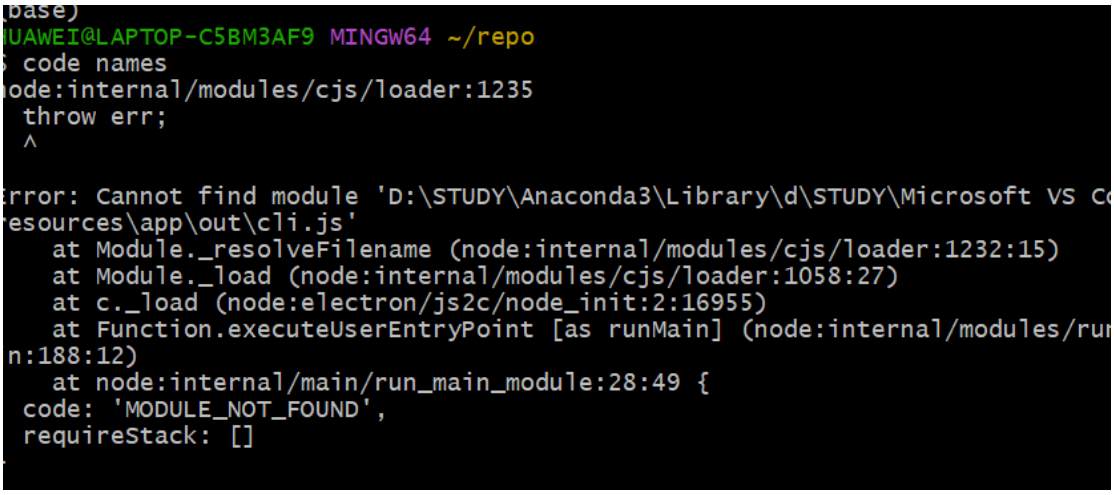
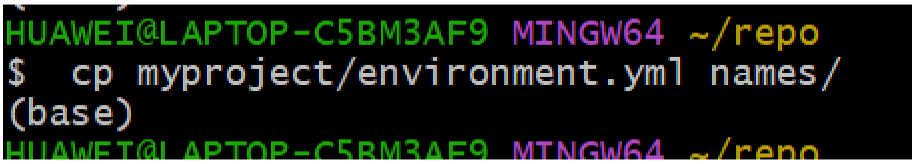


```
base)
UAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
 code names
ode:internal/modules/cjs/loader:1235
 throw err;
 ^

rror: Cannot find module 'D:\STUDY\Anaconda3\Library\d\STUDY\Microsoft VS Co
esources\app\out\cli.js'
   at Module._resolveFilename (node:internal/modules/cjs/loader:1232:15)
   at Module._load (node:internal/modules/cjs/loader:1058:27)
   at c._load (node:electron/js2c/node_init:2:16955)
   at Function.executeUserEntryPoint [as runMain] (node:internal/modules/ru
n:188:12)
   at node:internal/main/run_main_module:28:49 {
 code: 'MODULE_NOT_FOUND',
 requireStack: []
```

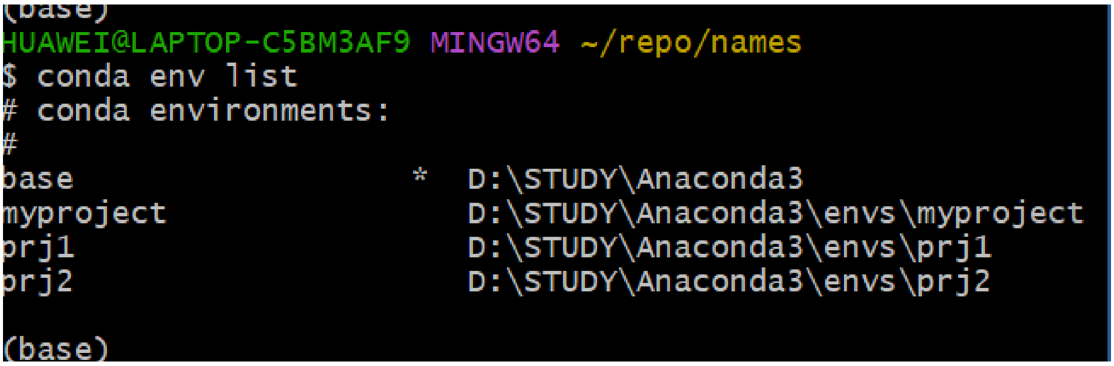


```
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
$  cp myproject/environment.yml names/
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
```

一、在 repo 中创建 names 文件夹并且在 code 打开。Cp 是负值命令，把环境的名字改为 names。

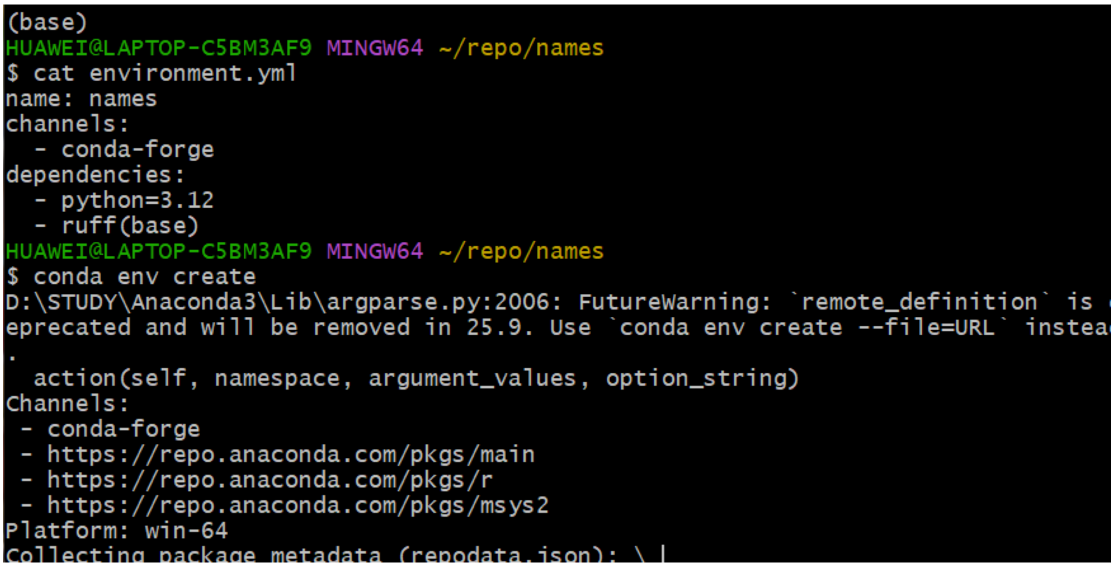
二、

看一下当前环境。

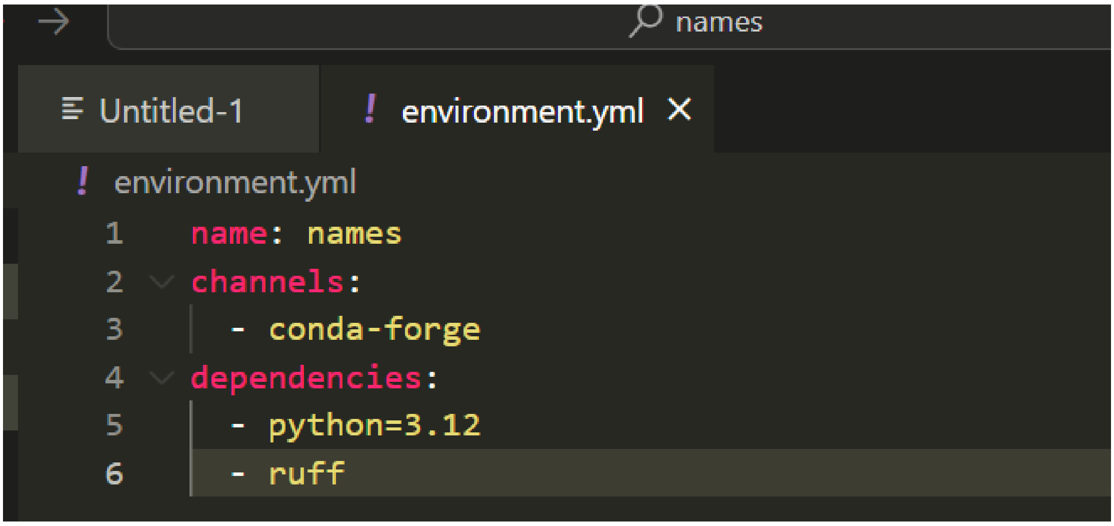


```
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/names
$ conda env list
# conda environments:
#
base                  *  D:\STUDY\Anaconda3
myproject                D:\STUDY\Anaconda3\envs\myproject
prj1                     D:\STUDY\Anaconda3\envs\prj1
prj2                     D:\STUDY\Anaconda3\envs\prj2

(base)
```

把在 vscode 的内容按 ctrlS 保存，然后



```
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/names
$ cat environment.yml
name: names
channels:
  - conda-forge
dependencies:
  - python=3.12
  - ruff(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/names
$ conda env create
D:\STUDY\Anaconda3\Lib\argparse.py:2006: FutureWarning: `remote_definition` is 
eprecated and will be removed in 25.9. Use `conda env create --file=URL` instea
.
  action(self, namespace, argument_values, option_string)
Channels:
 - conda-forge
 - https://repo.anaconda.com/pkgs/main
 - https://repo.anaconda.com/pkgs/r
 - https://repo.anaconda.com/pkgs/msys2
Platform: win-64
Collecting package metadata (repodata.json): \ |
```

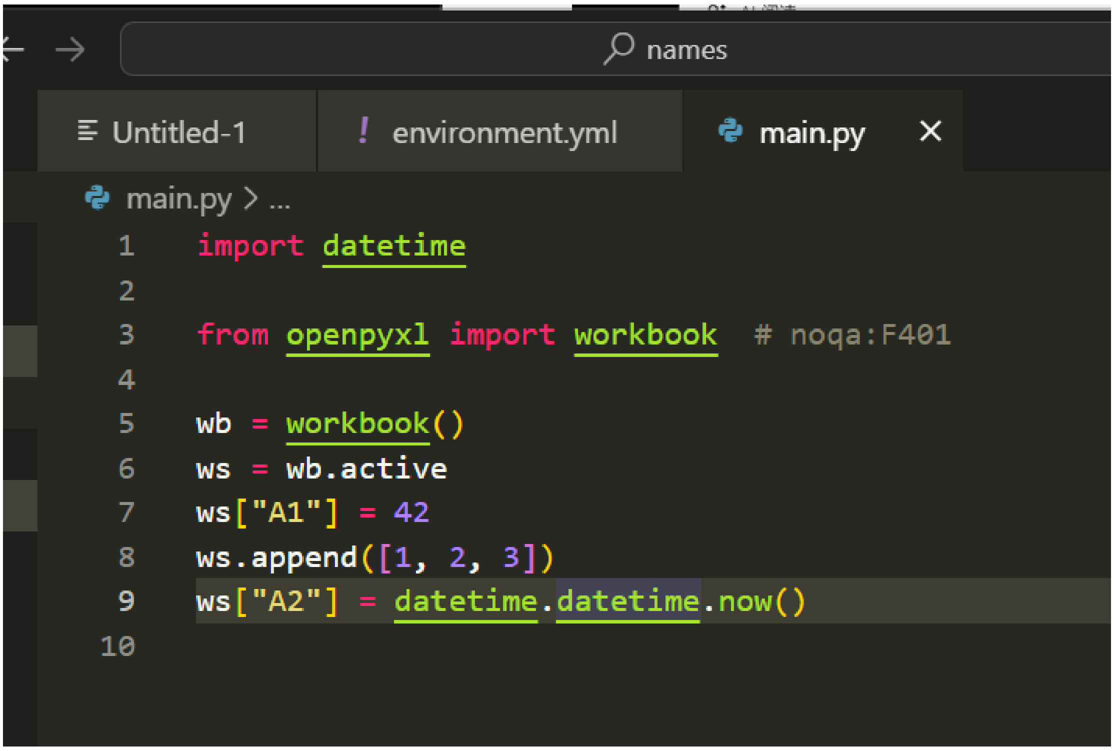


```yaml
name: names
channels:
  - conda-forge
dependencies:
  - python=3.12
  - ruff
```

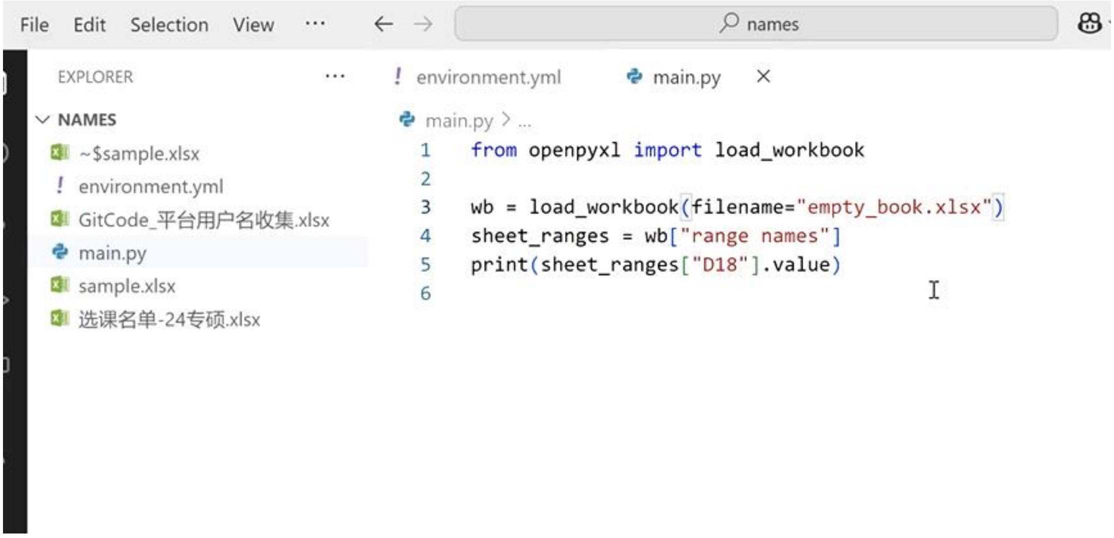
用 ctrl shift p 在 vscode 里打开搜索栏，搜索 interpreter,点击 find 寻找路径，在 anaconda3/names/python 点击 select。

```
$ conda env update
$ conda activate names
$ python main.py
```

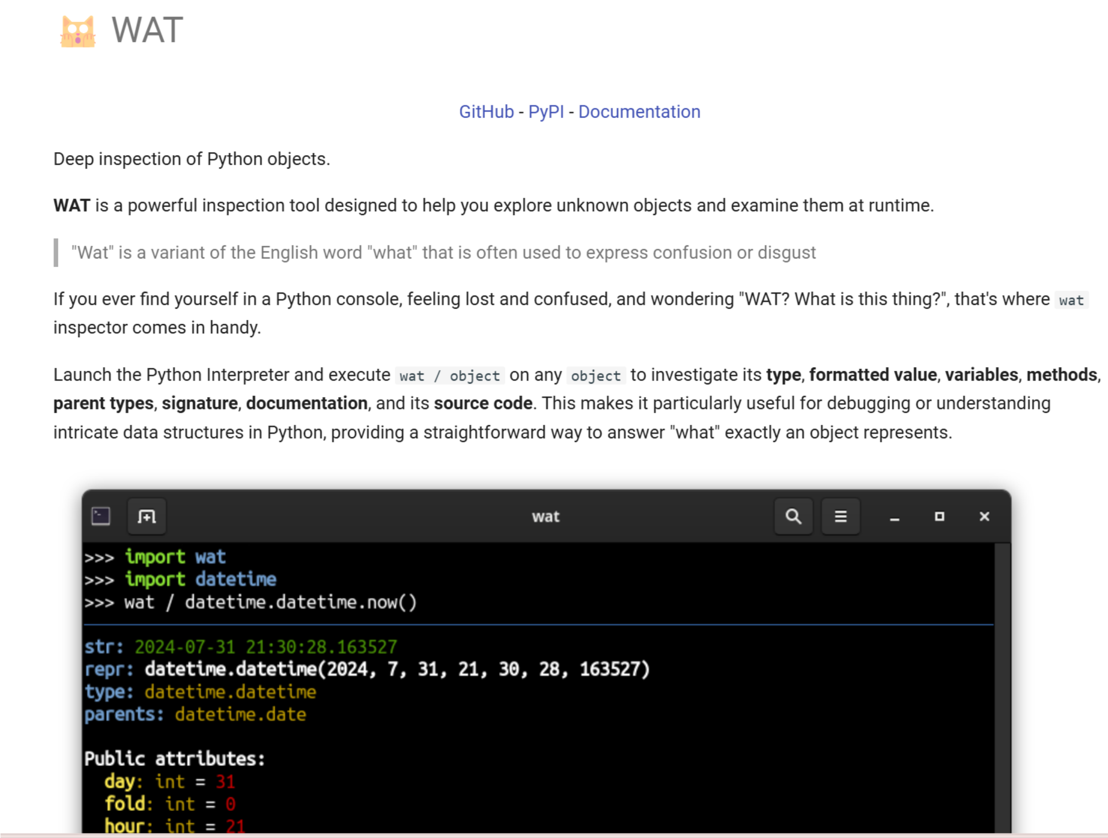
配置一下 main.py



```python
import datetime

from openpyxl import workbook  # noqa:F401

wb = workbook()
ws = wb.active
ws["A1"] = 42
ws.append([1, 2, 3])
ws["A2"] = datetime.datetime.now()
```



```python
from openpyxl import load_workbook

wb = load_workbook(filename="empty_book.xlsx")
sheet_ranges = wb["range names"]
print(sheet_ranges["D18"].value)
```



WAT

GitHub - PyPI - Documentation

Deep inspection of Python objects.

WAT is a powerful inspection tool designed to help you explore unknown objects and examine them at runtime.

> "Wat" is a variant of the English word "what" that is often used to express confusion or disgust

If you ever find yourself in a Python console, feeling lost and confused, and wondering "WAT? What is this thing?", that's where `wat` inspector comes in handy.

Launch the Python Interpreter and execute `wat / object` on any `object` to investigate its **type**, **formatted value**, **variables**, **methods**, **parent types**, **signature**, **documentation**, and its **source code**. This makes it particularly useful for debugging or understanding intricate data structures in Python, providing a straightforward way to answer "what" exactly an object represents.

```
>>> import wat
>>> import datetime
>>> wat / datetime.datetime.now()

str: 2024-07-31 21:30:28.163527
repr: datetime.datetime(2024, 7, 31, 21, 30, 28, 163527)
type: datetime.datetime
parents: datetime.date

Public attributes:
  day: int = 31
  fold: int = 0
  hour: int = 21
```

WAT