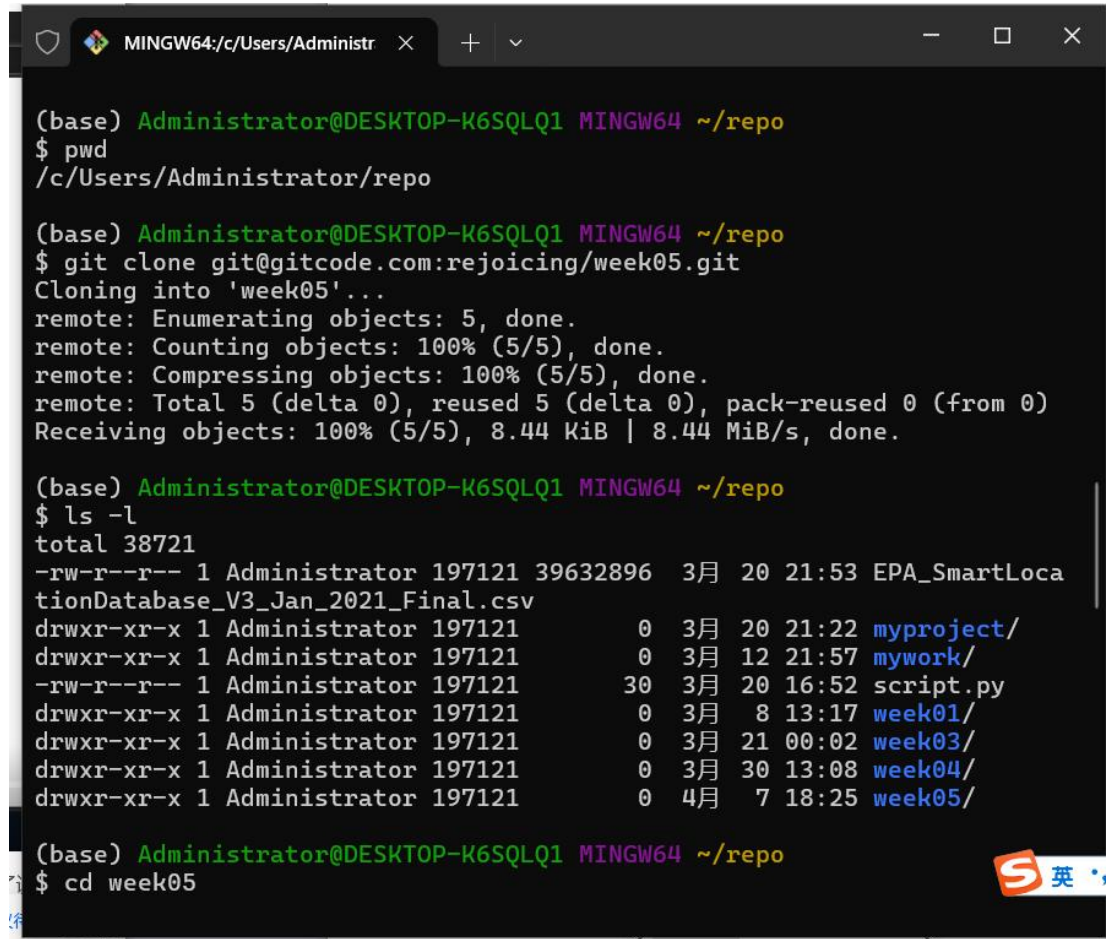


第五周学习笔记

1. 用 VS Code 打开项目目录，新建一个 environment.yml 文件，指定安装 Python 3.12，然后运行 conda env create 命令创建 Conda 环境



```
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ pwd
/c/Users/Administrator/repo

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ git clone git@gitcode.com:rejoicing/week05.git
Cloning into 'week05'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), 8.44 KiB | 8.44 MiB/s, done.

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ ls -l
total 38721
-rw-r--r-- 1 Administrator 197121 39632896  3月 20 21:53 EPA_SmartLoca
tionDatabase_V3_Jan_2021_Final.csv
drwxr-xr-x 1 Administrator 197121      0  3月 20 21:22 myproject/
drwxr-xr-x 1 Administrator 197121      0  3月 12 21:57 mywork/
-rw-r--r-- 1 Administrator 197121    30  3月 20 16:52 script.py
drwxr-xr-x 1 Administrator 197121      0  3月  8 13:17 week01/
drwxr-xr-x 1 Administrator 197121      0  3月 21 00:02 week03/
drwxr-xr-x 1 Administrator 197121      0  3月 30 13:08 week04/
drwxr-xr-x 1 Administrator 197121      0  4月  7 18:25 week05/

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ cd week05
```

```
MINGW64/c/Users/Administr... X + v
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ pwd
/c/Users/Administrator/repo/week05

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ ls -l
total 24
-rw-r--r-- 1 Administrator 197121 18805 4月 7 18:25 LICENSE
-rw-r--r-- 1 Administrator 197121 2239 4月 7 18:25 README.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ ls -la
./ ../ .git/ .gitignore LICENSE README.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ ls -al
total 36
drwxr-xr-x 1 Administrator 197121 0 4月 7 18:25 ./
drwxr-xr-x 1 Administrator 197121 0 4月 7 18:25 ../
drwxr-xr-x 1 Administrator 197121 0 4月 7 18:25 .git/
-rw-r--r-- 1 Administrator 197121 1307 4月 7 18:25 .gitignore
-rw-r--r-- 1 Administrator 197121 18805 4月 7 18:25 LICENSE
-rw-r--r-- 1 Administrator 197121 2239 4月 7 18:25 README.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ cat week04/environment.yml
cat: week04/environment.yml: No such file or directory

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ cat week04/environment.yml
```

```
MINGW64/c/Users/Administr... x + v
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ cd
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~
$ cd repo
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ pwd
/c/Users/Administrator/repo
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ cat week04/environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ cp week04/environment.yml week05/
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 Administrator 197121  91  4月  7 18:32 environment.yml
-rw-r--r-- 1 Administrator 197121 18805 4月  7 18:25 LICENSE
-rw-r--r-- 1 Administrator 197121  2239 4月  7 18:25 README.md
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$
```

2. 逐个创建 `use_of_{name}.py` 文件，其中 `{name}` 替换为上述要求掌握的对象类型，例如 `use_of_str.py`：

在全局作用域 (global scope) 内尝试键入 (活学活用) Python 代码，亲手验证概念 (Proof of Concept, PoC)

对于任何对象，都可以传给以下内置函数 (built-in function) 用于检视 (inspect)：

`id()` -- 返回对象在虚拟内存中的地址 (正整数)，如果 `id(a) == id(b)`，那么 `a is b` (`is` 是个运算符)

```
1 myproject D:\haozhiting\Anaconda3\envs\myproject
1 prj1 D:\haozhiting\Anaconda3\envs\prj1
prj2 D:\haozhiting\Anaconda3\envs\prj2
week04 D:\haozhiting\Anaconda3\envs\week04
week05 D:\haozhiting\Anaconda3\envs\week05

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ conda activate
(base)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2030697274352
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

type() -- 返回对象的类型

isinstance() -- 判断对象是否属于某个 (或某些) 类型

dir() -- 返回对象所支持的属性 (attributes) 的名称列表

str() -- 返回对象 print 时要显示在终端的字符串

可以调用 print() 函数将表达式 (expression) 输出到终端, 查看结果是否符合预期

```
MINGW64: c:/Users/Administrat...
bash: ython: command not found
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print(32)
32
>>> print(str(32))
32
>>> quit()
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
C:\Users\Administrator\repo\week05\use_of_str.py:14: SyntaxWarning: 'str' object is not callable; perhaps you missed a comma?
  print("dir(a):"(a))
2408635177216
2408635175232
[9, 5]
[2, 5]
2408635177216
2408635175232
<class 'list'>
isinstance(a, str): False
Traceback (most recent call last):
  File "C:\Users\Administrator\repo\week05\use_of_str.py", line 14, in <module>
```

可以利用 `assert` 语句查验某个表达式 (expression) 为真, 否则报错 (AssertionError) 退出
可以利用 `try` 语句拦截报错, 避免退出, 将流程 (flow) 转入 `except` 语句


```
MINGW64: c:/Users/Administrat...
[2, 5]
1746509895936
1746509893952
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
Traceback (most recent call last):
  File "C:\Users\Administrator\repo\week05\use_of_str.py", line 15, in
    <module>
      print("dir(a):"(a))
                        ^^^^^^^^^^^^^^^
TypeError: 'str' object is not callable
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2700958963968
2700958961984
[9, 5]
[2, 5]
2700958963968
2700958961984
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
type error
goodbye
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

可以调用 `breakpoint()` 函数暂停程序运行，进入 `pdb` 调试 (debug) 模式

```
MINGW64/c/Users/Administr... x + v
*** SyntaxError: invalid syntax
(Pdb) python use_of_str.py
*** SyntaxError: invalid syntax
(Pdb) q
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2099487643904
2099487641920
[9, 5]
[2, 5]
2099487643904
2099487641920
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
> c:\users\administrator\repo\week05\use_of_str.py(20)<module>()
-> print("type error")
(Pdb) l .
15     print(isinstance(a, (str, float)))
16     try:
17         assert isinstance(a, str)
18     except AssertionError:
19         breakpoint()
20     -> print("type error")
21
22     print("goodbye")
[EOF]
(Pdb) █
```

3.对于 每一个 上述要求掌握的对象类型 (将来遇到新的对象类型也应该如此), 我们首先应该熟悉如何通过 表达式 (expression) 得到他们的 实例 (instance), 一般包括以下途径:

字面值 (literal) (包括 f-string 语法)

推导式 (comprehension) (仅限 list、dict、set)

初始化 (init)

运算值 (operator)

索引值 (subscription)

返回值 (return value of function/method call)

4.对于 每一个 上述要求掌握的对象类型 (将来遇到新的对象类型也应该如此), 我们也要尝试验证其以下几个方面的 属性 (attributes):

对数学运算符 (+、-、*、/、//、%、@) 有没有支持

```
MINGW64:/c/Users/Administr
+ v
assert str([5,8,2])=='[5, 8,2]'
      ^
SyntaxError: invalid syntax
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
True
f-string
name: Tom
TAB a    b
  New Line aaa
bbb
xyz
abc
  eee
aaa

初始化

[5, 8, 2]
Traceback (most recent call last):
  File "C:\Users\Administrator\repo\week05\use_of_str.py", line 28, in
    <module>
      assert str([5, 8, 2]) == "[5, 8,2]"
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```



```
27 print(s)
28 assert str([5, 8, 2]) == "[5, 8, 2]"
29 assert str(1.1 + 2.2) != "3.3"
30
31 assert str() == ""
32
33 s = "="
34 x = id(s)
35 s = s * 20
36 y = id(s)
37 print(s)
38 assert x != y
39 s = "hello"
40 assert s[3] == "l"
41 assert s[-1] == "o"
42 assert s[:3] == "hel"
43 assert s[4] == s[-1]
44 try:
45     s[5]
46 except IndexError as e:
47     print(e)
48
49 s = "hello"
50 u = s.upper()
51 print(u)
52 print(s)
53
54 t = "name: {}, age {}"
55 print(t)
56 t1 = t.format("Jack", 21)
57 print(t1)
58
```

```
string index out of range
HELLO
hello
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
True
f-string
name: Tom
TAB a    b
  New Line aaa
bbb
xyz
abc
  eee
aaa

初始化

[5, 8, 2]
=====
string index out of range
HELLO
hello
name: {},age {}
name: Jack,age 21
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

```
59 s1 = "abc"
60 s2 = "ghi"
61 s = s1 + s2
62 assert s == "abcghi"
63 print(s2 + s1)
64
65 try:
66     print(s2 - s1)
67 except TypeError as e:
68     print(e)
69
70 s = "=="
71 s = s * 10
72 print(s)
73
74 s = "aaaa"
75 s = s / 2
76
```

Ln 73, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 ('base': conda)

```
MINGW64: c:/Users/Administr
name: Tom
TAB a  b
New Line aaa
bbb
xyz
abc
eee
aaa

初始化

[5, 8, 2]
=====
string index out of range
HELLO
hello
name: {},age {}
name: Jack,age 21
ghiabc
unsupported operand type(s) for -: 'str' and 'str'
=====
Traceback (most recent call last):
  File "C:\Users\Administrator\repo\week05\use_of_str.py", line 75, in
<module>
    s = s / 2
      ~^~~
TypeError: unsupported operand type(s) for /: 'str' and 'int'
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

如何判断相等 (==)

对于比较运算符 (>、<、>=、<=) 有没有支持

什么值被当作 True, 什么值被当作 False

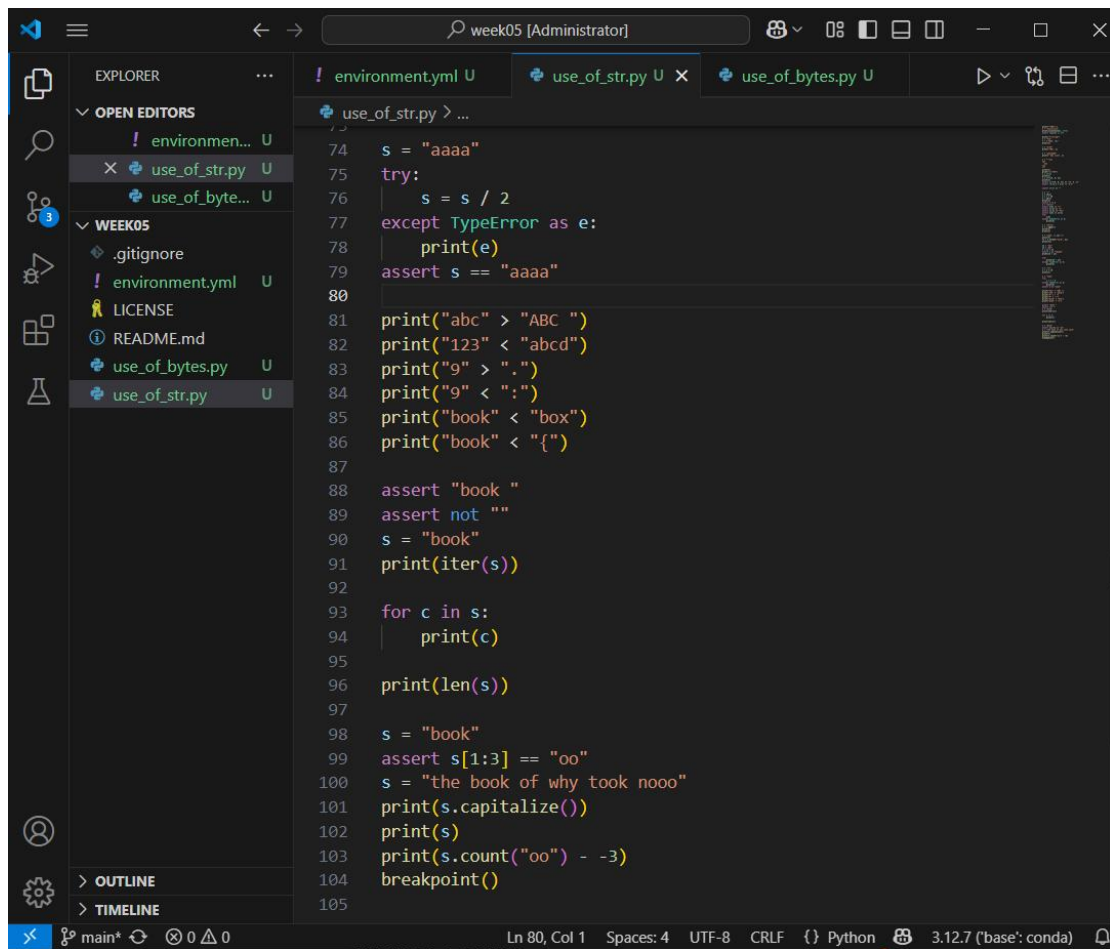
是否可迭代 (iterable), 如何做迭代 (for 循环)

是否支持返回长度 (len)

是否 (如何) 支持索引操作 (subscription) ([]) 运算符

拥有哪些常用方法 (method) 可供调用 (()) 运算符

建议先在 pdb 里试验, 然后把确定能够运行的代码写在 use_of_{name}.py 文件里



```
mingw64: c:/Users/Administr... x + v - □ ×
ghiabc
unsupported operand type(s) for -: 'str' and 'str'
=====
unsupported operand type(s) for /: 'str' and 'int'
True
True
True
True
True
True
True
<str_ascii_iterator object at 0x00000187E13FDD20>
b
o
o
k
4
The book of why took nooo
the book of why took nooo
6
True
True
True
False
True
False
rose:jack:bob
['rose', 'jack', 'bob']
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

字节串:

```
1  from pathlib import Path
2
3  s = b"hello"
4  print(s)
5  print(s[0])
6
7  p = Path("D:\\haozhiting\\Anaconda3\\python.exe")
8  s = p.read_bytes()
9  print(len(s))
10
11 p = Path("environment.yml")
12 b = p.read_bytes()
13 print(b[0])
14
15 s = b.decode()
16 assert isinstance(s, str)
17 b2 = s.encode()
18 assert isinstance(b2, bytes)
19 assert b2 == b
20 s = "你好"
21 b1 = s.encode("utf-8")
22 print(b1)
23 b2 = s.encode("gbk")
24 print(b2)
25 s = "abc你好🐼"
26 print(s)
27 b = s.encode()
28 breakpoint()
29
```



```
MINGW64:/c/Users/Administr ✕ + ▾
FileNotFoundError: [Errno 2] No such file or directory: 'C: \\IUsers
llqiang\\lanacoda3\\ lenvs\\ [weeke5\\python.exe'
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
C:\Users\Administrator\repo\week05\use_of_bytes.py:7: SyntaxWarning:
  invalid escape sequence '\h'
  p = Path("D:\haozhiting\Anaconda3\python.exe")
b'hello'
104
104208
110
b'\xe4\xbd\xa0\xe5\xa5\xbd'
b'\xc4\xe3\xba\xc3'
abc你好 😊
--Return--
> c:\users\administrator\repo\week05\use_of_bytes.py(28)<module>()->
None
-> breakpoint()
(Pdb) p b
b'abc\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\xa7\x90'
(Pdb) p b[3:].decode
<built-in method decode of bytes object at 0x000002116ACFD500>
(Pdb) p b[3:].decode()
'你好 😊'
(Pdb) p b[3:9].decode()
'你好 '
(Pdb) p b[9:].decode()
'😊'
(Pdb) █
```

整数:

week05 [Administrator]

EXPLORER

OPEN EDITORS

- environment.yml U
- use_of_str.py U
- use_of_byte... U
- use_of_int.py U

WEEK05

- .gitignore
- environment.yml U
- LICENSE
- README.md
- use_of_bytes.py U
- use_of_int.py U
- use_of_str.py U

use_of_int.py > ...

```
1 i = 42
2 x = 5
3 y = 7
4 z = x + y
5
6 x = 5
7 y = 17
8 assert y // x == 3
9 assert y % x == 2
10 assert 5
11
12 try:
13     assert 0
14 except AssertionError as e:
15     print(type(e))
16
17 x = 65534
18 breakpoint()
19 |
```

main* 0 0

Ln 19, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 ('base': conda)


```
1 import random
2
3 x = 3.14
4 print(type(x))
5 y = float("3.14")
6 print(type(y))
7
8 assert x == y
9 x = 5 / 3
10 print(x, type(x))
11 x = random.random()
12 print(x)
13 assert not 0.0
14
15 nan = float("nan")
16 print(nan + 3)
17 print(nan > 3)
18 print(nan < 3)
19 print(nan == 3)
20 pinf = float("inf")
21 print(3.14e-2)
22 print(pinf > 1e200)
23 print(pinf > pinf)
24 print(pinf == pinf)
25
26 ninf = float("-inf")
27 print(ninf)
28
```

```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.6716736282582211
nan
False
False
False
0.0314
True
False
True
-inf
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

布尔值:

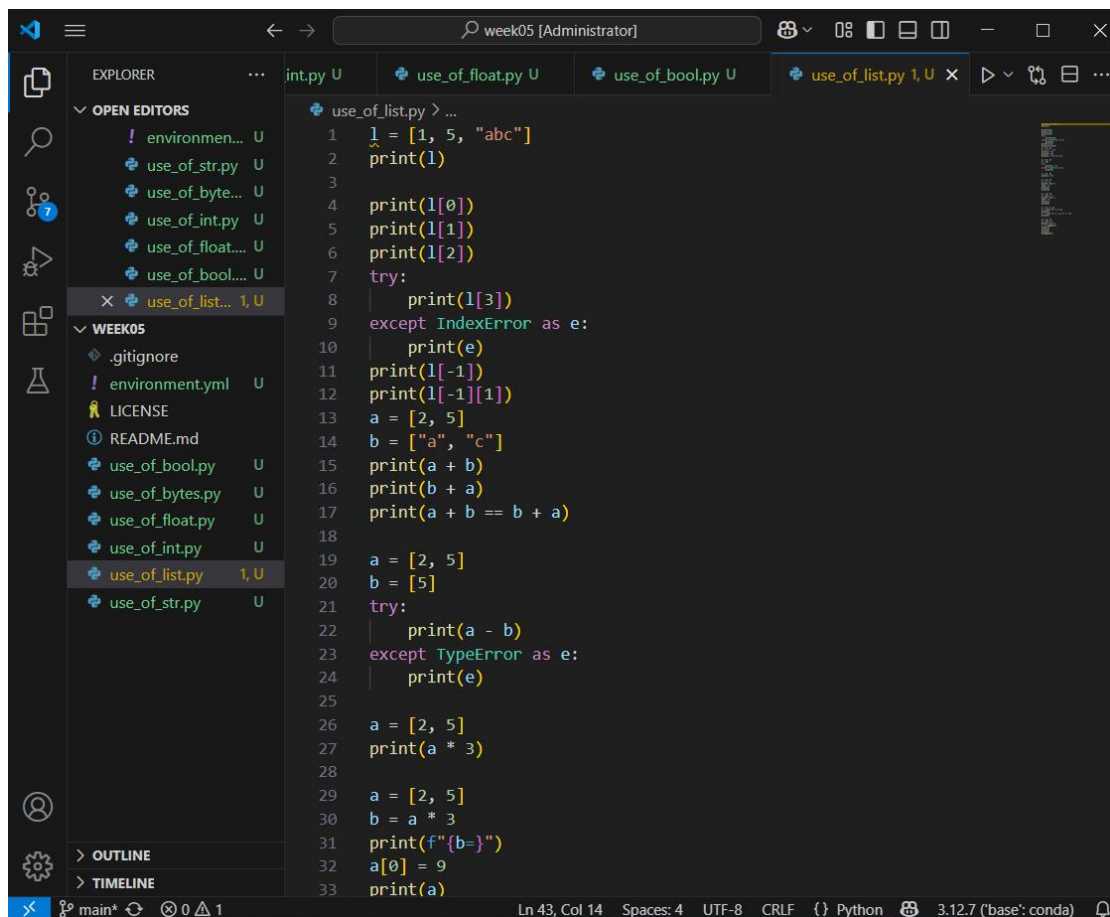
The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'week05' with several Python files. The file 'use_of_bool.py' is open in the editor. The code in the file is as follows:

```
1 t = True
2 f = False
3 print(t, f)
4
5 print(type(t))
6 print(isinstance(t, int))
7
```

Below the editor, a terminal window is open, showing the execution of the script. The output is as follows:

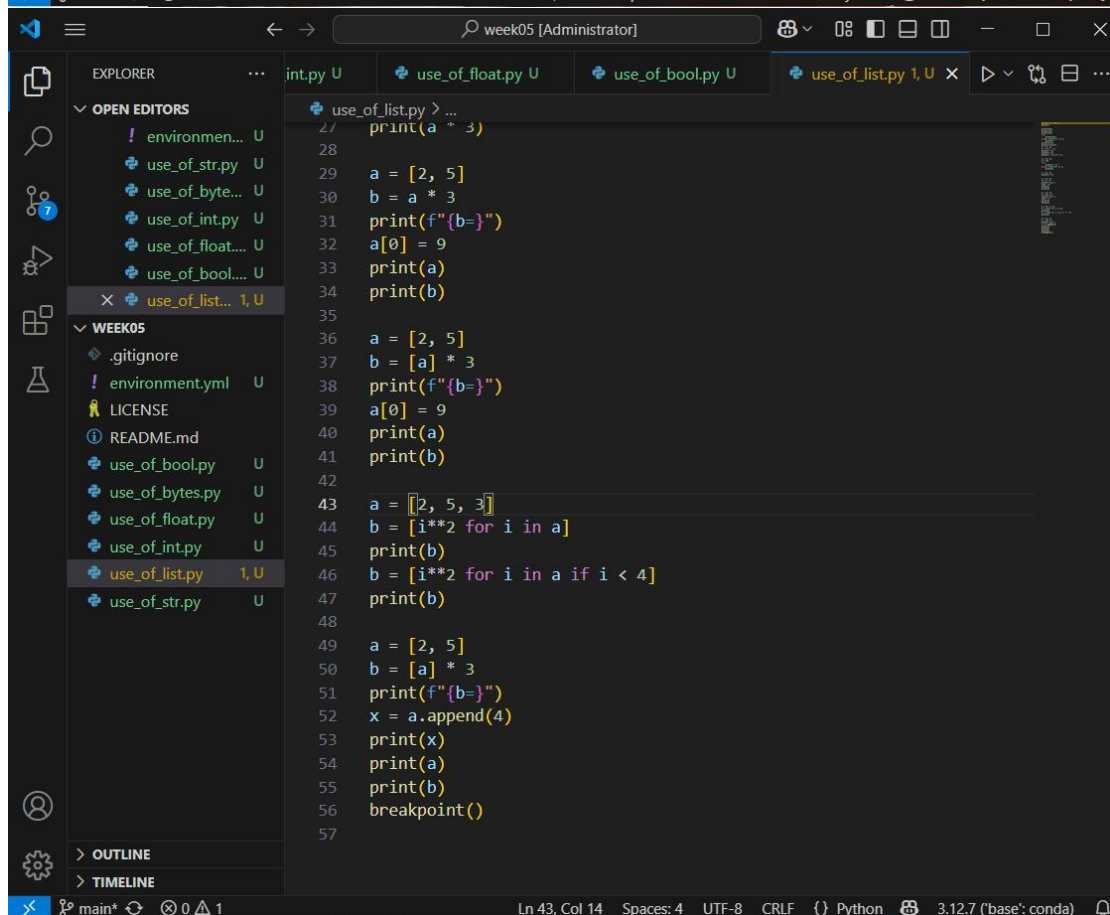
```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
<class 'bool'>
True
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

列表:



```
1 l = [1, 5, "abc"]
2 print(l)
3
4 print(l[0])
5 print(l[1])
6 print(l[2])
7 try:
8     print(l[3])
9 except IndexError as e:
10     print(e)
11 print(l[-1])
12 print(l[-1][1])
13 a = [2, 5]
14 b = ["a", "c"]
15 print(a + b)
16 print(b + a)
17 print(a + b == b + a)
18
19 a = [2, 5]
20 b = [5]
21 try:
22     print(a - b)
23 except TypeError as e:
24     print(e)
25
26 a = [2, 5]
27 print(a * 3)
28
29 a = [2, 5]
30 b = a * 3
31 print(f"{b=}")
32 a[0] = 9
33 print(a)
```

Ln 43, Col 14 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 ('base': conda)



```
27 print(a * 3)
28
29 a = [2, 5]
30 b = a * 3
31 print(f"{b=}")
32 a[0] = 9
33 print(a)
34 print(b)
35
36 a = [2, 5]
37 b = [a] * 3
38 print(f"{b=}")
39 a[0] = 9
40 print(a)
41 print(b)
42
43 a = [2, 5, 3]
44 b = [i**2 for i in a]
45 print(b)
46 b = [i**2 for i in a if i < 4]
47 print(b)
48
49 a = [2, 5]
50 b = [a] * 3
51 print(f"{b=}")
52 x = a.append(4)
53 print(x)
54 print(a)
55 print(b)
56 breakpoint()
57
```

Ln 43, Col 14 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 ('base': conda)


```
MINGW64:/c/Users/Administr × + v - □ ×
$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
b
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
unsupported operand type(s) for -: 'list' and 'list'
[2, 5, 2, 5, 2, 5]
b=[2, 5, 2, 5, 2, 5]
[9, 5]
[2, 5, 2, 5, 2, 5]
b=[[2, 5], [2, 5], [2, 5]]
[9, 5]
[[9, 5], [9, 5], [9, 5]]
[4, 25, 9]
[4, 9]
b=[[2, 5], [2, 5], [2, 5]]
None
[2, 5, 4]
[[2, 5, 4], [2, 5, 4], [2, 5, 4]]
--Return--
> c:\users\administrator\repo\week05\use_of_list.py(56)<module>()->N
one
-> breakpoint()
(Pdb) █
```

字典：

```
1 d = {"a": 1, "bb": 5, "cat": 3}
2 print(d)
3 print(type(d))
4
5 for a in d:
6     print(a)
7 for a in d:
8     print(d[a])
9 for a in d.values():
10    print(a)
11 l = [a for a in d.items()]
12 print(l)
13
14 for k, v in d.items():
15     print(k, v)
16
17 breakpoint()
18
```

```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_dict.py
{'a': 1, 'bb': 5, 'cat': 3}
<class 'dict'>
a
bb
cat
1
5
3
1
5
3
[('a', 1), ('bb', 5), ('cat', 3)]
a 1
bb 5
cat 3
--Return--
> c:\users\administrator\repo\week05\use_of_dict.py(17)<module>()->N
one
-> breakpoint()
(Pdb)
```

元组:

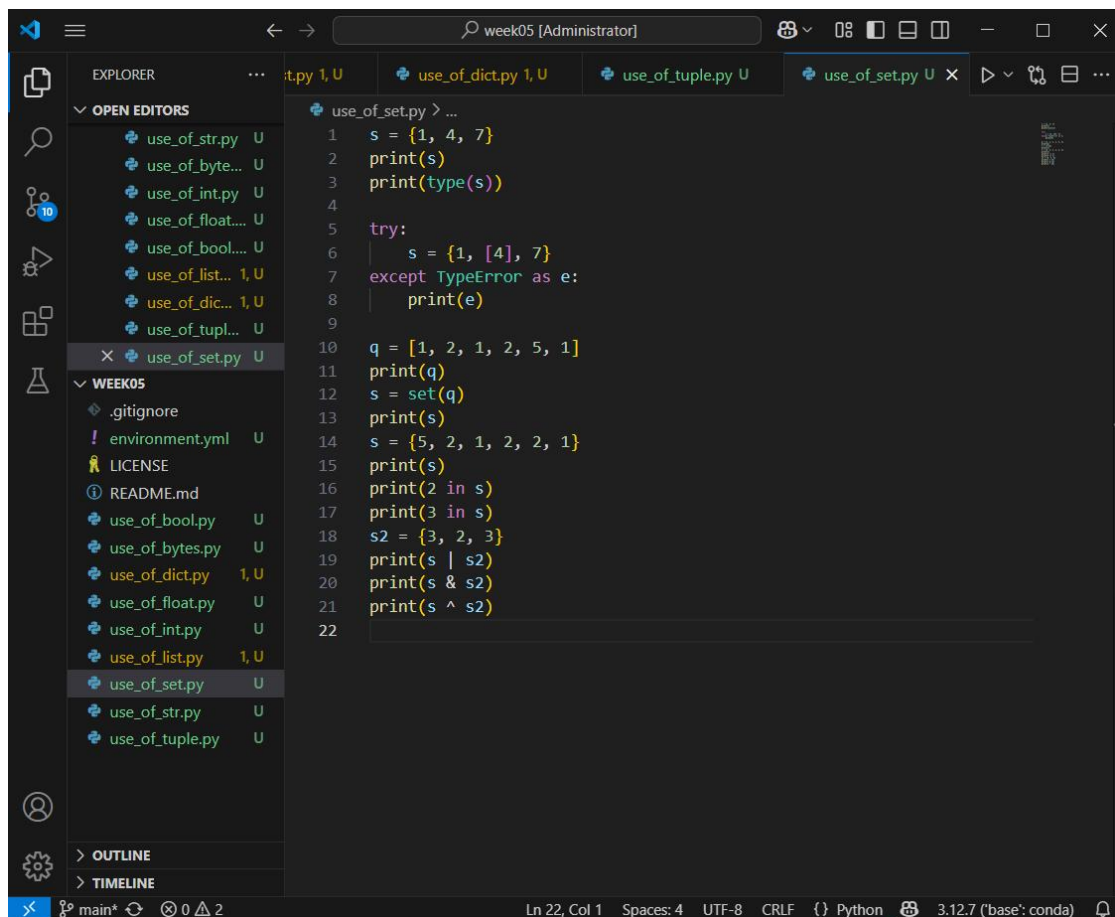
The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left lists files in the 'WEEK05' directory, including 'use_of_tuple.py'. The main editor window displays the code for 'use_of_tuple.py'. The code defines a tuple 't', prints its elements and type, attempts to modify an element (causing a TypeError), defines a dictionary 'd', prints its contents, and defines another tuple 't'.

```
1 t = (1, "a", 3.14)
2 print(t)
3 print(type(t))
4
5 print(t[0])
6 print(t[1])
7 print(t[2])
8
9 try:
10     t[0] = 9
11 except TypeError as e:
12     print(e)
13
14 d = {}
15 d["abc"] = 5
16 d[7] = 100
17 q = [3, 1]
18
19 try:
20     d[q] = 21
21 except TypeError as e:
22     print(e)
23 t = (3, 1)
24 d[t] = 21
25 print(d)
26 print(d[3, 1])
27
28 t = 1, 4, 0, 2
29 print(t)
30 print(type(t))
31
```

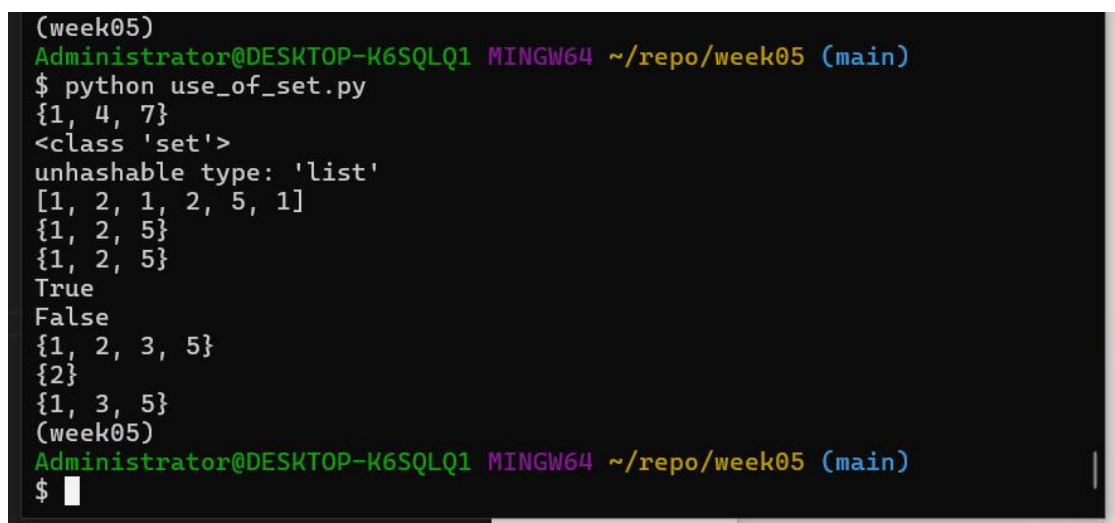
The screenshot shows a terminal window with the output of running 'python use_of_tuple.py'. The output matches the code execution: it prints the tuple, its type, attempts to modify it (resulting in a 'TypeError: unhashable type: 'list'' message), prints the dictionary, and prints the new tuple.

```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
'tuple' object does not support item assignment
unhashable type: 'list'
{'abc': 5, 7: 100, (3, 1): 21}
21
(1, 4, 0, 2)
<class 'tuple'>
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

集合:



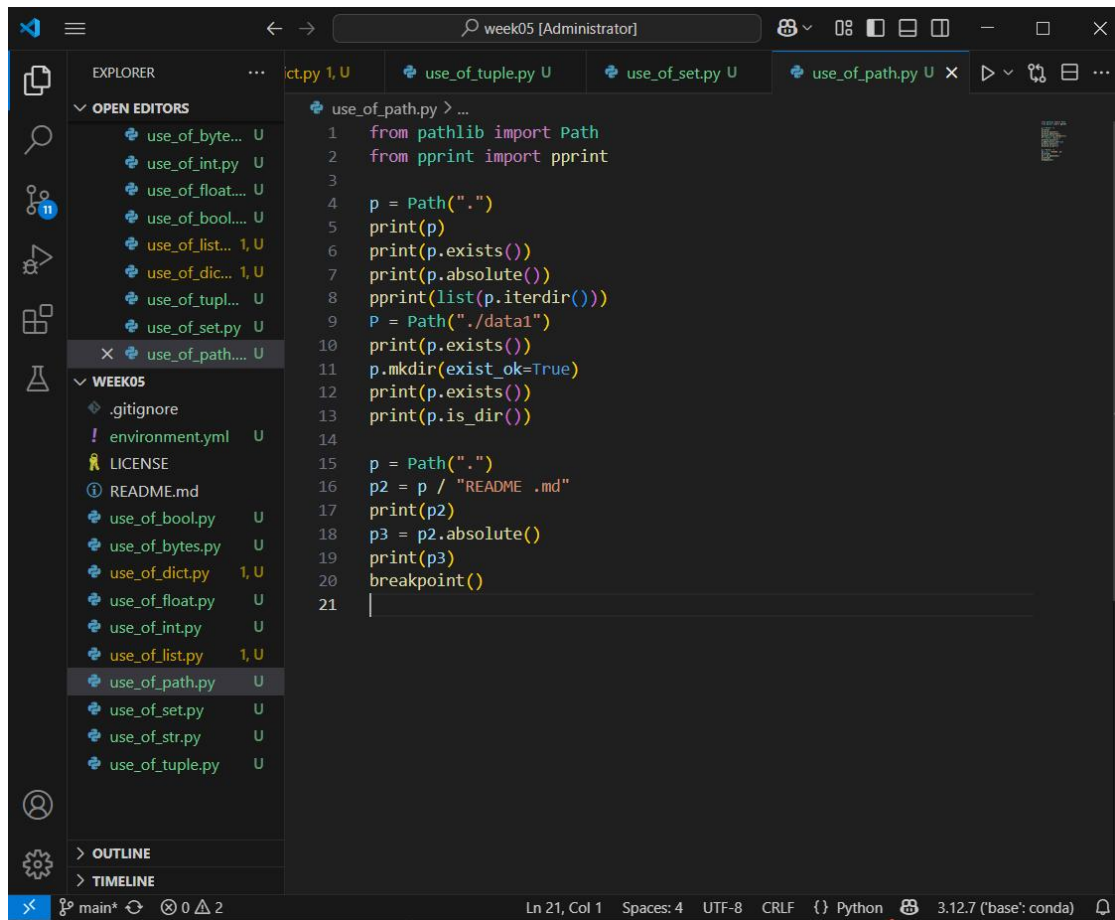
```
1 s = {1, 4, 7}
2 print(s)
3 print(type(s))
4
5 try:
6     s = {1, [4], 7}
7 except TypeError as e:
8     print(e)
9
10 q = [1, 2, 1, 2, 5, 1]
11 print(q)
12 s = set(q)
13 print(s)
14 s = {5, 2, 1, 2, 2, 1}
15 print(s)
16 print(2 in s)
17 print(3 in s)
18 s2 = {3, 2, 3}
19 print(s | s2)
20 print(s & s2)
21 print(s ^ s2)
22
```



```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_set.py
{1, 4, 7}
<class 'set'>
unhashable type: 'list'
[1, 2, 1, 2, 5, 1]
{1, 2, 5}
{1, 2, 5}
True
False
{1, 2, 3, 5}
{2}
{1, 3, 5}
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$
```

另外，Python 标准库 (standard library) 里的 `pathlib` 和 `datetime` 模块 (module) 提供了用于处理 路径 和 日期时间 的类型，也是非常基础、非常常用的。标准库模块都不需要安装 (pip/conda install)，但使用前需要导入 (import)。

(1) Pathlib:



```
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_path.py
.
True
C:\Users\Administrator\repo\week05
[WindowsPath('.git'),
 WindowsPath('.gitignore'),
 WindowsPath('environment.yml'),
 WindowsPath('LICENSE'),
 WindowsPath('README.md'),
 WindowsPath('use_of_bool.py'),
 WindowsPath('use_of_bytes.py'),
 WindowsPath('use_of_dict.py'),
 WindowsPath('use_of_float.py'),
 WindowsPath('use_of_int.py'),
 WindowsPath('use_of_list.py'),
 WindowsPath('use_of_path.py'),
 WindowsPath('use_of_set.py'),
 WindowsPath('use_of_str.py'),
 WindowsPath('use_of_tuple.py')]
True
True
True
README .md
C:\Users\Administrator\repo\week05\README .md
--Return--
> c:\users\administrator\repo\week05\use_of_path.py(20)<module>()->N
one
-> breakpoint()
(Pdb) █
```

(2) datetime:


```
use_of_datetime.py > ...
1 from datetime import date, datetime, timedelta # noqa: F401
2
3 t1 = date.today()
4 t2 = date(2025, 11, 11)
5 td = t2 - t1
6 print(td)
7 print(type(td))
8 print(td.days)
9
10 s1 = "2024-05-21"
11 s2 = "2024-12-04"
12 d1 = datetime.strptime(s1, "%Y-%m-%d")
13 d2 = datetime.strptime(s2, "%Y-%m-%d")
14 print(d1)
15 print(d2)
16 breakpoint()
17
```

```
(week05)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week05 (main)
$ python use_of_datetime.py
215 days, 0:00:00
<class 'datetime.timedelta'>
215
2024-05-21 00:00:00
2024-12-04 00:00:00
--Return--
> c:\users\administrator\repo\week05\use_of_datetime.py(16)<module>(
)->None
-> breakpoint()
(Pdb)
```

```
(Pdb) p d1
datetime.datetime(2024, 5, 21, 0, 0)
(Pdb) p format(d1, "%a")
'Tue'
(Pdb) p format(d2, "%a")
'Wed'
(Pdb) p d2.strftime("%B")
'December'
(Pdb) p d2
datetime.datetime(2024, 12, 4, 0, 0)
(Pdb) p d2.strftime("%B, %d")
'December, 04'
(Pdb) p d2.strftime("%Y-%m-%d")
'2024-12-04'
(Pdb)
```