

先fork课程创建环境,注意,git clone在repo文件夹下建立week05.

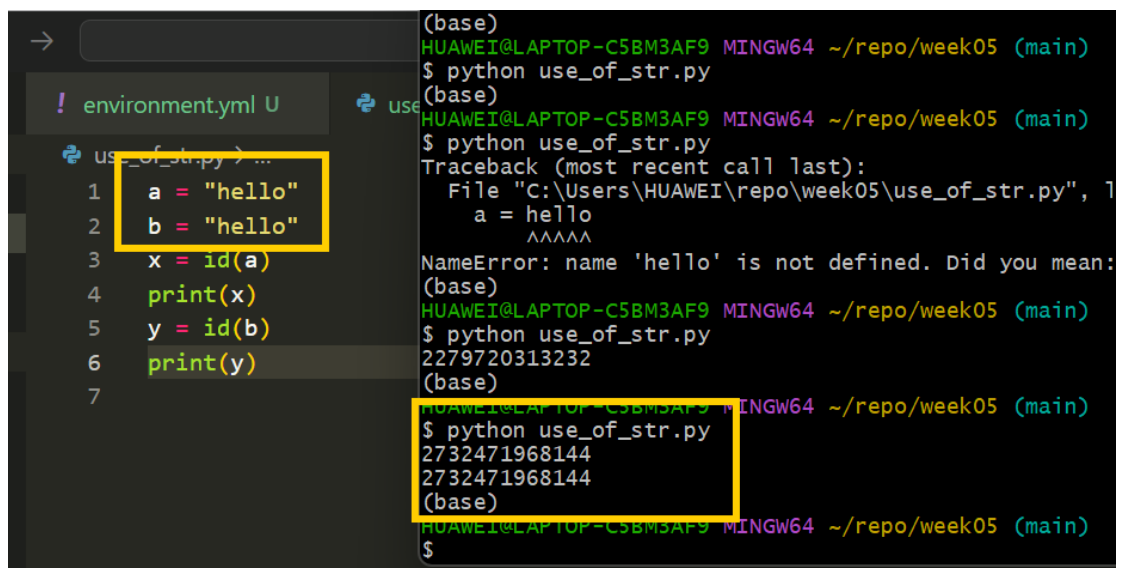
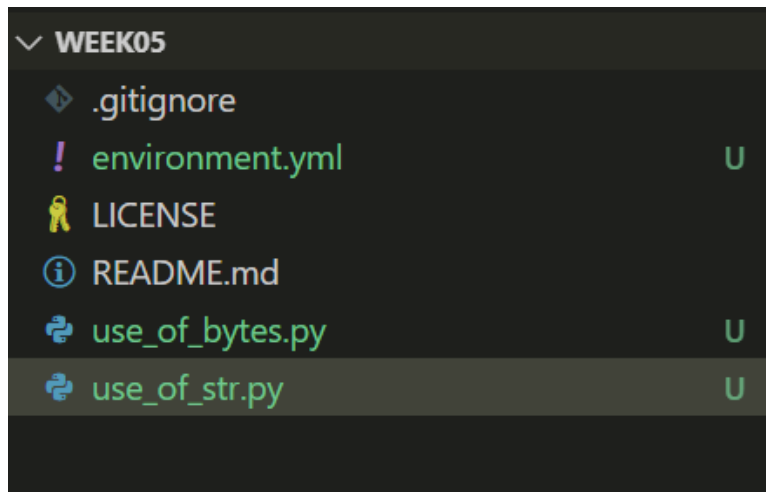
将 environment.yml 环境复制到 week05 下,并用 vscode 打开。

```
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
$ cat week04/environment.yml
cat: week04/environment.yml: No such file or directory
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
$ cat names/environment.yml
name: names
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
$ cp names/environment.yml week05/
(base)
```

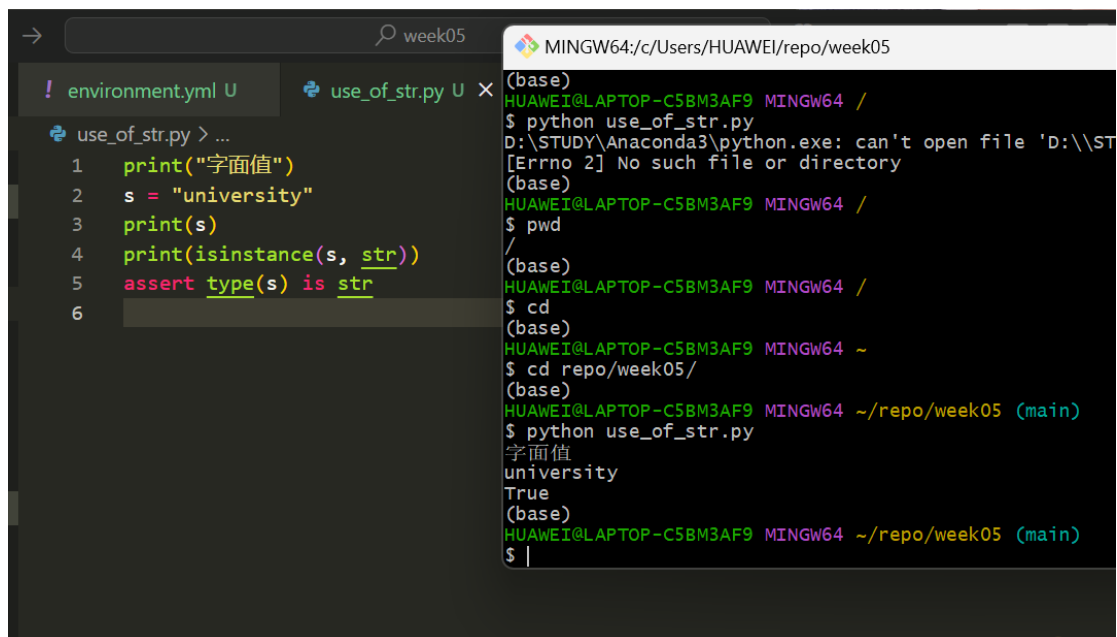
把主目录调整到 week05,可以看得环境文件,然后配置环境。(conda env create)

```
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo
$ cd week05
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ ls -l
total 25
-rw-r--r-- 1 HUAWEI 197609 18805 Apr  6 22:59 LICENSE
-rw-r--r-- 1 HUAWEI 197609  2239 Apr  6 22:59 README.md
-rw-r--r-- 1 HUAWEI 197609   91 Apr  6 23:07 environment.yml
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ conda env create
D:\STUDY\Anaconda3\Lib\argparse.py:2006: FutureWarning: `remote_def
eprecated and will be removed in 25.9. Use `conda env create --file
'
  action(self, namespace, argument_values, option_string)
```

创建两个文件



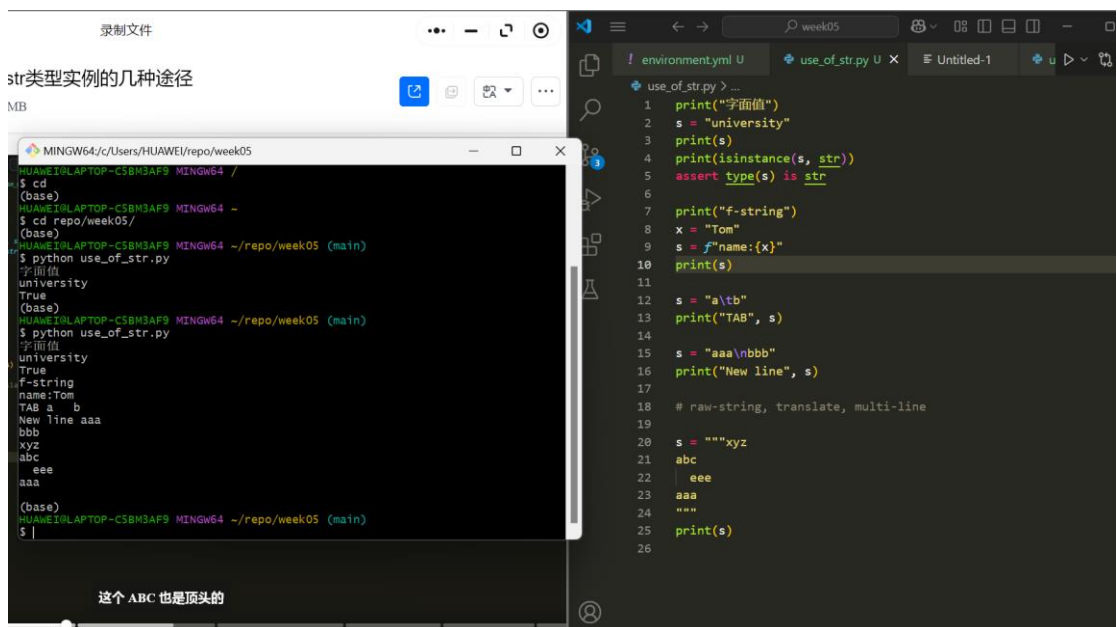
字面值



```
! environment.yml U use_of_str.py U X
use_of_str.py > ...
1 print("字符串")
2 s = "university"
3 print(s)
4 print(isinstance(s, str))
5 assert type(s) is str
6

(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 /
$ python use_of_str.py
D:\STUDY\Anaconda3\python.exe: can't open file 'D:\\ST
[Errno 2] No such file or directory
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 /
$ pwd
/
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 /
$ cd
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~
$ cd repo/week05/
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字符串
university
True
True
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ |
```

继续实践。



```
录制文件
str类型实例的几种途径
MB

MINGW64/c/Users/HUAWEI/repo/week05
$ cd
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~
$ cd repo/week05/
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字符串
university
True
True
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字符串
university
True
True
f-string
name:Tom
TAB a b
New line aaa
bbb
xyz
abc
eee
aaa
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ |

这个 ABC 也是顶头的

! environment.yml U use_of_str.py U X
use_of_str.py > ...
1 print("字符串")
2 s = "university"
3 print(s)
4 print(isinstance(s, str))
5 assert type(s) is str
6
7 print("f-string")
8 x = "Tom"
9 s = f"name:{x}"
10 print(s)
11
12 s = "a\tb"
13 print("TAB", s)
14
15 s = "aaa\nbbb"
16 print("New line", s)
17
18 # raw-string, translate, multi-line
19
20 s = """xyz
21 abc
22 eee
23 aaa
24 """
25 print(s)
26
```

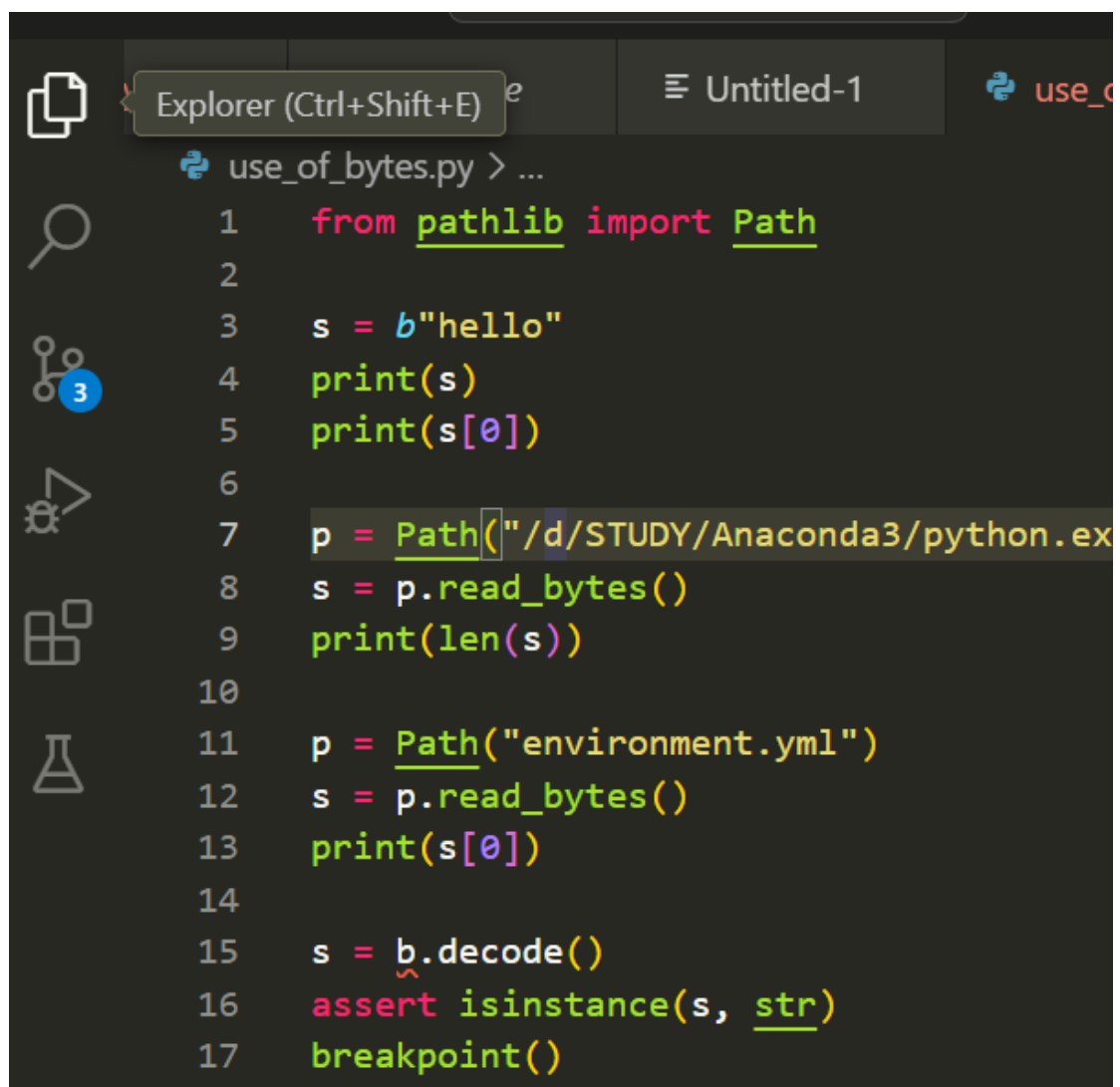
边写边运行，及时查看报错，有些错误可能是写错了或者看错了，如果发现不了可以借助 ai。

补充：规则下字母有排序大小，比如小写字母大于大写字母，老师展示了。

```
print("abc" > "ABC")
print("123" > "abcd")
print("9" > ".")
print("9" < ":")
print("book" < "box")
```

补充了迭代 for 循环，可以一直按 next。

## 二、use\_of\_bytes.py



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'use\_of\_bytes.py' with three files. The code editor shows the following code:

```
1  from pathlib import Path
2
3  s = b"hello"
4  print(s)
5  print(s[0])
6
7  p = Path("/d/STUDY/Anaconda3/python.exe")
8  s = p.read_bytes()
9  print(len(s))
10
11 p = Path("environment.yml")
12 s = p.read_bytes()
13 print(s[0])
14
15 s = b.decode()
16 assert isinstance(s, str)
17 breakpoint()
```

### 三、use\_of\_int.py

### 四、use\_of\_float.py

演示-part5-float~dict等类型

操作演示-part5-float~dict等类型

你来说我能不能得到其他的随机数

```
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
Traceback (most recent call last):
  File "C:\Users\HUAWEI\repo\week05\use_of_float.py", line 5, in <module>
    print(type(y))
    ^
NameError: name 'y' is not defined
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.25178027245733214
(base)
HUAWEI@LAPTOP-C5BM3AF9 MINGW64 ~/repo/week05 (main)
$
```

接上文继续...



```
← → week05 use_of_list.py 1, U use_of_float.py 1, U use_of_bool.py U
use_of_list.py > ...
1  l = [1, 5, "abc"]
2  print(1)
3
4  print(1[0])
5  print(1[1])
6  print(1[2])
7
8  try:
9      print(1[3])
10 except IndexError as e:
11     print(e)
12
13 print(1[-1])
14 print(1[-1][1])
15
16
17 a = [2, 5]
18 b = ["a", "c"]
19 print(a + b)
20 print(b + a)
21 print(a + b == b + a)
22
23 a = [2, 5]
24 b = [5]
25 try:
26     print(a - b)
27 except TypeError as e:
28     print(e)
29
30 a = [2, 5]
31 b = a * 3
32 a[0] = 9
33 print(a)
34 print(b)
```

```
a = [2, 5]
b = ["a", "c"]
print(a + b)
print(b + a)

a = [2, 5]
b = [1]
try:
    print(a - b)
except TypeError as e:
    print(e)

a = [2, 5]
print(a * 3)

a = [2, 5]
b = a * 3
print(f"{b}")
a[0] = 9
print(a)
print(b)

a = [2, 5]
b = [a] * 3
print(f"{b}")
a[0] = 9
print(a)
print(b)

$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
unsupported operand type(s) for -: 'list' and 'list'
[2, 9, 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]]
(week05)
$ g++3x64 CLANGARM64 ~/repo/week05 (main)
$

29
30 a = [2, 5]
31 b = a * 3
32
33
34 a = [2, 5]
35 b = a * 3
36 print(f"{b}")
37 a[0] = 9
38 print(a)
39 print(b)
40
41
42 a = [2, 5]
43 b = [a] * 3
44 print(f"{b}")
45 a[0] = 9
46 print(a)
47 print(b)
```

你要知道这些对你要有这个概念

## 六、use\_of\_dict.py

```
use_of_dict.py 4, U × use_of_float.py 1, U use_of_bool.py U ▶ ▾ 🔍 □

use_of_dict.py > ...
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
8 for a in d:
9     print(d[a])
10
11 for a in d.values():
12     print(a)
13
14 l = [a for a in d.items()]
15 print(l)
16
17 for k,v in d.items():
18     print(k,v)
19
20 breakpoint()
21
```



```

    print(a - b)
except TypeError as e:
    print(e)

a = [2, 5]
b = a * 3

a = [2, 5]
b = a * 3
print(f"(b={b})")
a[0] = 9
print(a)
print(b)

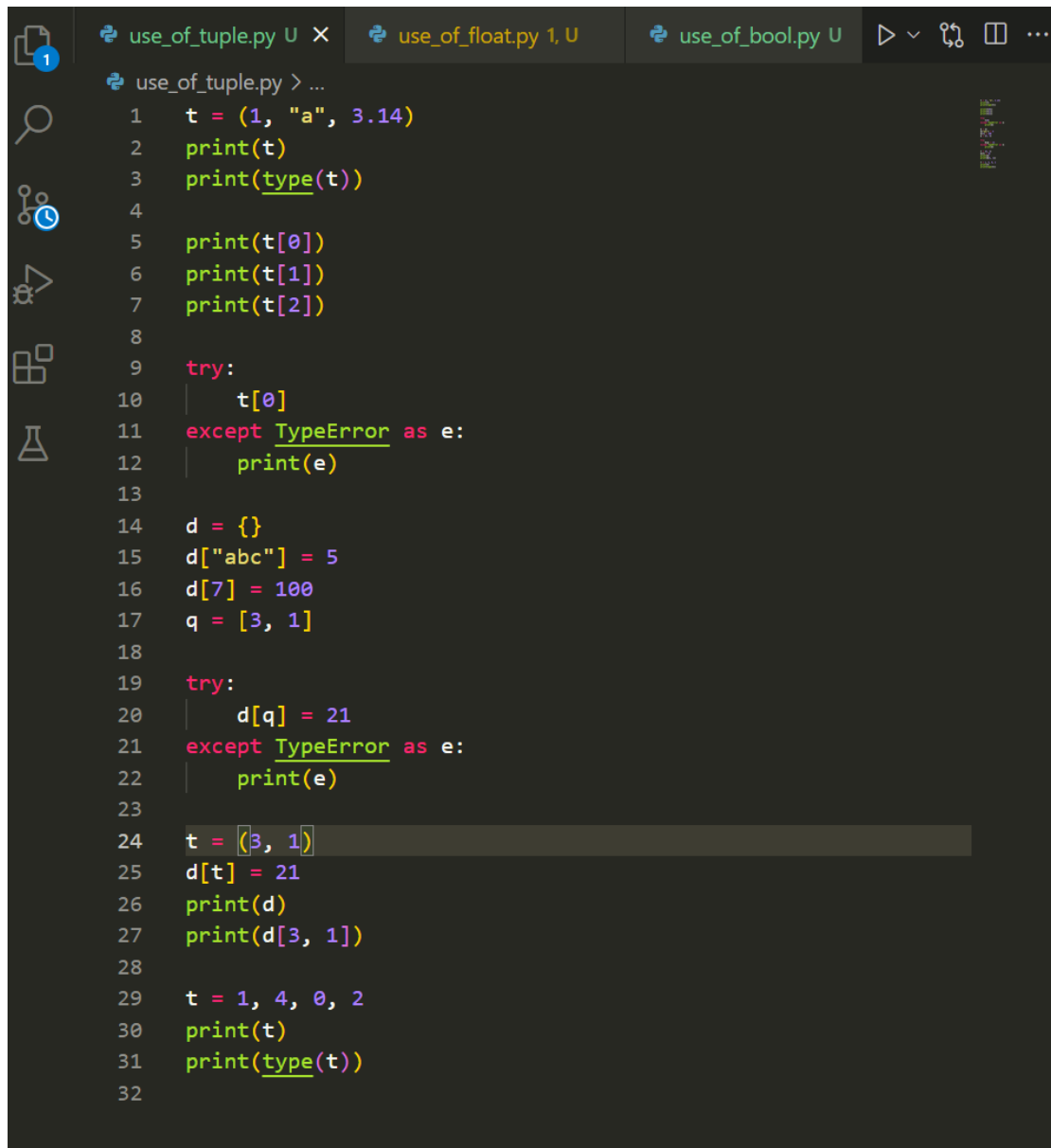
a = [2, 5]
b = [a] * 3
print("(b=)")
a[0] = 9
print(a)
print(b)

a = [2, 5, 3]
b = [1**2 for i in a]
print(b)
b = [i**2 for i in a if i < 4]
print(b)

a = [2,5]
b = [a] * 3
print(f"(B=)")
x = a.append(4)
print(a)
print(b)
breakpoint()

```

七、use\_of\_tuple.py



The image shows a Python IDE with three tabs: `use_of_tuple.py`, `use_of_float.py`, and `use_of_bool.py`. The `use_of_tuple.py` tab is active, displaying the following code:

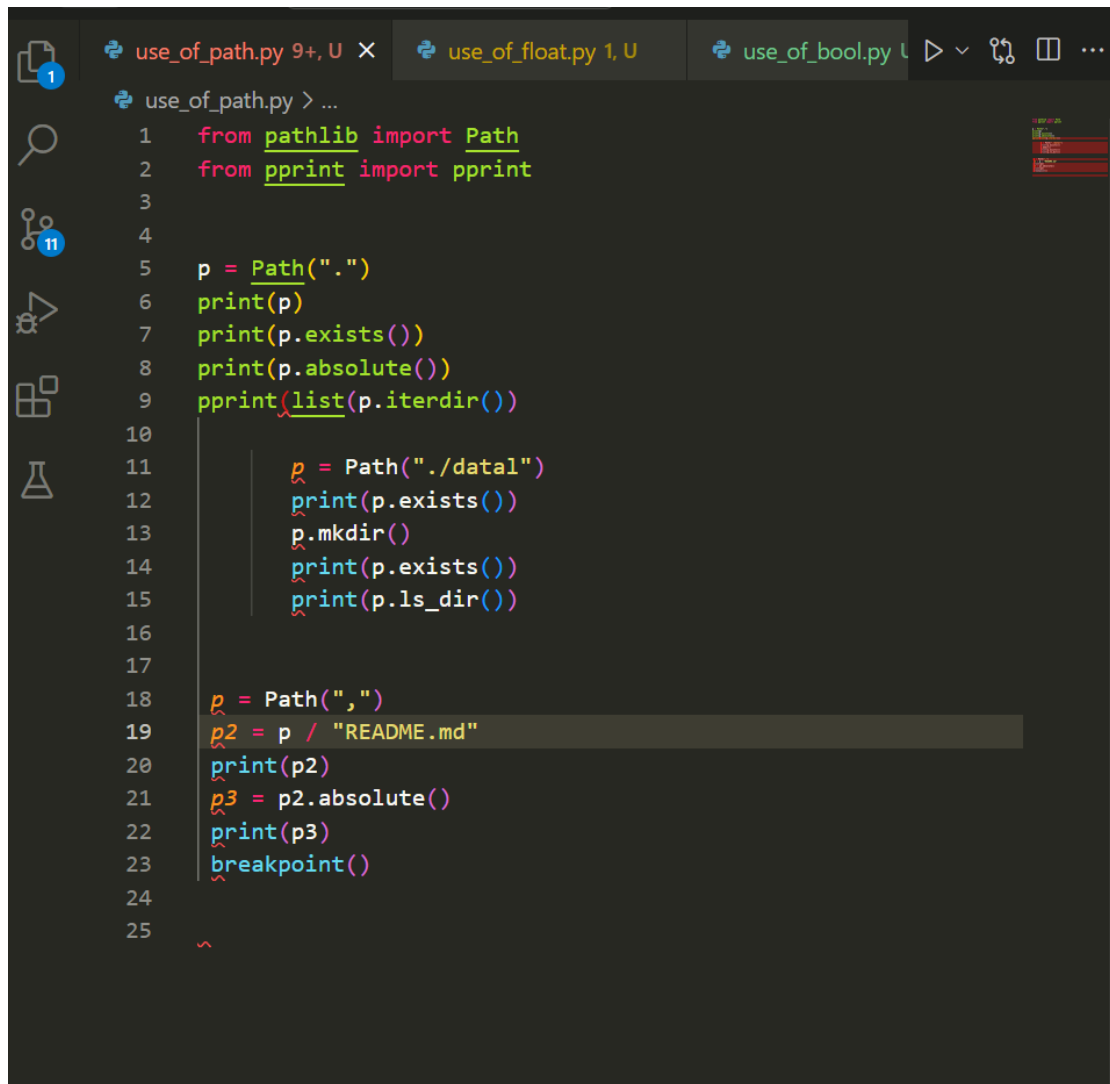
```
1  t = (1, "a", 3.14)
2  print(t)
3  print(type(t))
4
5  print(t[0])
6  print(t[1])
7  print(t[2])
8
9  try:
10     t[0]
11 except TypeError as e:
12     print(e)
13
14 d = {}
15 d["abc"] = 5
16 d[7] = 100
17 q = [3, 1]
18
19 try:
20     d[q] = 21
21 except TypeError as e:
22     print(e)
23
24 t = (3, 1)
25 d[t] = 21
26 print(d)
27 print(d[3, 1])
28
29 t = 1, 4, 0, 2
30 print(t)
31 print(type(t))
32
```

八、use\_of\_set.py



```
use_of_set.py > ...
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
15 s = {5, 2, 1, 2, 2, 1}
16 print(s)
17 print(2 in s)
18 print(3 in s)
19
20 s2 = {3, 2, 3}
21 print(s | s2)
22 print(s & s2)
23
```

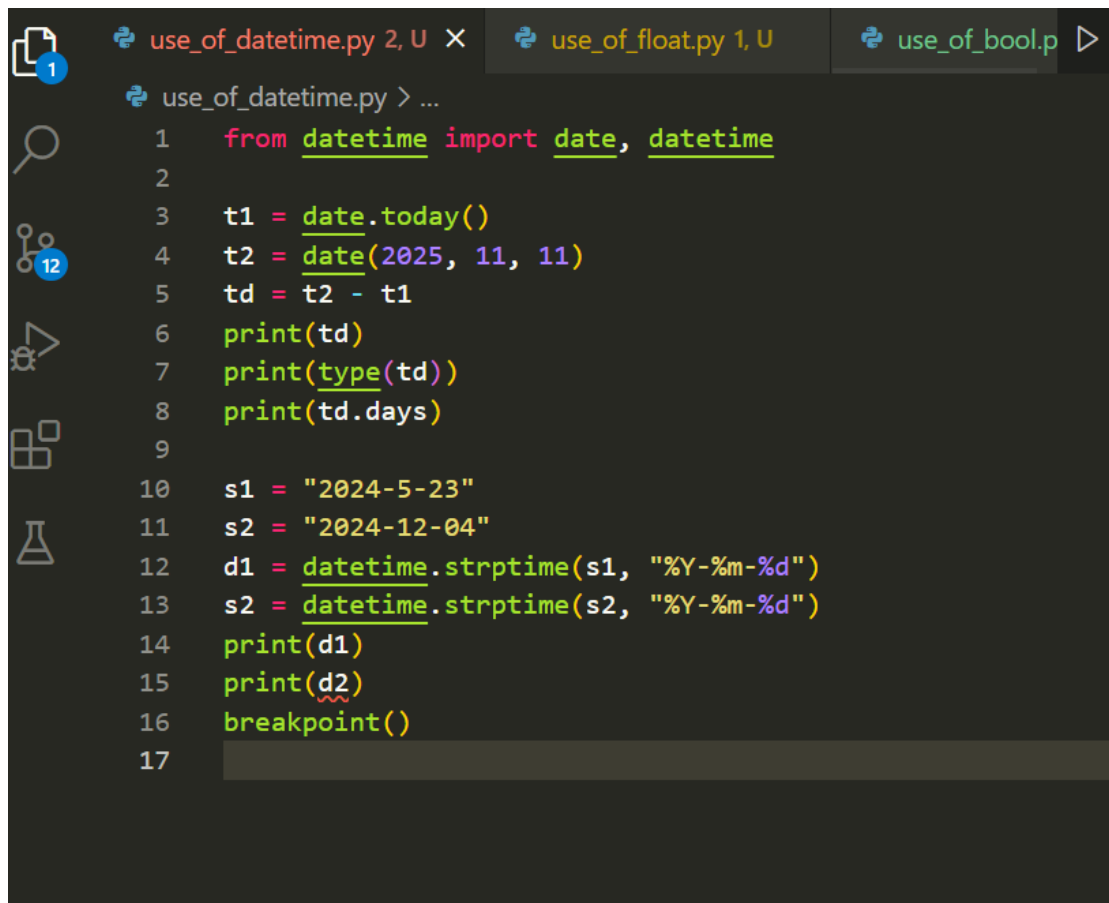
九、use\_of\_path.py



The image shows a code editor with three tabs: 'use\_of\_path.py 9+, U', 'use\_of\_float.py 1, U', and 'use\_of\_bool.py'. The active tab is 'use\_of\_path.py'. The code in the editor is as follows:

```
1 from pathlib import Path
2 from pprint import pprint
3
4
5 p = Path(".")
6 print(p)
7 print(p.exists())
8 print(p.absolute())
9 pprint(list(p.iterdir()))
10
11     p = Path("./data1")
12     print(p.exists())
13     p.mkdir()
14     print(p.exists())
15     print(p.ls_dir())
16
17
18 p = Path(",")
19 p2 = p / "README.md"
20 print(p2)
21 p3 = p2.absolute()
22 print(p3)
23 breakpoint()
24
25
```

十、use\_of\_datetime.py



```
1  from datetime import date, datetime
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 = "2024-5-23"
11 s2 = "2024-12-04"
12 d1 = datetime.strptime(s1, "%Y-%m-%d")
13 s2 = datetime.strptime(s2, "%Y-%m-%d")
14 print(d1)
15 print(d2)
16 breakpoint()
17
```