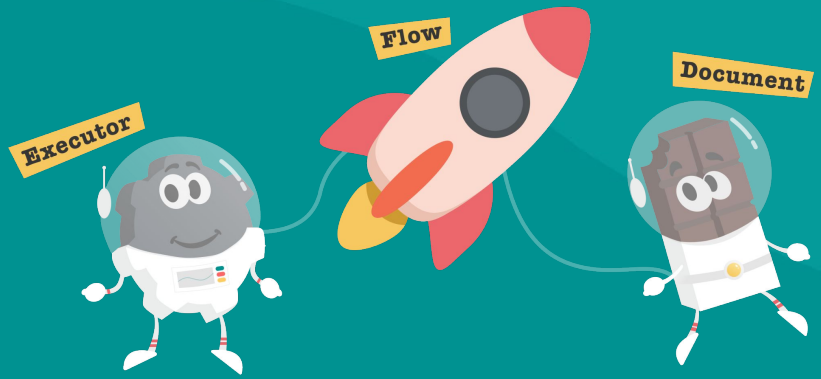


Another introduction to Python Import

A gentle introduction to import

Nan Wang, 2022.5.28 Python Meetup Shanghai

王 楠

- 2020-now, Co-founder & CTO, Jina AI
- 2017-20, Senior Researcher, Tencent
- 2015-17, Data scientist, Zalando, Germany
- 2009-14, Ph. D., Ruhr-Universität Bochum, Germany

Forbes

FORBES AI30 DACH 2020

Diese 30 Start-ups im Bereich Künstliche Intelligenz (KI) haben in der DACH-Region im vergangenen Jahr besonders auf sich aufmerksam gemacht.

TC

Join TechCrunch+

Login

Search Q

Startups

TechCrunch+

Audio

Newsletters

Videos

Advertise

Events

More

Jina.ai raises $30M for its for its neural search platform

Frederic Lardinois @frederici / 7:00 PM GMT+8 • November 22, 2021

Comment

Berlin-based Jina.ai, an open source startup that uses neural search to help its users find information in their unstructured data (including videos and images), today announced that it has raised a $30 million Series A funding round led by Canaan Partners. New investor Mango Capital, as well as existing investors GGV Capital, SAPIO and Yunqi Partners also participated in this round, which brings the company's total funding to $39 million to date.

Jina.ai CEO and co-founder Han Xiao, who co-founded the company together with Nan Wang and Bing He, explained that the idea behind neural search is to use deep learning neural networks to go beyond traditional keyword-based search tools. Making use of relatively new machine learning technologies like transfer learning and representation learning, the company's core Jina framework can help developers quickly build search tools for their specific use cases.

"Given an image, audio, video or whatever — we first use deep neural networks to translate this data format into a universal representation," Xiao explained. "In this case, it's mostly a mathematic vector — 100-dimensional vectors. And then, the matching [algorithm] does not count how many letters match but counts the mathematical distance, the vector distance between these two vectors. In this way, you can basically use this kind of methodology to solve all kinds of data search problems or relevance problems."

Xiao described Jina as akin to TensorFlow for search (with TensorFlow being Google's open source machine learning framework). Just like TensorFlow or PyTorch defined the design pattern of how people design AI systems, Jina wants to define how people build neural search systems — and become the de facto standard for doing so in the process.

Jina AI Raises $30 Million to Build Neural Search

By Ben Schwartz | *Nov 30, 2021*

CHANNEL: Information Management

PHOTO: CLINT ADAIR ON UNSPLASH

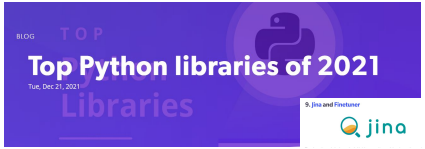
Neural search company Jina AI announced $30 million in series A financing this month. Founded in 2020, the Berlin-based company has now raised a total of $39 million. The latest funding round was led by Westport, Conn.-based venture capital firm Canaan Partners.

The idea behind "neural search" is to enable businesses to build search solutions that generate insights from unstructured data that lead to more effective business decisions. Jina AI's core project, Jina, is an open-source program being built on GitHub which users can utilize to create their own cloud-native neural search solution. Jina AI said this can be done in a matter of hours and matches businesses' need for a lightweight development cycle.

PUBLIC NOISES

Mattermost

Open Source Projects Contribute to in 2022

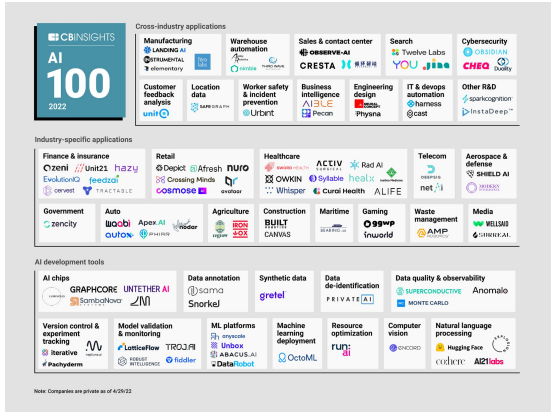


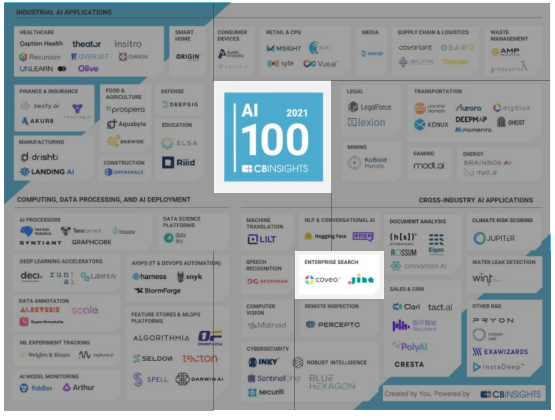
ZDNet

Open–source growth and venture capital investment: Data, databases, challenges, and opportunities

Then we have Jina AI, a neural search ecosystem for businesses and developers. In terms of databases and data-related platforms,...

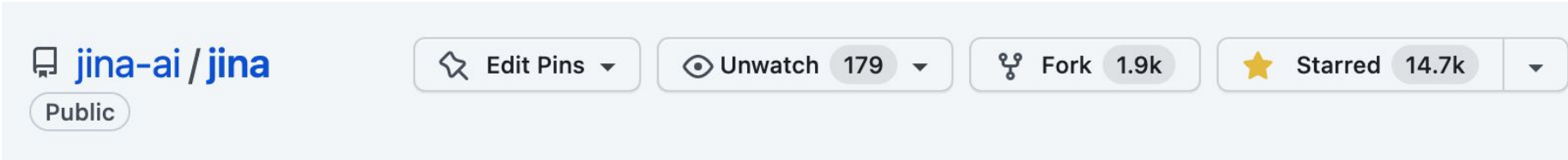
7 Jul 2021





introduction

Jina



Jina: Cloud-native neural search framework for ***any*** kind of data

- **Top 1** on GitHub Trending [1]
- **Top 10** Python Libraries 2021 [2]

1. 2021.8.21, 2021.10.24, 2021.12.24
2. https://tryolabs.com/blog/2021/12/21/top-python-libraries-2021

CONTENT

Demos

01

https://github.com/nan-wang/python-meetup-intro-import-2022.git

Demo 1: module importing

```
demo1
├── main.py
└── module_foo.py
```

Demo 2: intra-package importing

```
demo2
├── main.py
└── pkg_foo
    ├── __init__.py
    ├── module_bar.py
    └── module_foo.py
```

$ python main.py

```
1 import pkg_foo.module_foo
2 Traceback (most recent call last):
3   File "main.py", line 1, in <module>
4     from pkg_foo.module_foo import print_foo
5   File "/Users/nw/demo2/pkg_foo/module_foo.py", line 4,
  in <module>
6     from module_bar import bar_var
7 ModuleNotFoundError: No module named 'module_bar'
8
```

Demo 2: intra-package importing

```
demo2
├── main.py
└── pkg_foo
    ├── __init__.py
    ├── module_bar.py
    └── module_foo.py
```

demo2/pkg_foo/module_foo.py

```
1 -from module_bar import bar_var
2 +from .module_bar import bar_var
```

Demo 3: package importing

```
demo3
├── pkg_foo
│   ├── __init__.py
│   ├── module_bar.py
│   └── module_foo.py
└── scripts
    └── main.py
```

$ python scripts/main.py

```
Traceback (most recent call last):
  File "scripts/main.py", line 1, in <module>
    from pkg_foo.module_foo import print_foo
ModuleNotFoundError: No module named 'pkg_foo'
```

Demo 3: package importing

```
demo3
├── pkg_foo
│   ├── __init__.py
│   ├── module_bar.py
│   └── module_foo.py
└── scripts
    └── main.py
```

$ python scripts/main.py

```
Traceback (most recent call last):
  File "scripts/main.py", line 1, in <module>
    from ..pkg_foo.module_foo import print_foo
ImportError: attempted relative import with no known
parent package
```

Demo 3: package importing

```
demo3
├── pkg_foo
│   ├── __init__.py
│   ├── module_bar.py
│   └── module_foo.py
└── scripts
    └── main.py
```

scripts/main.py

```
-from ..pkg_foo.module_foo import print_foo
+import sys
+sys.path.append('/Users/nanwang/Codes/jina-ai/python-meetup-hangzhou-2022/demo3')
+
+from pkg_foo.module_foo import print_foo
```

Why do we fix in different ways?

demo2/pkg_foo/module_foo.py

```
1 -from module_bar import bar_var
2 +from .module_bar import bar_var
```

scripts/main.py

```
1 -from ..pkg_foo.module_foo import print_foo
2 +import sys
3 +sys.path.append('/Users/nanwang/Codes/jina-ai/python-
  meetup-hangzhou-2022/demo3')
4 +
5 +from pkg_foo.module_foo import print_foo
```

Module

02

Let’s take a second look at Module

A module is a file containing Python definitions and statements.

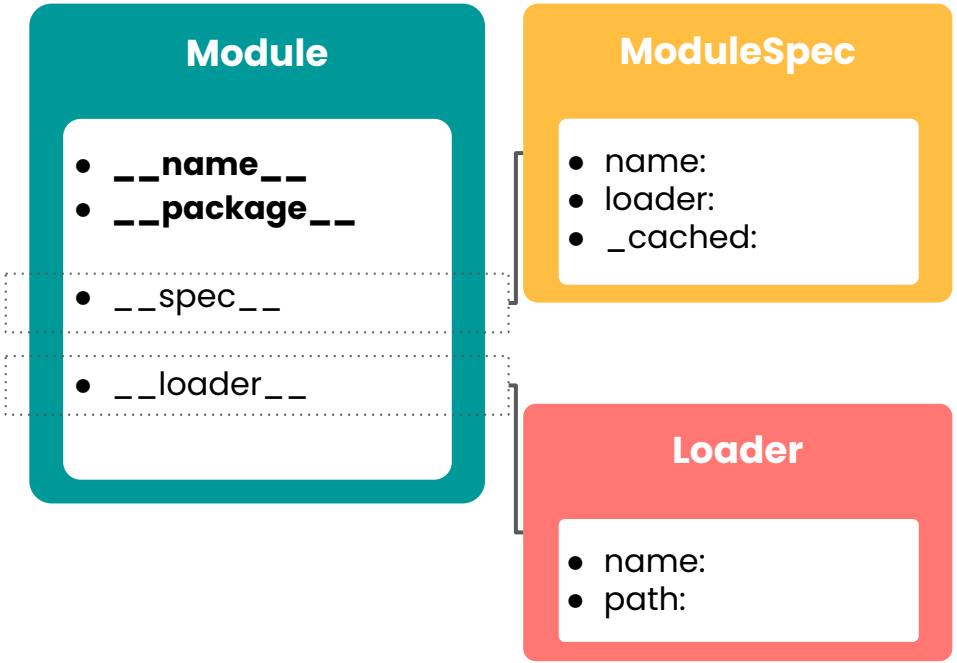
You can think of packages as the directories on a file system and modules as files within directories

*…, but **don’t take this analogy too literally** since packages and modules need not originate from the file system.*

— https://docs.python.org/3/reference/import.html

Module, ModuleSpec &Loader

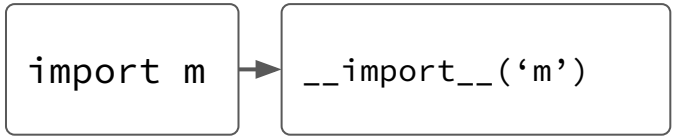
- **Module**: An object that serves as an organizational unit of Python code.
- **Module spec**: An encapsulation of the module's import-related information
- **Loader**: An object that loads a module.



Behind import

02

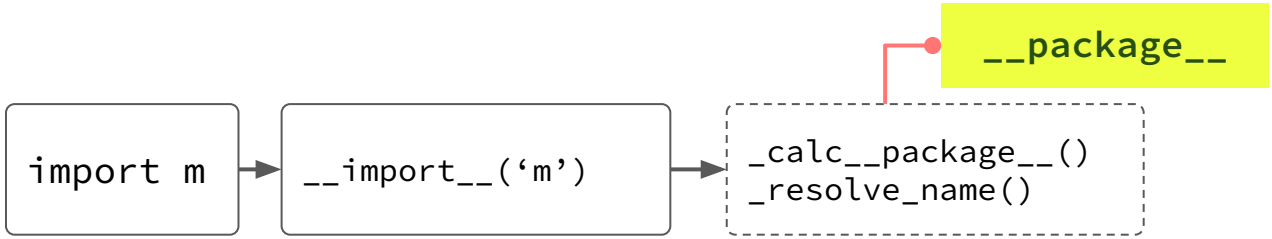
Import module



importlib/_bootstrap.py

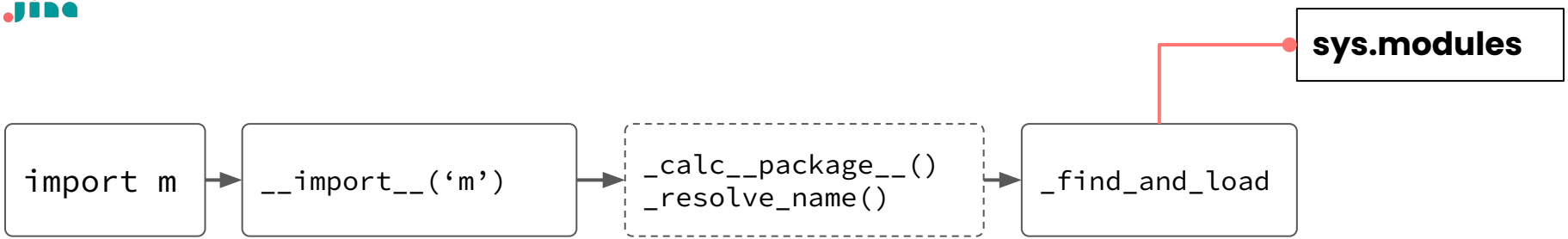
```
def __import__(name, globals=None, locals=None,
fromlist=(), level=0):
    ...
```

import m	__import__('m', globals(), locals(), None, 0)
from foo_pkg import m	__import__('pkg', globals(), locals(), ['m'], 0)
from m import bar	__import__('m', globals(), locals(), ['bar'], 0)
from ..m import bar	__import__('m', globals(), locals(), ['bar'], 2)
from m import *	__import__('m', globals(), locals(), ['*'], 0)
import foo_pkg.m	__import__('foo_pkg.m', globals(), locals(), None, 0)



Optional:

Calculate current module for relative imports.

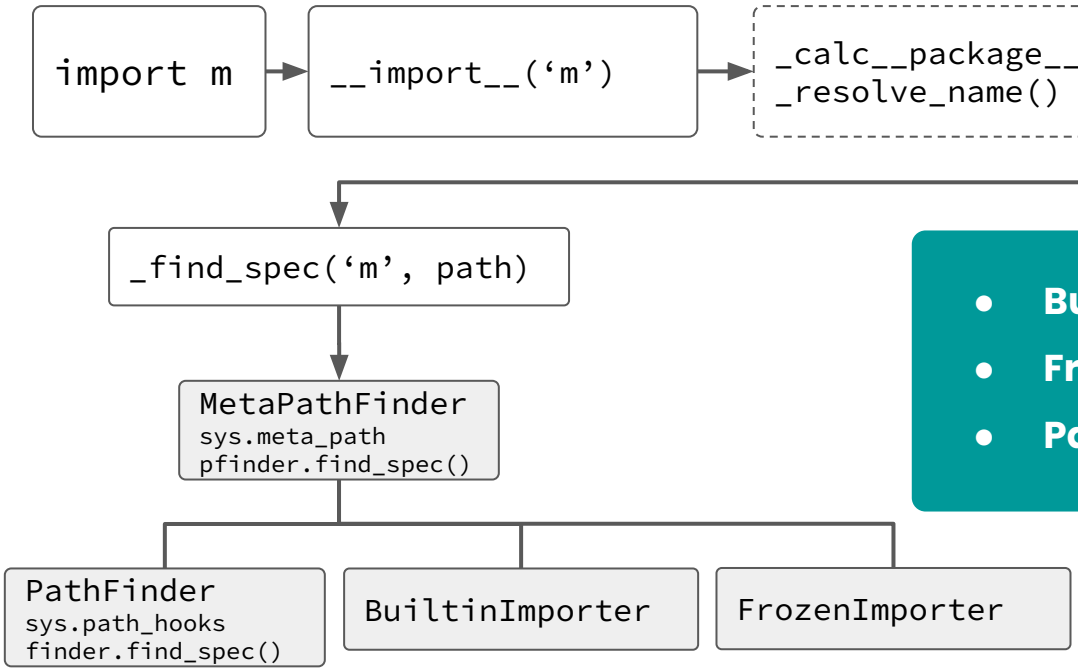


Finder:

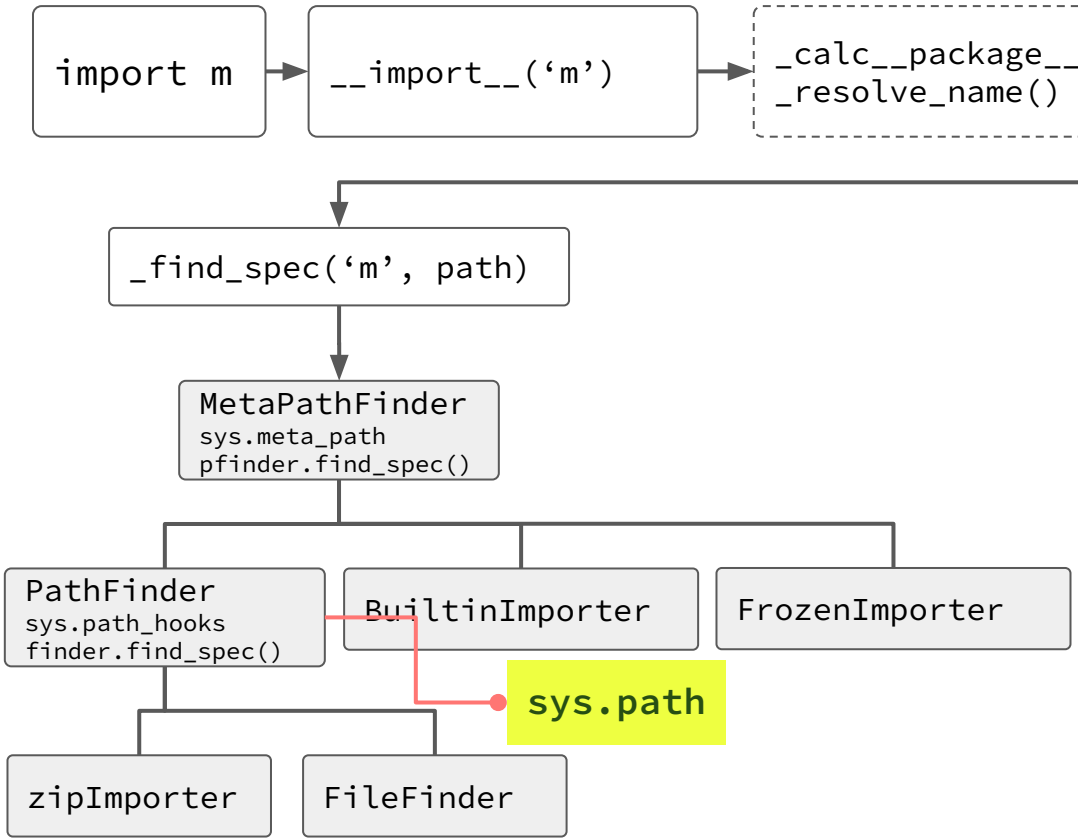
An object that tries to find the loader for a module that is being imported.

Loader:

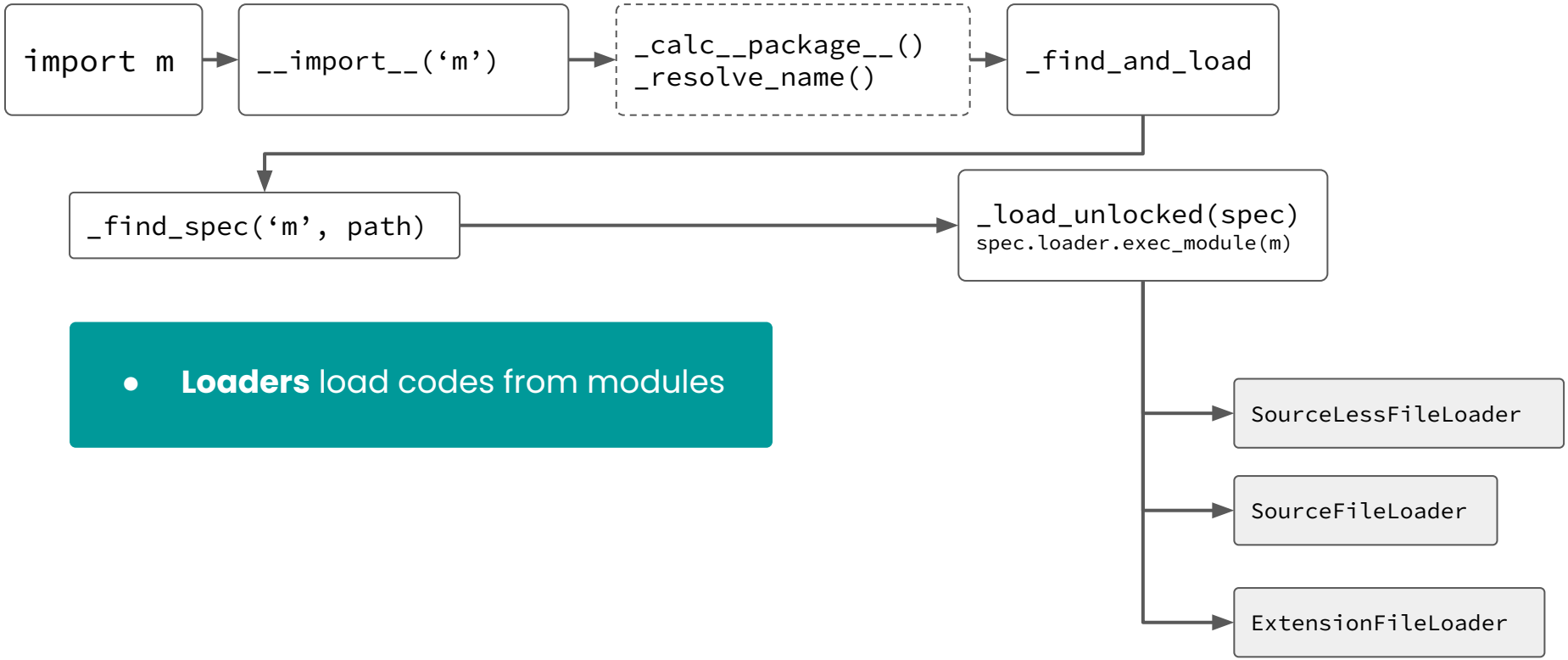
An object that loads a module



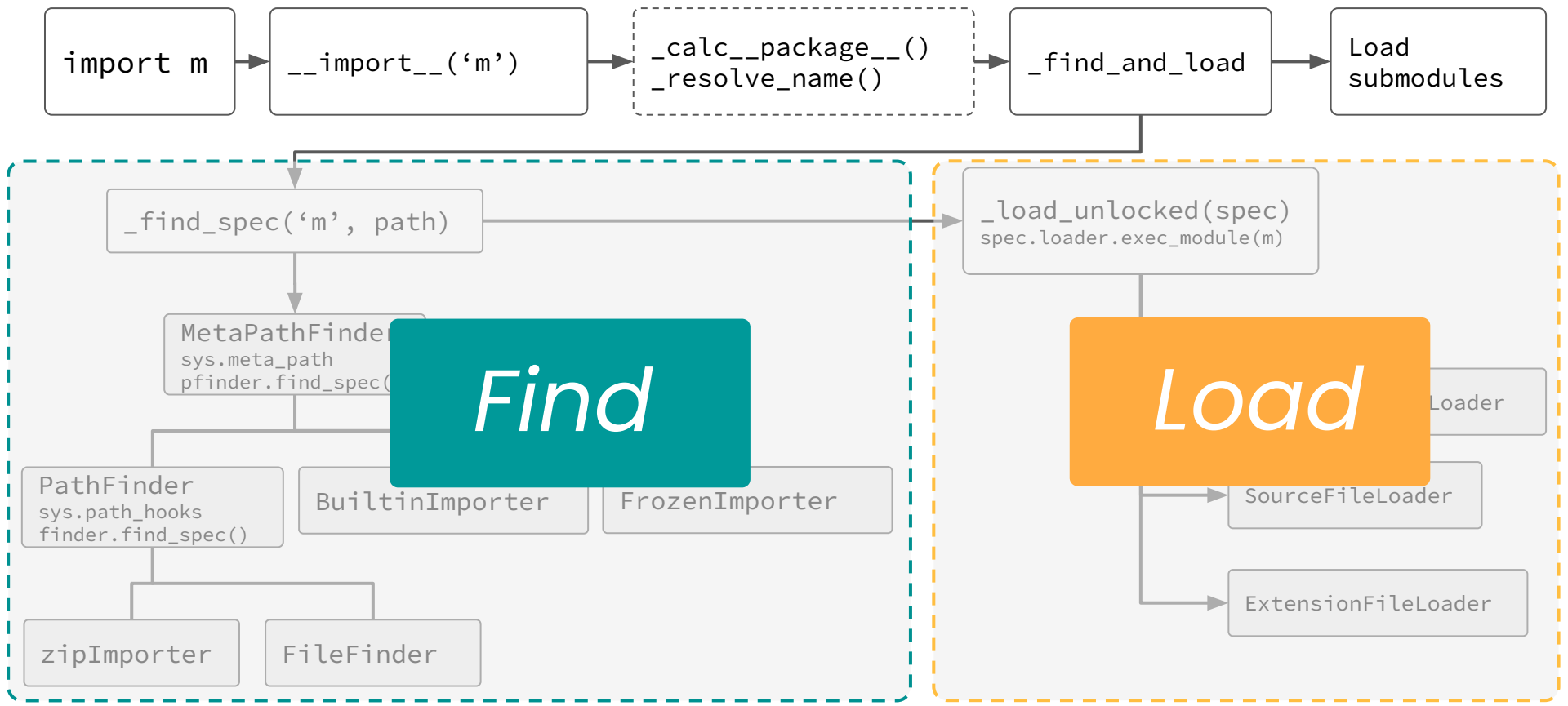
- **BuitinImporter**: c modules compiled
- **FrozenImporter**: Python modules compile
- **PathFinder**: others



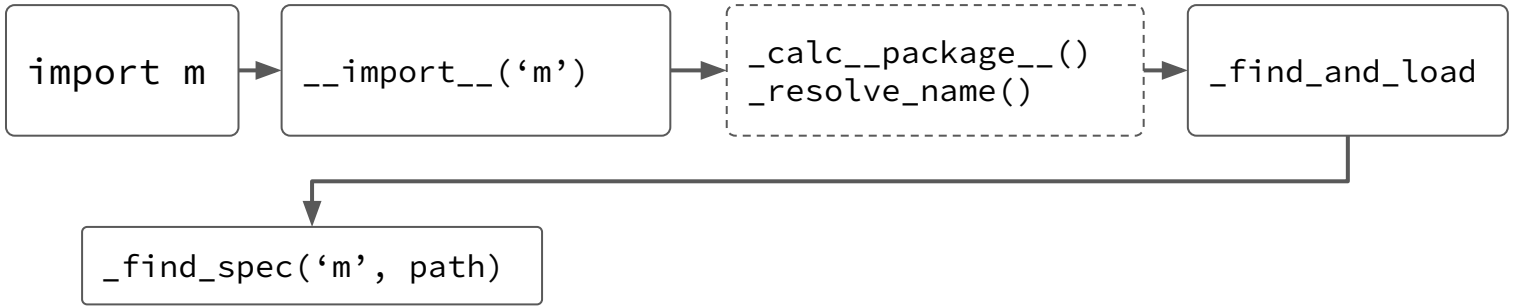
- **PathFinder** doesn't do search
- **Path Entry Finder** knows how to find modules
  - zipImporter
  - FileFinder



- **Loaders** load codes from modules


import m
__import__('m')
_calc__package__()
_resolve_name()
_find_and_load
Load submodules
_find_spec('m', path)
MetaPathFinder
sys.meta_path
pfinder.find_spec(
Find
PathFinder
sys.path_hooks
finder.find_spec()
BuiltinImporter
FrozenImporter
zipImporter
FileFinder
_load_unlocked(spec)
spec.loader.exec_module(m)
Load
Loader
SourceFileLoader
ExtensionFileLoader

Let’s go back to our confusing demos

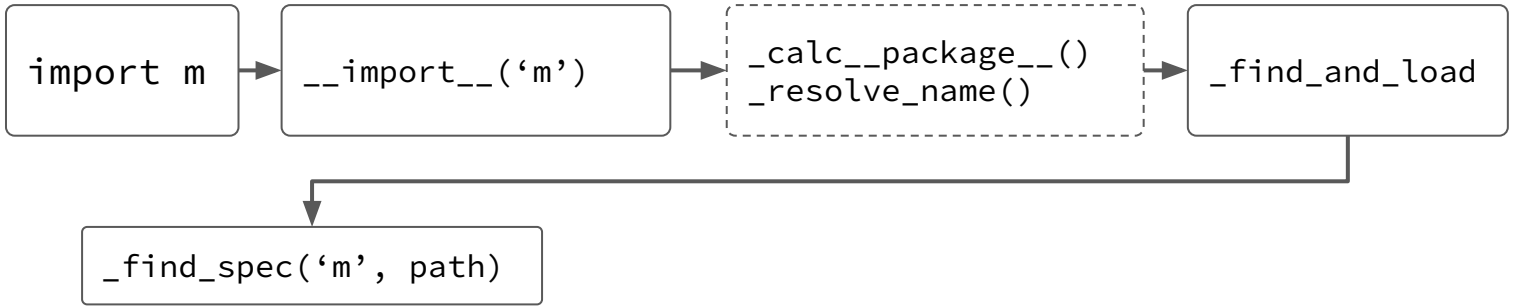


pkg_foo/module_foo.py

```
1 from module_bar import bar_var
2 ...
```

$ python main.py

```
1 import pkg_foo.module_foo
2 Traceback (most recent call last):
3   File "main.py", line 1, in <module>
4     from pkg_foo.module_foo import print_foo
5   File "/Users/nw/demo2/pkg_foo/module_foo.py", line 4,
  in <module>
6     from module_bar import bar_var
7 ModuleNotFoundError: No module named 'module_bar'
8
```



demo2/pkg_foo/module_foo.py

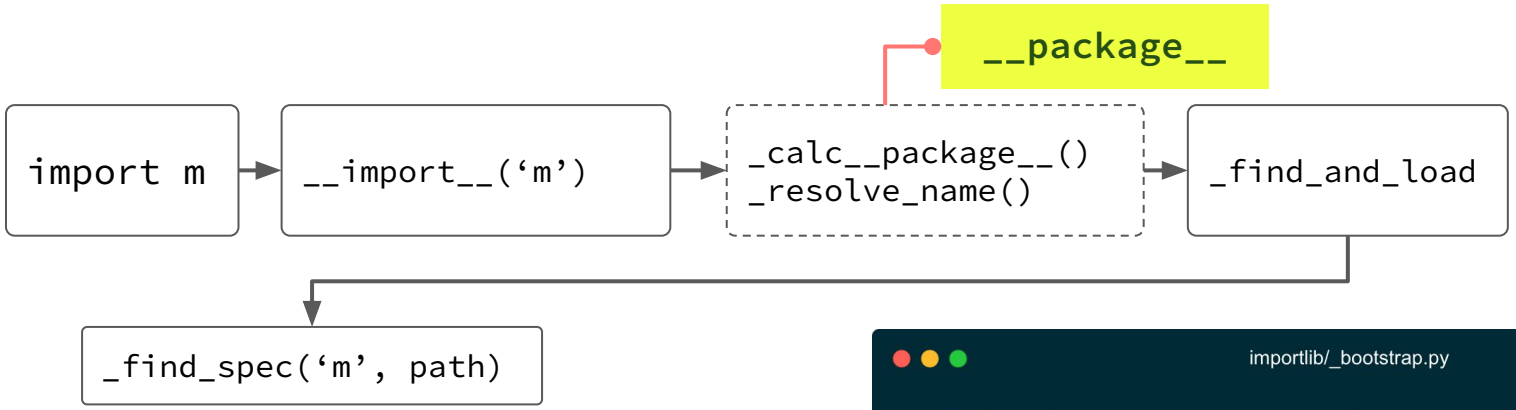
```
1 -from module_bar import bar_var
2 +from .module_bar import bar_var
```

```
from m import b_v
```

```
__import__('m', globals(), locals(), ['b_v'], level=0)
```

```
from .m import b_v
```

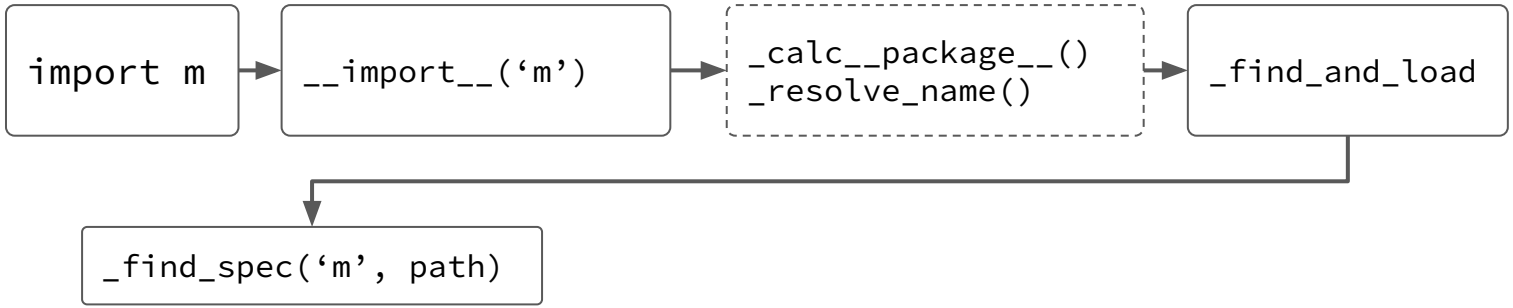
```
__import__('m', globals(), locals(), ['b_v'], level=1)
```



	__package__
module	None
submodule	parent.__name__
package	__name__

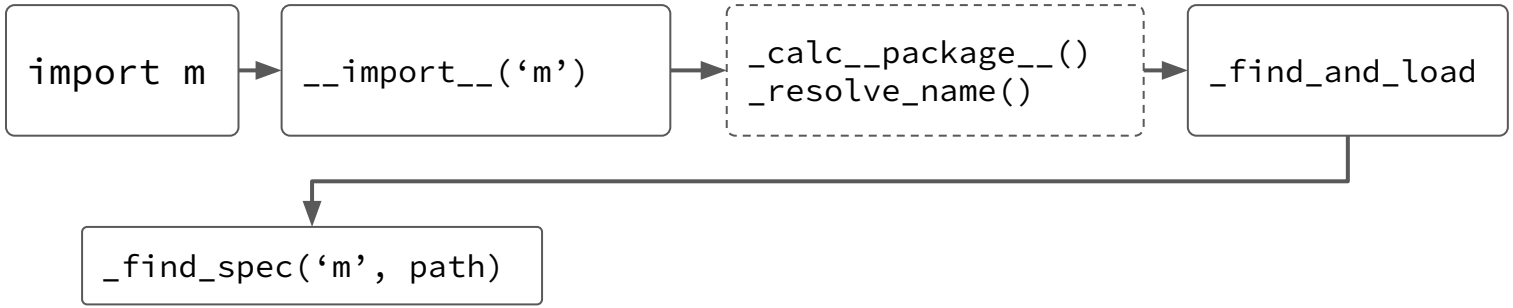
importlib/_bootstrap.py

```
1  def _calc___package__(globals):
2      """Calculate what __package__ should be.
3
4      __package__ is not guaranteed to be defined or could
   be set to None
5      to represent that its proper value is unknown.
6
7      """
8      package = globals.get('__package__')
9      spec = globals.get('__spec__')
10     …
11     return package
```




```
demo2
├── main.py              -> from p_foo.m_foo import print_foo
└── pkg_foo
    ├── __init__.py
    ├── module_bar.py
    └── module_foo.py    -> from m_bar import var_bar
```

	__package__
from m_bar	None
from .m_bar	'pkg_foo'

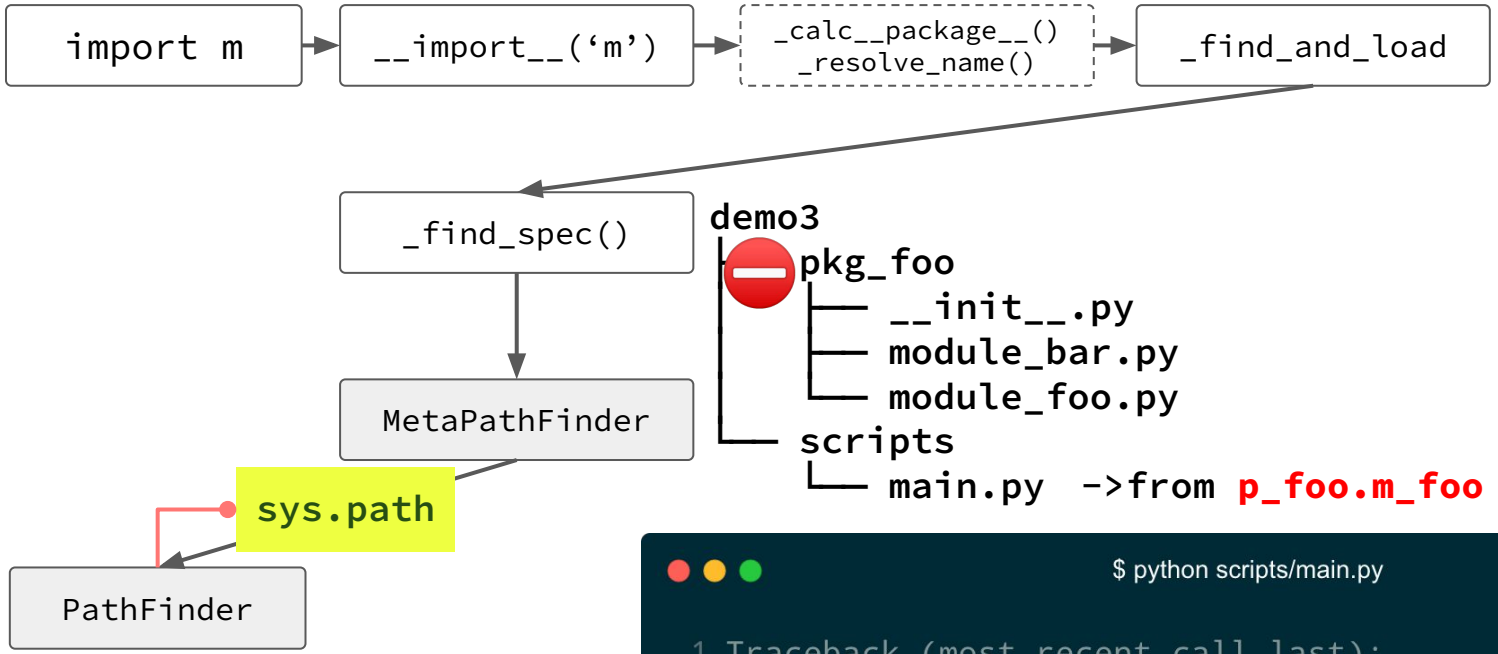



```
demo2
├── main.py              -> from p_foo.m_foo import print_foo
└── pkg_foo
    ├── __init__.py
    ├── module_bar.py
    └── module_foo.py    -> from .m_bar import var_bar
                           -> path='.../demo3/pkg_foo/'
```

	__package__
from m_bar	None
from .m_bar	'pkg_foo'

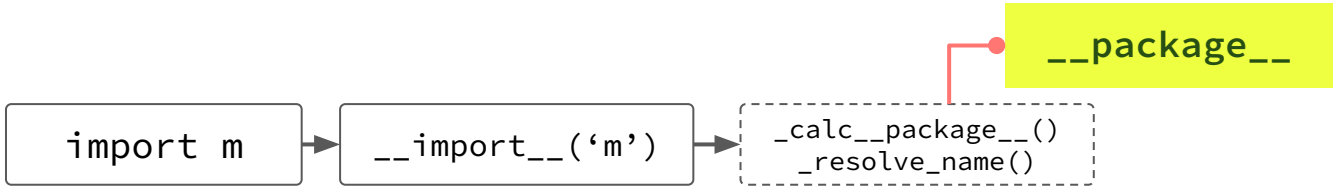
Revisit Demo 2

- Relative import use **`__package__`** of the current module
- Absolute import must have module locations in **`sys.path`**



$ python scripts/main.py

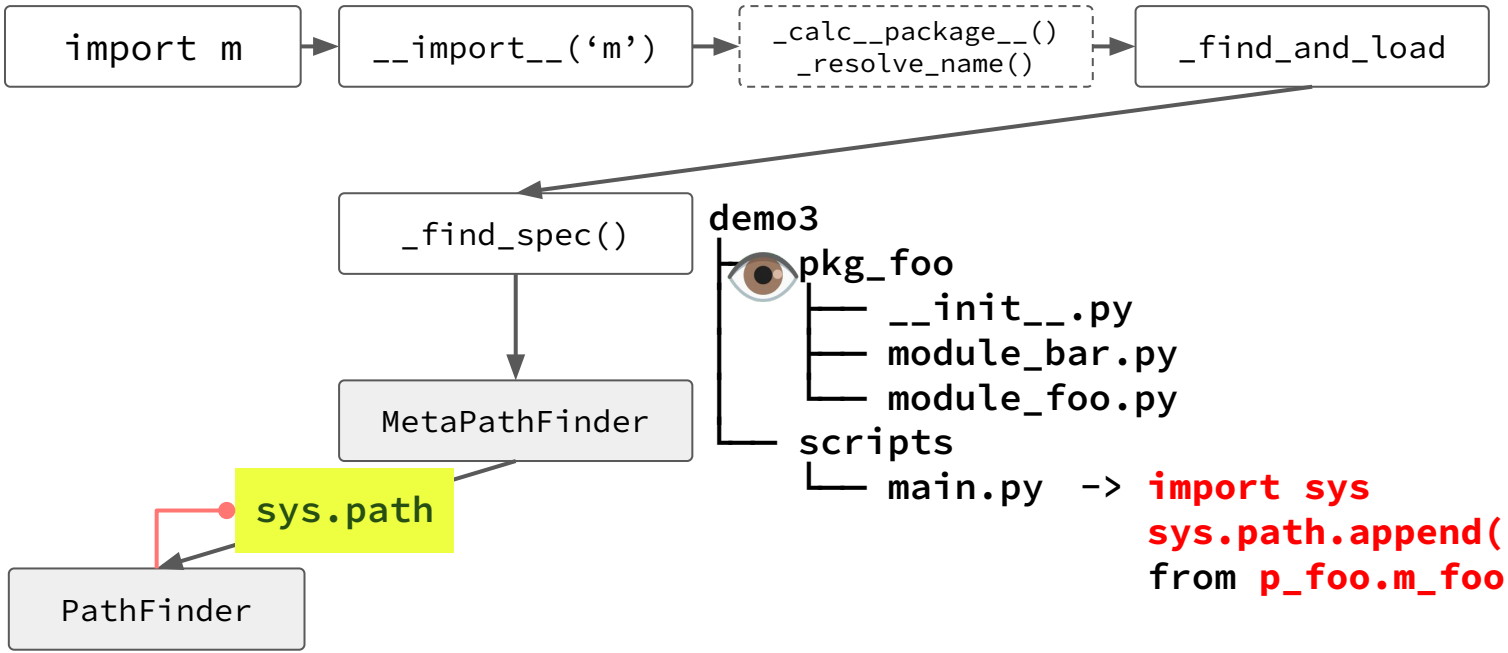
```
1 Traceback (most recent call last):
2   File "scripts/main.py", line 1, in <module>
3     from pkg_foo.module_foo import print_foo
4 ModuleNotFoundError: No module named 'pkg_foo'
```




```
demo3
├── pkg_foo
│   ├── __init__.py
│   ├── module_bar.py
│   └── module_foo.py
└── scripts
    └── main.py  -> from ..p_foo.m_foo import print_foo
```

$ python scripts/main.py

```
1 Traceback (most recent call last):
2   File "scripts/main.py", line 1, in <module>
3     from ..pkg_foo.module_foo import print_foo
4 ImportError: attempted relative import with no known
  parent package
```


import m
__import__('m')
_calc__package__()
_resolve_name()
_find_and_load
_find_spec()
MetaPathFinder
sys.path
PathFinder
demo3
pkg_foo
__init__.py
module_bar.py
module_foo.py
scripts
main.py -> import sys
sys.path.append('..')
from p_foo.m_foo import print_foo

Revisit Demo 3

- Relative import only works for packages
- **`sys.path`** includes an invocation-dependent current directory
- Absolute import must have module locations in **`sys.path`**

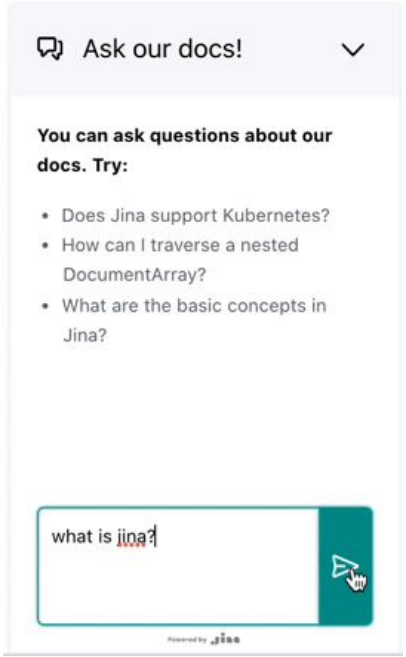
How do we use import in Jina

04

What is Neural Search?

Question-Answer Chatbot

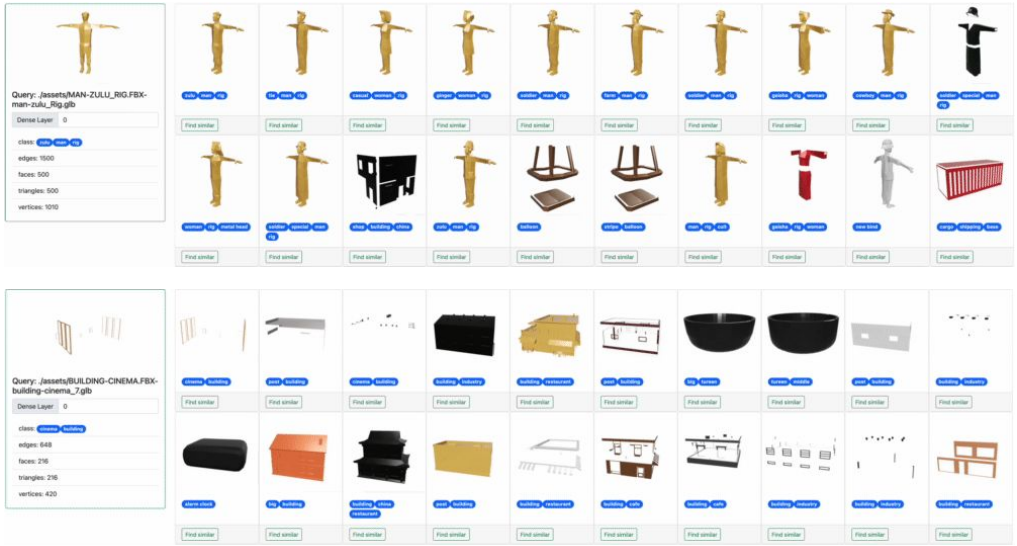
Answer arbitrary questions in a closed domain



What is Neural Search?

3D Mesh Look-alike

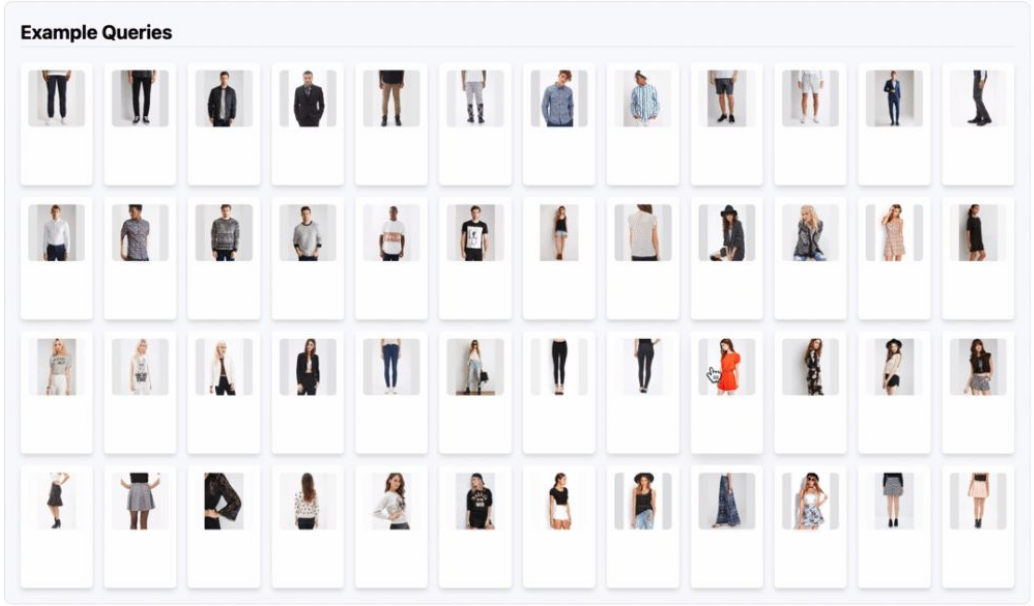
Easily find “look-alike” assets in game development



What is Neural Search?

Find Similar Fashion Items

Given a fashion item, find top-k most visually similar fashion items in stock.



Product Landscape

Jina Ecosystem, Comprehensive End2end Neural Search Experience

Productization

DocsQA

From DocsQA bot to user interaction analytics for business insights.

Finetuner API

Hosted Finetuning product aiming to solve last-mile search fine-tuning fitting enterprise reliability and availability expectations.

Jina Now

Self-service ordering kiosks for building Neural Search System.

And more ...

OSS

DocArray

targets at data scientists and AI engineers, focus on monolith programming, wrapping all vector databases (horizontal)

Jina

targets at scaling out local solution built via DocArray to the cloud (vertical)

Hub

OSS provides extensive opensource executors as NSS component for user to plug and play

Finetuner

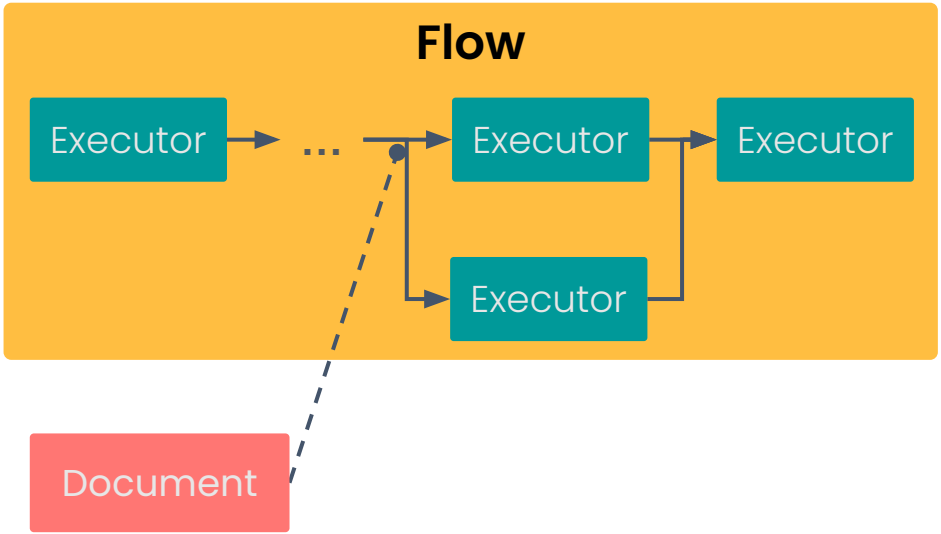
Oss version solve basic finetuning elements with loads of manual settings to set up search results finetune

Clip-as-Service

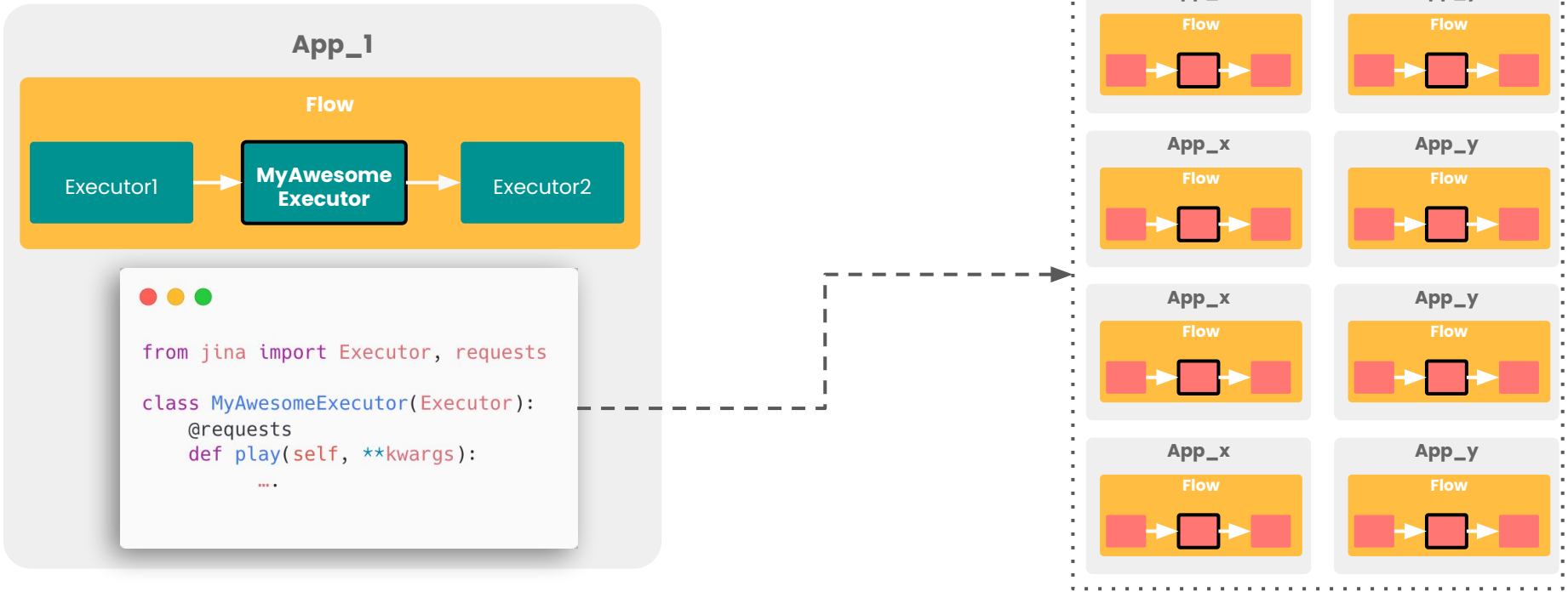
Embed images and sentences into fixed-length vectors with CLIP. The upgrade of Bert-as-Service

3 Basic Concepts

- **D**ocument: the basic data type
- **E**xecutor: a component to perform a single or a group of tasks
- **F**low: orchestrates Executors into a processing pipeline



Share your awesome Executor



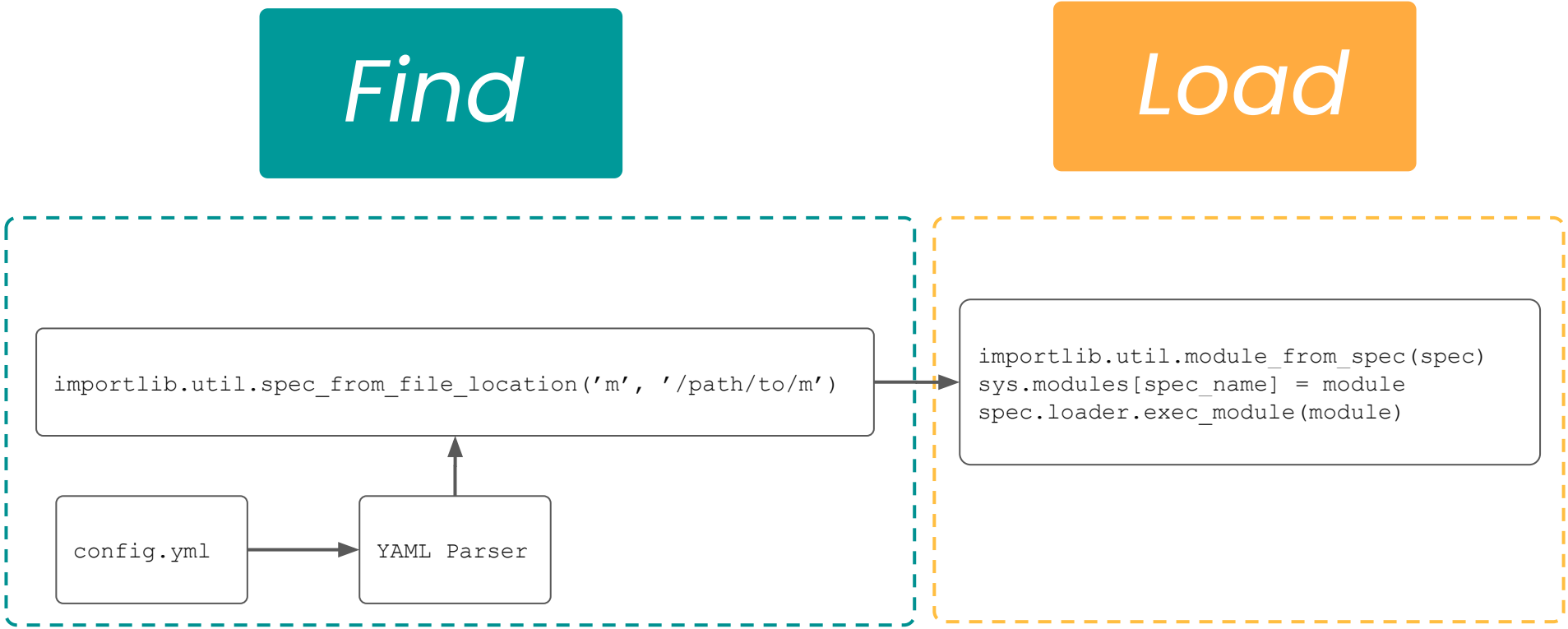
How we do imports in Jina?

jina/importer.py

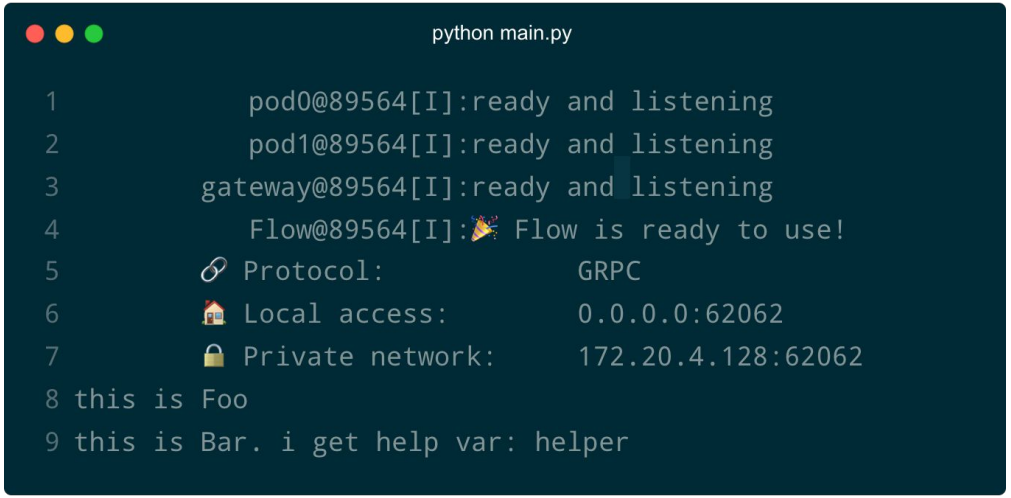
```python
def add_modules(*paths):
    """
    Import modules from paths.

    :param paths: Paths of the modules.
    """
    from jina.jaml.helper import complete_path

    paths = [complete_path(m) for m in paths]

    for p in paths:
        ...
        _path_import(p)
```

jina/importer.py

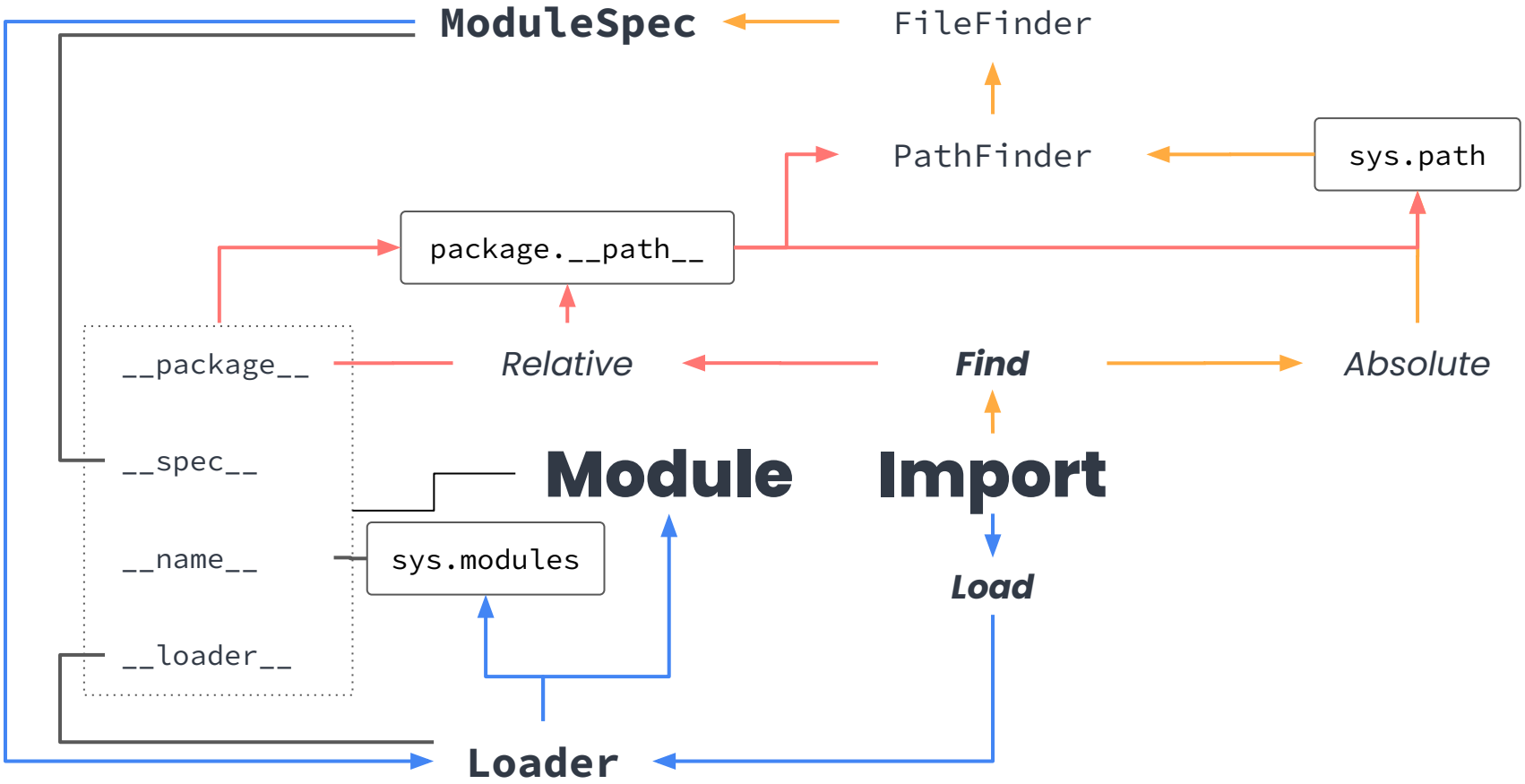
```python
def _path_import(absolute_path: str):
    import importlib.util

    try:
        default_spec_name = 'user_module'
        user_module_name = os.path.splitext(os.path.basename(absolute_path))[0]
        if user_module_name == '__init__':
            spec_name = default_spec_name
        elif user_module_name not in sys.modules:
            spec_name = user_module_name
        else:
            spec_name = f'{default_spec_name}.{user_module_name}'
        spec = importlib.util.spec_from_file_location(spec_name, absolute_path)
        module = importlib.util.module_from_spec(spec)
        sys.modules[spec_name] = module
        spec.loader.exec_module(module)
    except Exception as ex:
        ...
```


Find
Load
importlib.util.spec_from_file_location('m', '/path/to/m')
config.yml
YAML Parser
importlib.util.module_from_spec(spec)
sys.modules[spec_name] = module
spec.loader.exec_module(module)

Demo 4

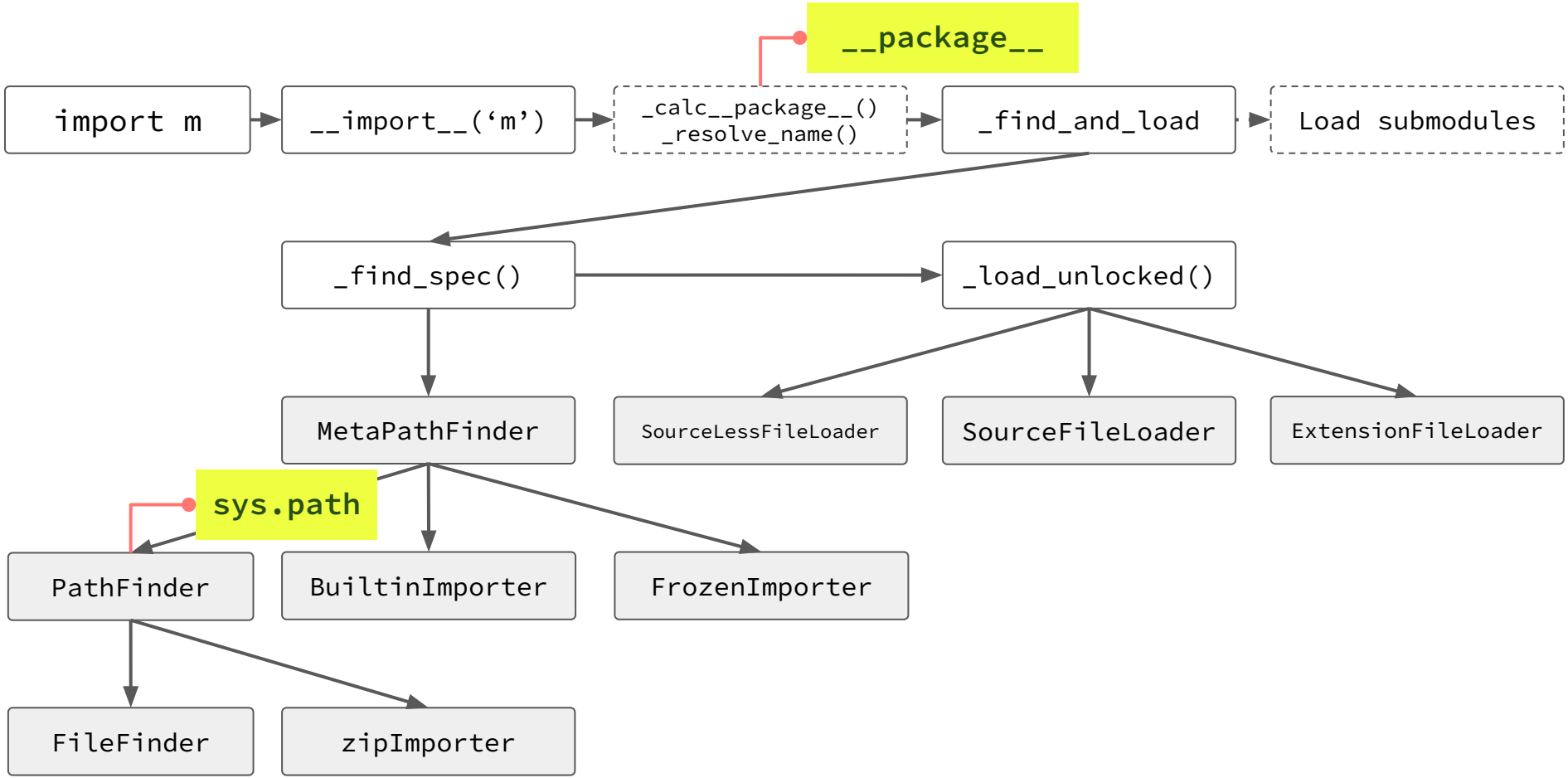



```
demo4
├── main.py
├── multi_files_executor
│   ├── config.yml
│   └── executor
│       ├── __init__.py
│       ├── my_executor.py
│       └── utils
│           ├── __init__.py
│           └── helper.py
└── single_file_executor
    ├── config.yml
    └── my_executor.py
```


ModuleSpec
FileFinder
PathFinder
sys.path
package.__path__
__package__
Relative
Find
Absolute
__spec__
Module
Import
__name__
sys.modules
Load
__loader__
Loader

Take-home Messages

- **`import`**的过程包含两个步骤: 查找**`ModuleSpec`**和加载**`Module`**.
- 相对路径需要使用当前模块的 **`__package__`** 信息
  - 相对路径只对**`package`**有效
- 绝对路径只查找 **`sys.path`**
  - **`sys.path`** 自动包含当前python脚本的运行目录


__package__
import m
__import__('m')
_calc__package__()
_resolve_name()
_find_and_load
Load submodules
_find_spec()
_load_unlocked()
MetaPathFinder
SourceLessFileLoader
SourceFileLoader
ExtensionFileLoader
sys.path
PathFinder
BuiltinImporter
FrozenImporter
FileFinder
zipImporter

Reference

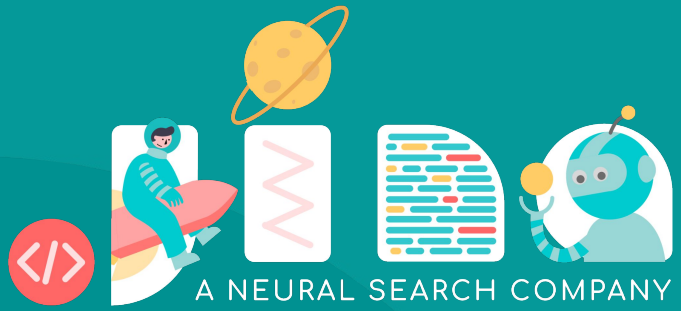
- Python behind the scenes #11: how the Python import system works
- official documentation: 5. The import system — Python 3.10.2 documentation
- import usage in procastinate: from hat import rabbit
- import developer talk: How Import Works - Brett Cannon - PyConAr 2012

.jina

Smart code goes into the libraries
Boring code goes into the projects.

.jina

Get bored?
Wanna write smart code?



WE ARE HIRING

jobs.jina.ai

Global

jina.ai

@ JinaAI_

get.jina.ai

slack.jina.ai