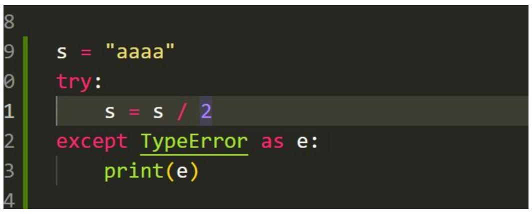
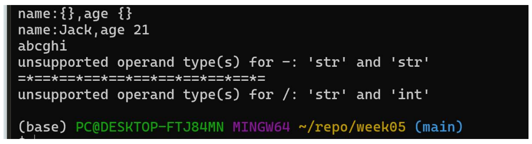
金融计算机第五周作业

——Part3&4&5&6

三、str 类型支持的各种操作和方法

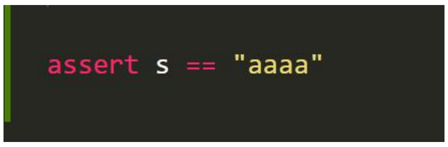

```python
s = "aaaa"
try:
    s = s / 2
except TypeError as e:
    print(e)
```


```
name:{},age {}
name:Jack,age 21
abcghi
unsupported operand type(s) for -: 'str' and 'str'
=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
unsupported operand type(s) for /: 'str' and 'int'

(base) PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
```

运行发现对于字符串的出发类型是不支持的

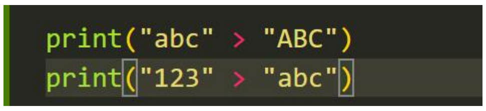
1.验证属性

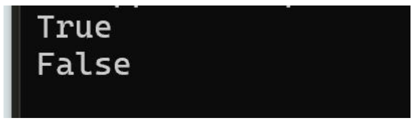
1）判断是否支持相等（==）


```python
assert s == "aaaa"
```

增加一行代码，最终结果没有变化，即未报错，则说明是相等的

2）比较运算符


```python
print("abc" > "ABC")
print("123" > "abc")
```


```
True
False
```

字符串也可以比较大小，支持比较运算符

ASCII 编码字符排序

数字优先： 0 - 9 的数字在字符排序中优先，它们按照从小到大的顺序排列，即 0 在前，9 在后。这是因为在 ASCII 编码中，数字字符的编码值是连续且递增的。

大写字母其次： 在数字之后，是大写字母 A - Z 的排列。同样，它们在 ASCII 编码中也是连续且按照字母表顺序递增的，例如 A 的编码值小于 B 的编码值，所以 A 排在 B 前面。

小写字母最后： 小写字母 a - z 排在最后，也是按照字母表顺序依次排列。在 ASCII 编码中，小写字母的编码值大于大写字母和数字。例如，'a' 的编码值大于 'Z' 的编码值。

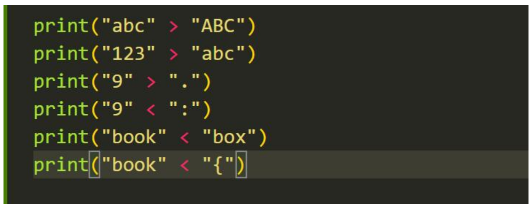
字典序

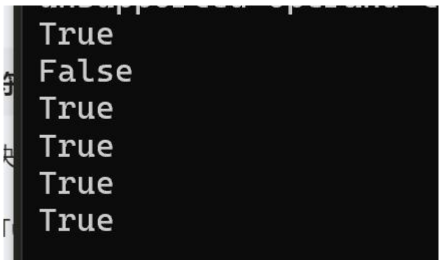
也称为字典顺序或词典序，是一种对字符序列进行排序的方法，类似于字典中单词的排列方式。

对于两个字符序列，从左到右依次比较对应位置的字符。在比较时，依据字符的编码值（如 ASCII 码或 Unicode 码）来确定字符的大小。

当遇到第一个不同的字符时，具有较小编码值的字符所在的序列被认为是较小的序列，即排在前面。

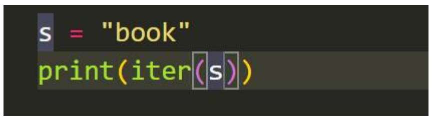
如果一个序列是另一个序列的前缀，那么较短的序列排在前面。

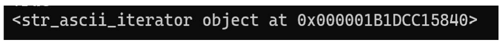

```python
print("abc" > "ABC")
print("123" > "abc")
print("9" > ".")
print("9" < ":")
print("book" < "box")
print("book" < "{")
```


```
True
False
True
True
True
True
```

一些进一步的例子

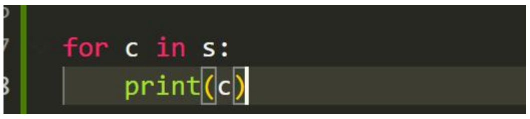
3）是否可迭代


```python
s = "book"
print(iter(s))
```

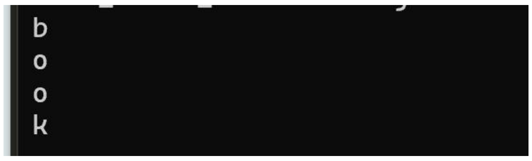

```
<str_ascii_iterator object at 0x000001B1DCC15840>
```

使用 **iter()**可以看是否可以迭代

使用 for 循环进行迭代

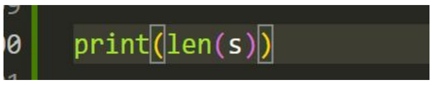


```
for c in s:
    print(c)
```

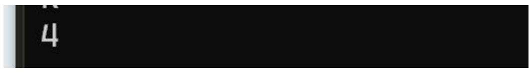


```
b
o
o
k
```

4）是否支持返回长度



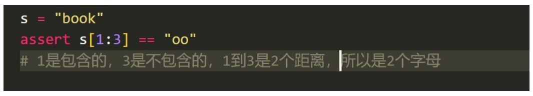
```
print(len(s))
```



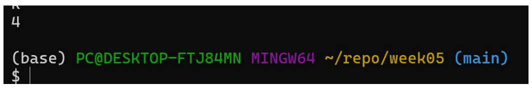
```
4
```

字符串有长度的概念

5）如何支持索引操作



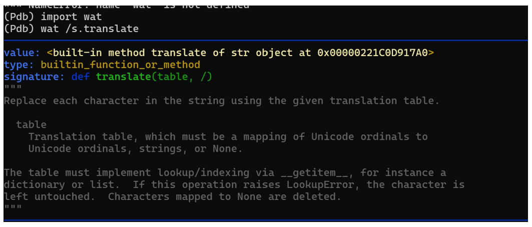
```
s = "book"
assert s[1:3] == "oo"
# 1是包含的，3是不包含的，1到3是2个距离，所以是2个字母
```



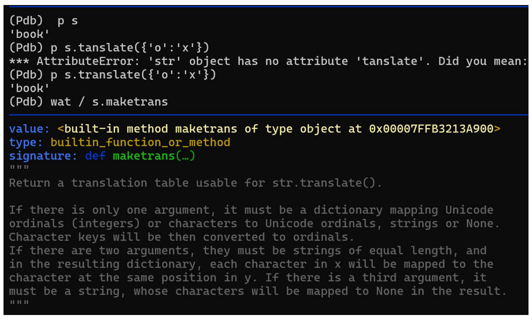
```
4

(base) PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$
```

最后未报错，说明代码中内容正确

6）



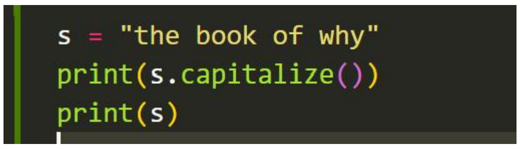
```
(Pdb) import wat
(Pdb) wat /s.translate

value: <built-in method translate of str object at 0x00000221C0D917A0>
type: builtin_function_or_method
signature: def translate(table, /)
"""
Replace each character in the string using the given translation table.

  table
    Translation table, which must be a mapping of Unicode ordinals to
    Unicode ordinals, strings, or None.

The table must implement lookup/indexing via __getitem__, for instance a
dictionary or list.  If this operation raises LookupError, the character is
left untouched.  Characters mapped to None are deleted.
"""
```



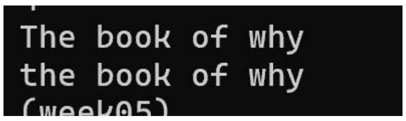
```
(Pdb)  p s
'book'
(Pdb) p s.tanslate({'o':'x'})
*** AttributeError: 'str' object has no attribute 'tanslate'. Did you mean:
(Pdb) p s.translate({'o':'x'})
'book'
(Pdb) wat / s.maketrans

value: <built-in method maketrans of type object at 0x00007FFB3213A900>
type: builtin_function_or_method
signature: def maketrans(…)
"""
Return a translation table usable for str.translate().

If there is only one argument, it must be a dictionary mapping Unicode
ordinals (integers) or characters to Unicode ordinals, strings or None.
Character keys will be then converted to ordinals.
If there are two arguments, they must be strings of equal length, and
in the resulting dictionary, each character in x will be mapped to the
character at the same position in y. If there is a third argument, it
must be a string, whose characters will be mapped to None in the result.
"""
```

常用的是

①capitalize--字符串的第一个字母大写

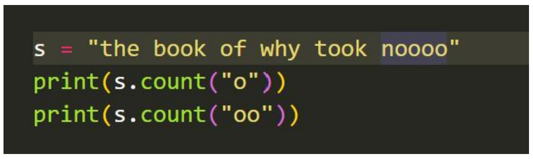


```
s = "the book of why"
print(s.capitalize())
print(s)
```

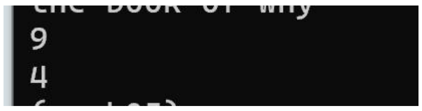


```
The book of why
the book of why
```

②Count 的作用：



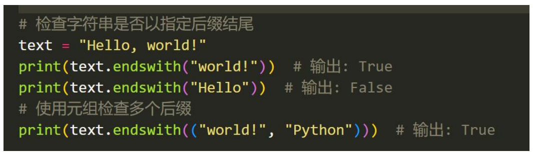
```
s = "the book of why took noooo"
print(s.count("o"))
print(s.count("oo"))
```



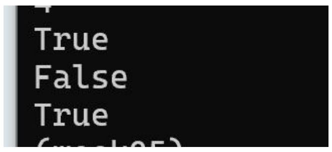
```
9
4
```

③s.endwith

endswith 是字符串对象的内置方法，它可以判断字符串是否以指定的后缀结束，同时还能指定开始和结束的索引范围



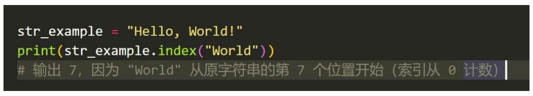
```
# 检查字符串是否以指定后缀结尾
text = "Hello, world!"
print(text.endswith("world!"))  # 输出: True
print(text.endswith("Hello"))  # 输出: False
# 使用元组检查多个后缀
print(text.endswith(("world!", "Python")))  # 输出: True
```



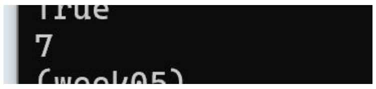
```
True
False
True
```

④s.index

index 是一个常用的方法，可用于字符串、列表、元组等序列类型，用于查找某个元素首次出现的索引位置



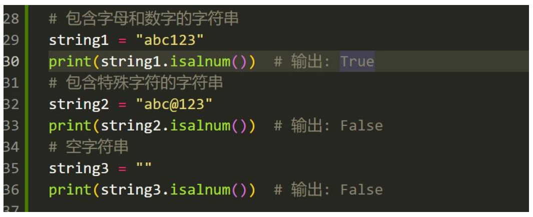
```
str_example = "Hello, World!"
print(str_example.index("World"))
# 输出 7，因为 "World" 从原字符串的第 7 个位置开始（索引从 0 计数）
```



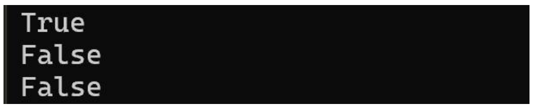
```
7
```

⑤s.isalnum

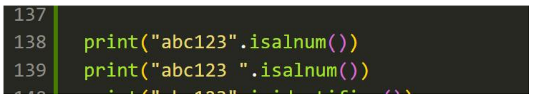
isalnum 是字符串对象的一个内置方法。如果字符串中的所有字符都是字母（大写或小写）或者数字，并且字符串至少有一个字符，那么该方法返回 True；否则返回 False



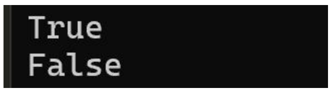
```
# 包含字母和数字的字符串
string1 = "abc123"
print(string1.isalnum())  # 输出: True
# 包含特殊字符的字符串
string2 = "abc@123"
print(string2.isalnum())  # 输出: False
# 空字符串
string3 = ""
print(string3.isalnum())  # 输出: False
```



```
True
False
False
```



```
print("abc123".isalnum())
print("abc123 ".isalnum())
```


```
True
False
```

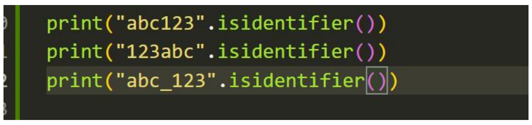
⑥s.isidentifier

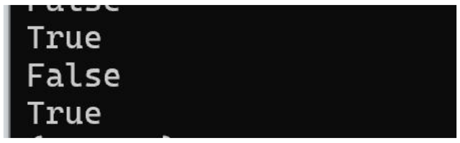
isidentifier 是 Python 字符串对象的一个内置方法，用于判断一个字符串是否是有效的 Python 标识符。

• 标识符只能由字母（大写或小写）、数字和下划线_组成。

• 标识符不能以数字开头。

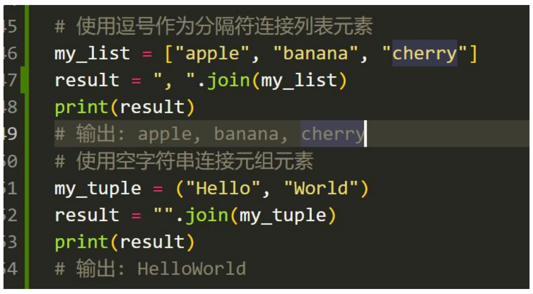
• 标识符不能是 Python 的关键字（如 if、else、for 等）

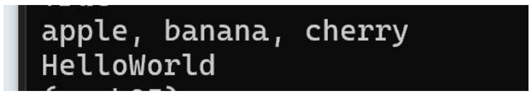

```python
print("abc123".isidentifier())
print("123abc".isidentifier())
print("abc_123".isidentifier())
```


```
True
False
True
```

⑦s.join

join 是数组对象的方法，它将数组中的所有元素连接成一个字符串，并返回该字符串。元素之间用指定的分隔符分隔，如果省略分隔符，默认使用逗号


```python
# 使用逗号作为分隔符连接列表元素
my_list = ["apple", "banana", "cherry"]
result = ", ".join(my_list)
print(result)
# 输出: apple, banana, cherry
# 使用空字符串连接元组元素
my_tuple = ("Hello", "World")
result = "".join(my_tuple)
print(result)
# 输出: HelloWorld
```


```
apple, banana, cherry
HelloWorld
```

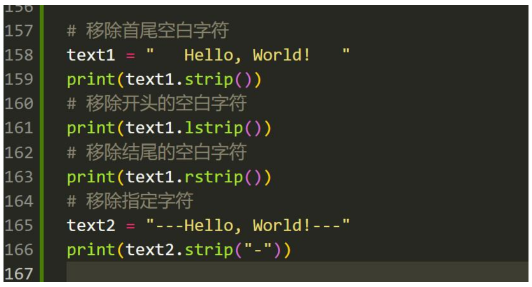
⑧s.strip

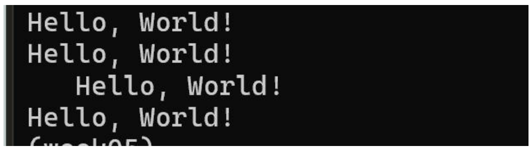
在 Python 里，strip 是字符串对象的内置方法，它有三种形式，分别是 strip()、lstrip() 和 rstrip()。

• strip()：用于移除字符串首尾的指定字符，若未指定字符，默认移除空白字符。

• lstrip()：专门移除字符串左侧（开头）的指定字符，默认移除空白字符。

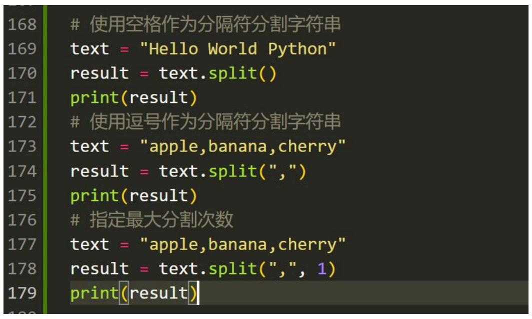
• rstrip()：用于移除字符串右侧（结尾）的指定字符，默认移除空白字符。

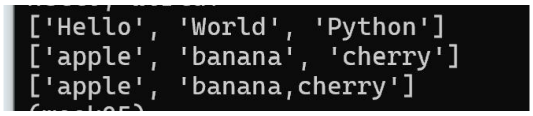

```python
# 移除首尾空白字符
text1 = "   Hello, World!   "
print(text1.strip())
# 移除开头的空白字符
print(text1.lstrip())
# 移除结尾的空白字符
print(text1.rstrip())
# 移除指定字符
text2 = "---Hello, World!---"
print(text2.strip("-"))
```


```
Hello, World!
Hello, World!
    Hello, World!
Hello, World!
```

⑨s.split

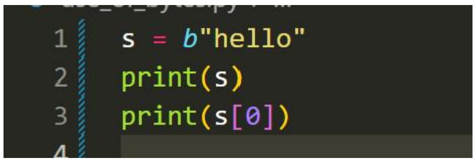
split 是字符串对象的内置方法，它可以根据指定的分隔符将字符串分割成多个子字符串，并返回一个列表。如果不指定分隔符，默认会以空白字符（如空格、制表符、换行符等）作为分隔符

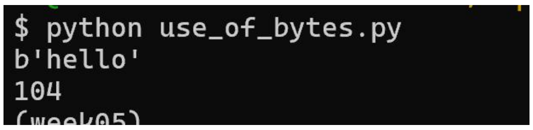

```python
# 使用空格作为分隔符分割字符串
text = "Hello World Python"
result = text.split()
print(result)
# 使用逗号作为分隔符分割字符串
text = "apple,banana,cherry"
result = text.split(",")
print(result)
# 指定最大分割次数
text = "apple,banana,cherry"
result = text.split(",", 1)
print(result)
```

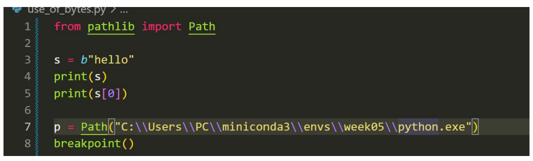

```
['Hello', 'World', 'Python']
['apple', 'banana', 'cherry']
['apple', 'banana,cherry']
```

四、bytes 编解码和 int 整数

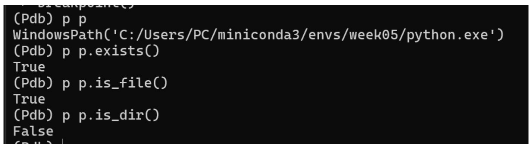
1.bytes


```python
s = b"hello"
print(s)
print(s[0])
```

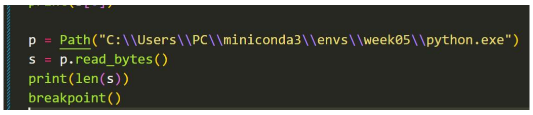

```
$ python use_of_bytes.py
b'hello'
104
(week05)
```


```python
from pathlib import Path

s = b"hello"
print(s)
print(s[0])

p = Path("C:\\Users\\PC\\miniconda3\\envs\\week05\\python.exe")
breakpoint()
```

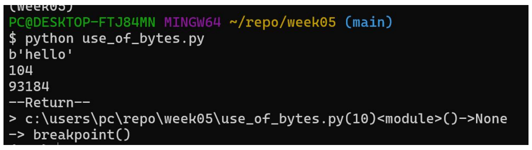
Tips：路径撰写，如果是前面写 r，那么就是单个反斜杠，否则双反斜杠，或写为：/c/Users/PC/miniconda3/envs/week05/python



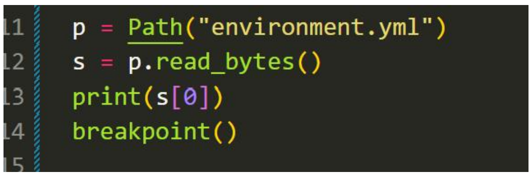
```
(Pdb) p p
WindowsPath('C:/Users/PC/miniconda3/envs/week05/python.exe')
(Pdb) p p.exists()
True
(Pdb) p p.is_file()
True
(Pdb) p p.is_dir()
False
```



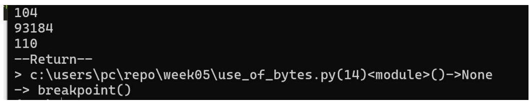
```
p = Path("C:\\Users\\PC\\miniconda3\\envs\\week05\\python.exe")
s = p.read_bytes()
print(len(s))
breakpoint()
```



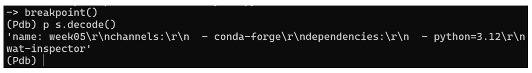
```
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'hello'
104
93184
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(10)<module>()->None
-> breakpoint()
```



```
p = Path("environment.yml")
s = p.read_bytes()
print(s[0])
breakpoint()
```



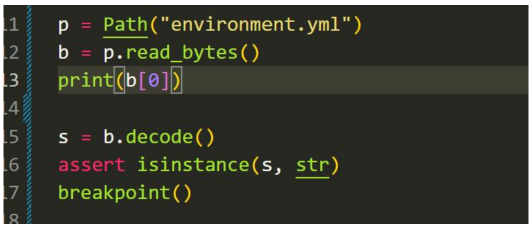
```
104
93184
110
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(14)<module>()->None
-> breakpoint()
```



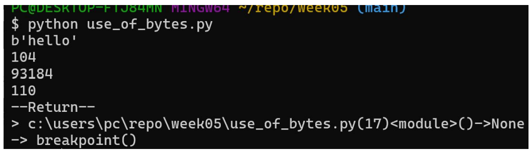
```
-> breakpoint()
(Pdb) p s.decode()
'name: week05\r\nchannels:\r\n  - conda-forge\r\ndependencies:\r\n  - python=3.12\r\n
wat-inspector'
(Pdb)
```

s.decode()

是解码的代码，解码后变为字符串

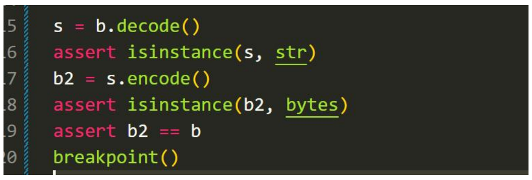


```
p = Path("environment.yml")
b = p.read_bytes()
print(b[0])

s = b.decode()
assert isinstance(s, str)
breakpoint()
```



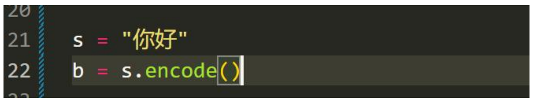
```
$ python use_of_bytes.py
b'hello'
104
93184
110
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(17)<module>()->None
-> breakpoint()
```

没有报错说明 assert 那一行为正确的

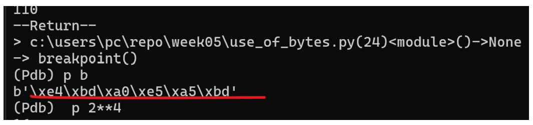


```
s = b.decode()
assert isinstance(s, str)
b2 = s.encode()
assert isinstance(b2, bytes)
assert b2 == b
breakpoint()
```

运行结果仍未报错，说明代码内容为正确的

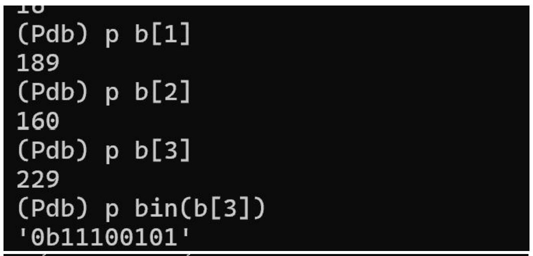


```
s = "你好"
b = s.encode()
```

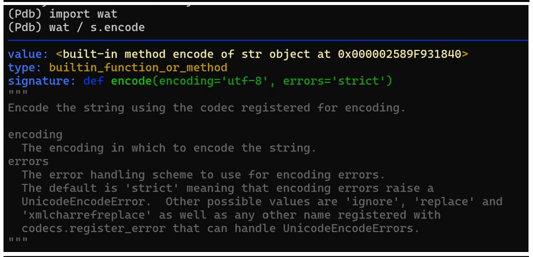


```
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(24)<module>()->None
-> breakpoint()
(Pdb) p b
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb)  p 2**4
```

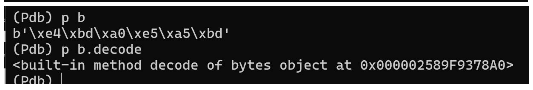
b 的结果如图画红线部分所示



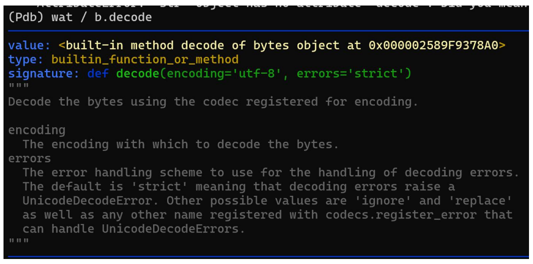
```
(Pdb) p b[1]
189
(Pdb) p b[2]
160
(Pdb) p b[3]
229
(Pdb) p bin(b[3])
'0b11100101'
```



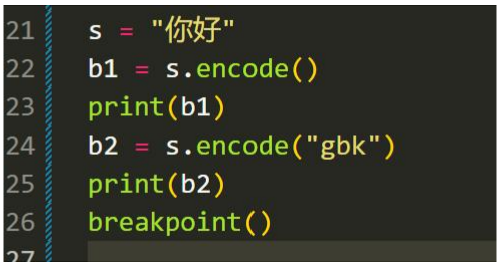
```
(Pdb) import wat
(Pdb) wat / s.encode

value: <built-in method encode of str object at 0x000002589F931840>
type: builtin_function_or_method
signature: def encode(encoding='utf-8', errors='strict')
"""
Encode the string using the codec registered for encoding.

encoding
  The encoding in which to encode the string.
errors
  The error handling scheme to use for encoding errors.
  The default is 'strict' meaning that encoding errors raise a
  UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
  'xmlcharrefreplace' as well as any other name registered with
  codecs.register_error that can handle UnicodeEncodeErrors.
"""
```



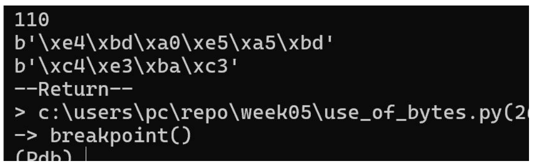
```
(Pdb) p b
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb) p b.decode
<built-in method decode of bytes object at 0x000002589F9378A0>
(Pdb)
```



```
(Pdb) wat / b.decode

value: <built-in method decode of bytes object at 0x000002589F9378A0>
type: builtin_function_or_method
signature: def decode(encoding='utf-8', errors='strict')
"""
Decode the bytes using the codec registered for encoding.

encoding
  The encoding with which to decode the bytes.
errors
  The error handling scheme to use for the handling of decoding errors.
  The default is 'strict' meaning that decoding errors raise a
  UnicodeDecodeError. Other possible values are 'ignore' and 'replace'
  as well as any other name registered with codecs.register_error that
  can handle UnicodeDecodeErrors.
"""
```

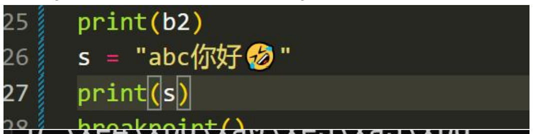


```
s = "你好"
b1 = s.encode()
print(b1)
b2 = s.encode("gbk")
print(b2)
breakpoint()
```

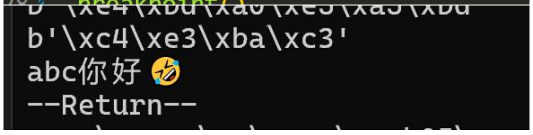


```
110
b'\xe4\xbd\xa0\xe5\xa5\xbd'
b'\xc4\xe3\xba\xc3'
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(2
-> breakpoint()
```

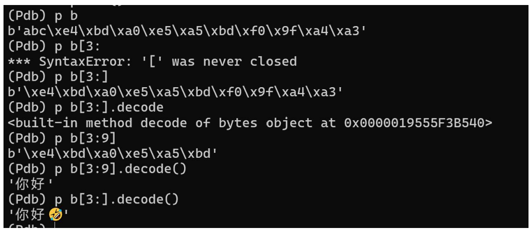
Emoji 打印出来还是 emoji 没有变化



```
print(b2)
s = "abc你好🤣"
print(s)
```



```
b'\xc4\xe3\xba\xc3'
abc你好🤣
--Return--
```



```
(Pdb) p b
b'abc\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\xa4\xa3'
(Pdb) p b[3:
*** SyntaxError: '[' was never closed
(Pdb) p b[3:]
b'\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\xa4\xa3'
(Pdb) p b[3:].decode
<built-in method decode of bytes object at 0x0000019555F3B540>
(Pdb) p b[3:9]
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb) p b[3:9].decode()
'你好'
(Pdb) p b[3:].decode()
'你好🤣'
```

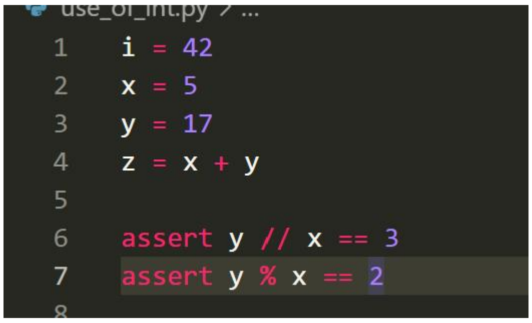
通过解码得到结果

Decode 是解码

Encode 是编码

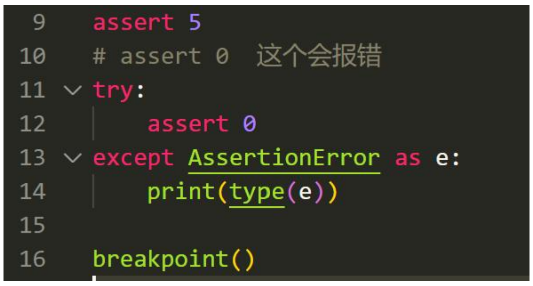
总称为：编解码

2.int 整数

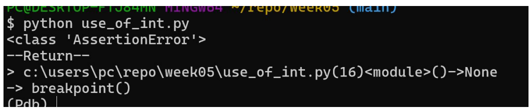


```python
i = 42
x = 5
y = 17
z = x + y

assert y // x == 3
assert y % x == 2
```

运行结果未产生报错



```python
assert 5
# assert 0   这个会报错
try:
    assert 0
except AssertionError as e:
    print(type(e))

breakpoint()
```

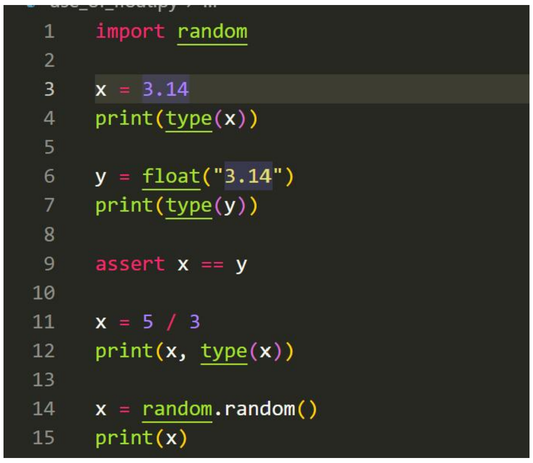


```
$ python use_of_int.py
<class 'AssertionError'>
--Return--
> c:\users\pc\repo\week05\use_of_int.py(16)<module>()->None
-> breakpoint()
```

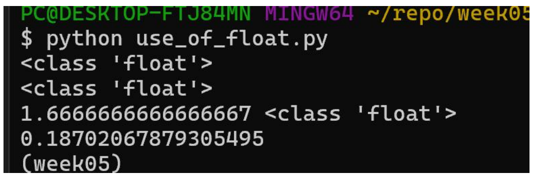
整数不能循环迭代

五、float ~ dict 等类型

1.float 浮点数

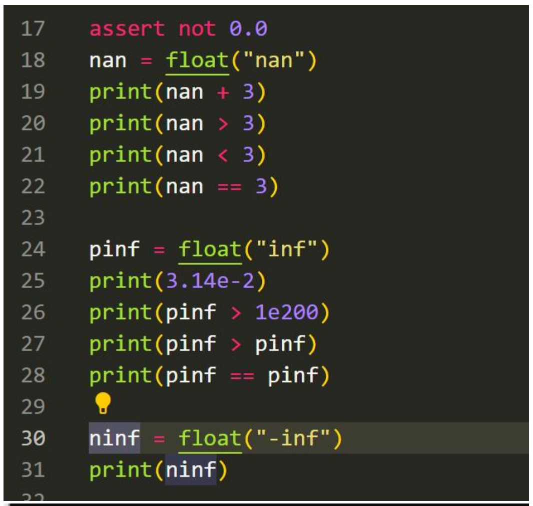


```python
import random

x = 3.14
print(type(x))

y = float("3.14")
print(type(y))

assert x == y

x = 5 / 3
print(x, type(x))

x = random.random()
print(x)
```

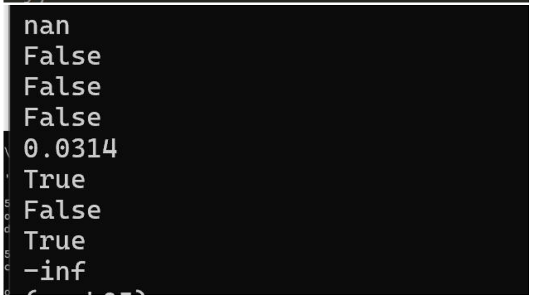


```
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.18702067879305495
(week05)
```

Pinf 是正无穷，ninf 是负无穷

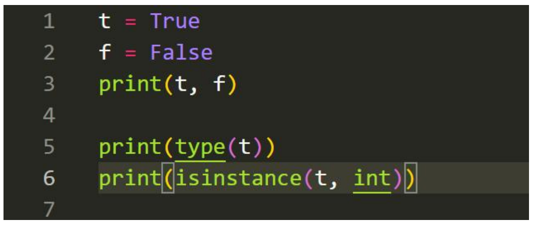


```python
assert not 0.0
nan = float("nan")
print(nan + 3)
print(nan > 3)
print(nan < 3)
print(nan == 3)

pinf = float("inf")
print(3.14e-2)
print(pinf > 1e200)
print(pinf > pinf)
print(pinf == pinf)

ninf = float("-inf")
print(ninf)
```

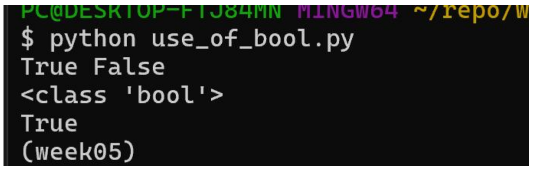


```
nan
False
False
False
0.0314
True
False
True
-inf
```

2.bool 布尔值

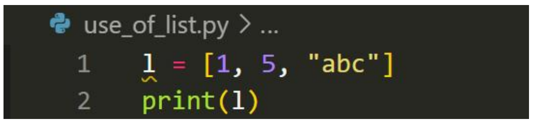


```python
t = True
f = False
print(t, f)

print(type(t))
print(isinstance(t, int))
```

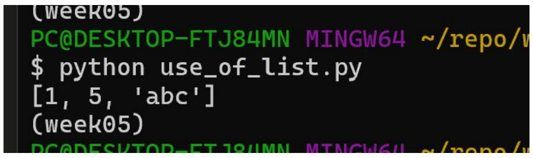


```
$ python use_of_bool.py
True False
<class 'bool'>
True
(week05)
```

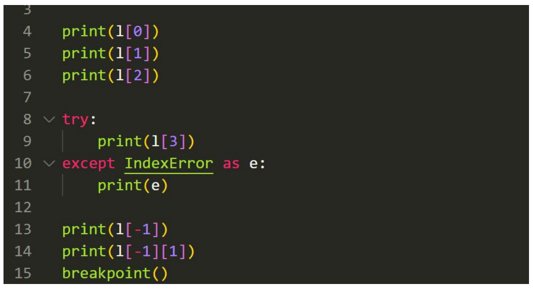
3.list 列表



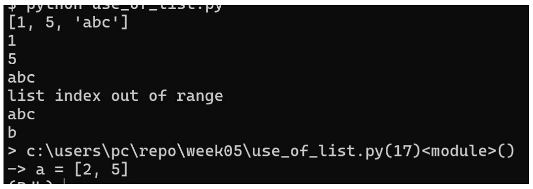
```python
l = [1, 5, "abc"]
print(l)
```



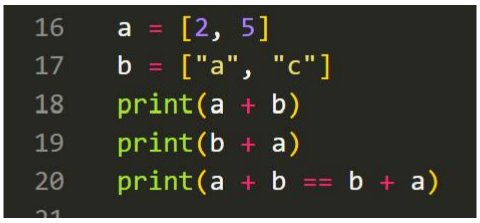
```
(week05)
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/w
$ python use_of_list.py
[1, 5, 'abc']
(week05)
```



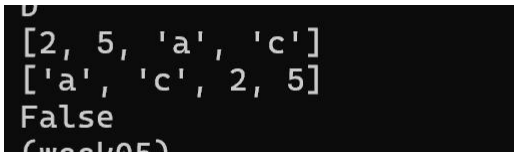
```python
print(l[0])
print(l[1])
print(l[2])

try:
    print(l[3])
except IndexError as e:
    print(e)

print(l[-1])
print(l[-1][1])
breakpoint()
```



```
$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
b
> c:\users\pc\repo\week05\use_of_list.py(17)<module>()
-> a = [2, 5]
```

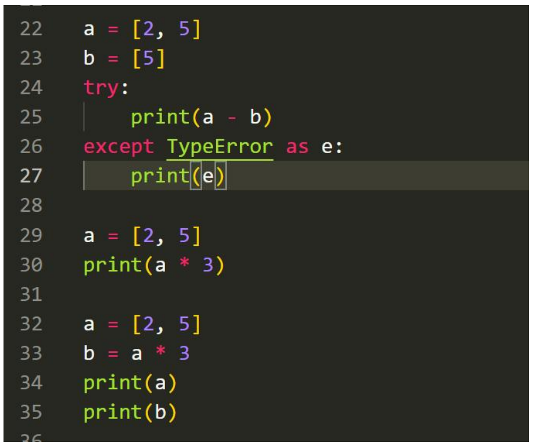


```python
a = [2, 5]
b = ["a", "c"]
print(a + b)
print(b + a)
print(a + b == b + a)
```

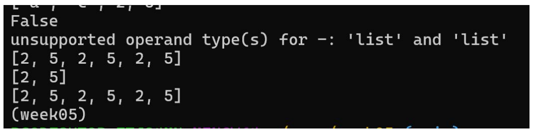


```
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
```

列表没有加法交换律

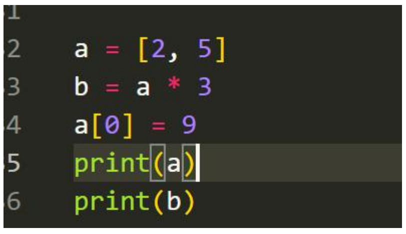


```python
a = [2, 5]
b = [5]
try:
    print(a - b)
except TypeError as e:
    print(e)

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

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

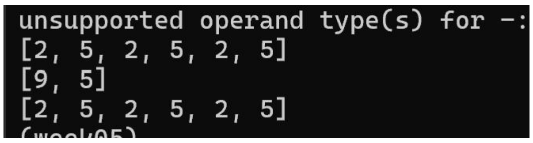


```
False
unsupported operand type(s) for -: 'list' and 'list'
[2, 5, 2, 5, 2, 5]
[2, 5]
[2, 5, 2, 5, 2, 5]
(week05)
```

列表不能做减法

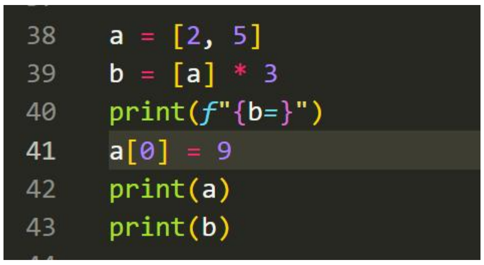


```python
a = [2, 5]
b = a * 3
a[0] = 9
print(a)
print(b)
```

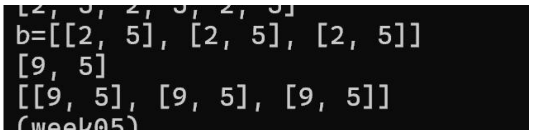


```
unsupported operand type(s) for -:
[2, 5, 2, 5, 2, 5]
[9, 5]
[2, 5, 2, 5, 2, 5]
```

a 被修改，b 没有被修改



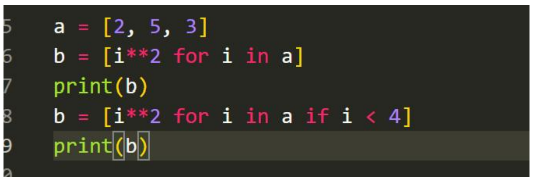
```python
a = [2, 5]
b = [a] * 3
print(f"{b=}")
a[0] = 9
print(a)
print(b)
```



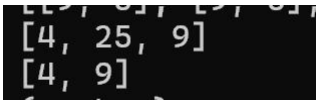
```
b=[[2, 5], [2, 5], [2, 5]]
[9, 5]
[[9, 5], [9, 5], [9, 5]]
```

在修改的时候一定要关注修改的对象到底是谁，修改结果是什么！！！

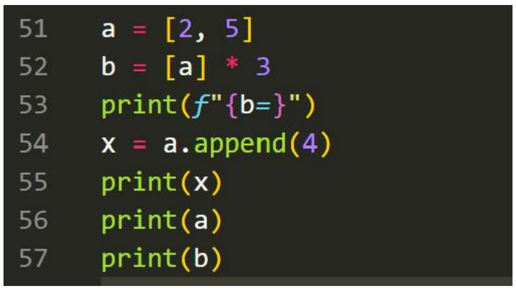
对象思维！！！



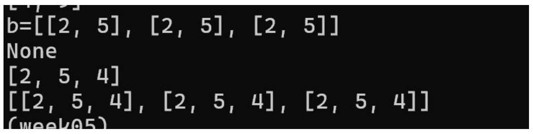
```python
a = [2, 5, 3]
b = [i**2 for i in a]
print(b)
b = [i**2 for i in a if i < 4]
print(b)
```



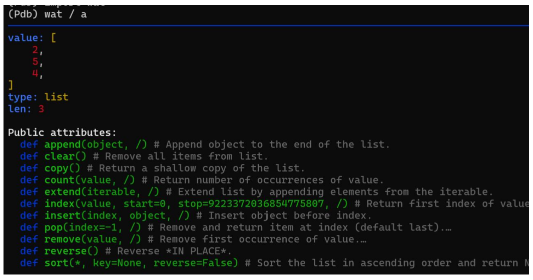
```
[4, 25, 9]
[4, 9]
```



```python
a = [2, 5]
b = [a] * 3
print(f"{b=}")
x = a.append(4)
print(x)
print(a)
print(b)
```

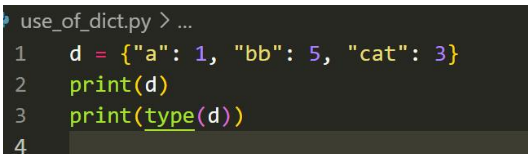


```
b=[[2, 5], [2, 5], [2, 5]]
None
[2, 5, 4]
[[2, 5, 4], [2, 5, 4], [2, 5, 4]]
(week05)
```

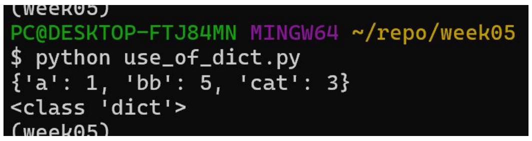


```
(Pdb) wat / a

value: [
    2,
    5,
    4,
]
type: list
len: 3

Public attributes:
  def append(object, /) # Append object to the end of the list.
  def clear() # Remove all items from list.
  def copy() # Return a shallow copy of the list.
  def count(value, /) # Return number of occurrences of value.
  def extend(iterable, /) # Extend list by appending elements from the iterable.
  def index(value, start=0, stop=9223372036854775807, /) # Return first index of value
  def insert(index, object, /) # Insert object before index.
  def pop(index=-1, /) # Remove and return item at index (default last).…
  def remove(value, /) # Remove first occurrence of value.…
  def reverse() # Reverse *IN PLACE*.
  def sort(*, key=None, reverse=False) # Sort the list in ascending order and return N
```

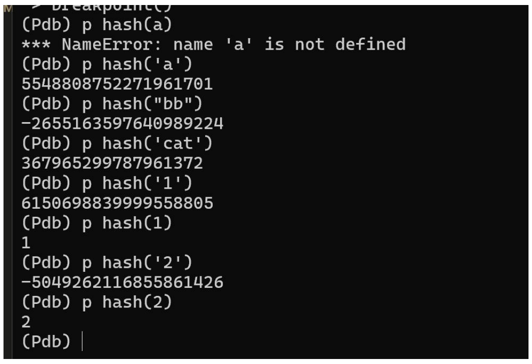
4.dict 字典



```
use_of_dict.py > ...
d = {"a": 1, "bb": 5, "cat": 3}
print(d)
print(type(d))
```



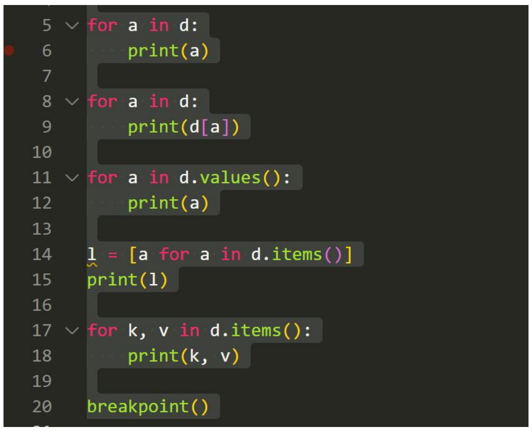
```
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05
$ python use_of_dict.py
{'a': 1, 'bb': 5, 'cat': 3}
<class 'dict'>
```



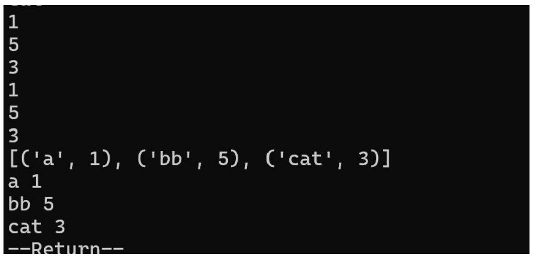
```
(Pdb) p hash(a)
*** NameError: name 'a' is not defined
(Pdb) p hash('a')
5548808752271961701
(Pdb) p hash("bb")
-2655163597640989224
(Pdb) p hash('cat')
367965299787961372
(Pdb) p hash('1')
6150698839999558805
(Pdb) p hash(1)
1
(Pdb) p hash('2')
-5049262116855861426
(Pdb) p hash(2)
2
(Pdb)
```

Hash 得到的是哈希值

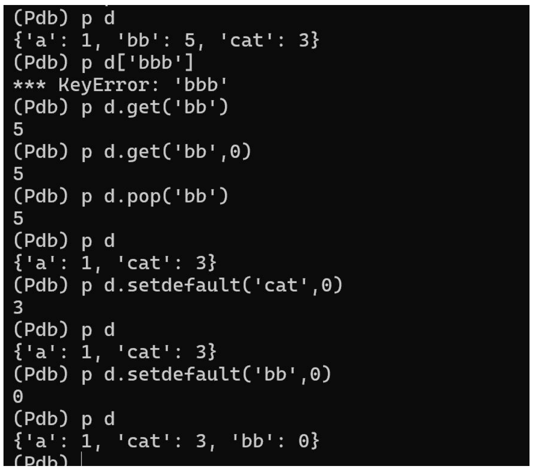
哈希值（Hash Value），也称为散列值或哈希码，是通过特定的哈希函数对数据（如文件、字符串、图像等）进行计算后得到的固定长度的数值。



```
for a in d:
    print(a)

for a in d:
    print(d[a])

for a in d.values():
    print(a)

l = [a for a in d.items()]
print(l)

for k, v in d.items():
    print(k, v)

breakpoint()
```



```
1
5
3
1
5
3
[('a', 1), ('bb', 5), ('cat', 3)]
a 1
bb 5
cat 3
--Return--
```

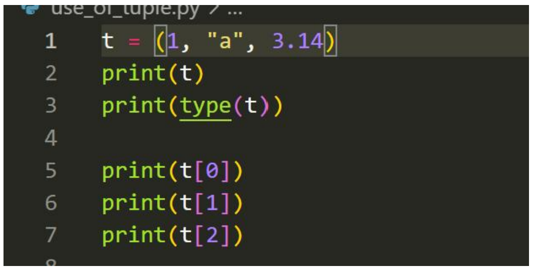


```
(Pdb) p d
{'a': 1, 'bb': 5, 'cat': 3}
(Pdb) p d['bbb']
*** KeyError: 'bbb'
(Pdb) p d.get('bb')
5
(Pdb) p d.get('bb',0)
5
(Pdb) p d.pop('bb')
5
(Pdb) p d
{'a': 1, 'cat': 3}
(Pdb) p d.setdefault('cat',0)
3
(Pdb) p d
{'a': 1, 'cat': 3}
(Pdb) p d.setdefault('bb',0)
0
(Pdb) p d
{'a': 1, 'cat': 3, 'bb': 0}
```

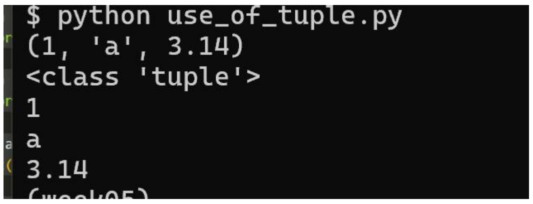
setdefault 是 Python 字典（dict）对象的一个方法，主要用于在字典中查找指定的键，如果该键存在，就返回其对应的值；如果该键不存在，就会在字典中插入这个键，并为其设置一个默认值，然后返回这个默认值

六、tuple ~ date 等类型

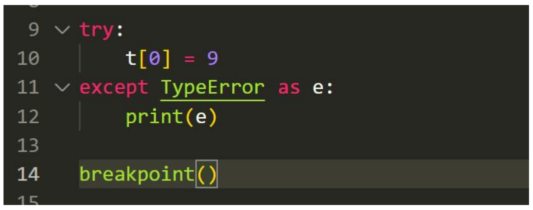
1.tuple 元组()



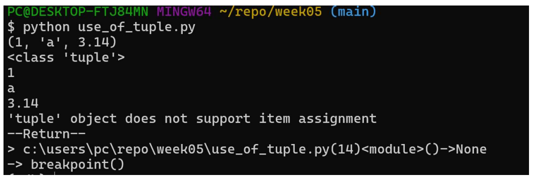
```
use_of_tuple.py > ...
t = (1, "a", 3.14)
print(t)
print(type(t))

print(t[0])
print(t[1])
print(t[2])
```



```
$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
```



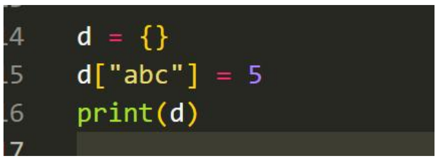
```
try:
    t[0] = 9
except TypeError as e:
    print(e)

breakpoint()
```

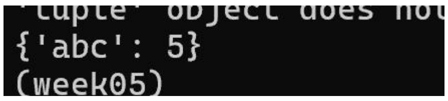


```
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
'tuple' object does not support item assignment
--Return--
> c:\users\pc\repo\week05\use_of_tuple.py(14)<module>()->None
-> breakpoint()
```

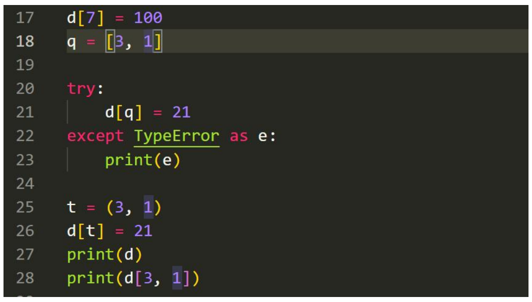
元组和列表的区别

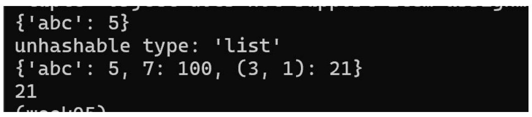
大多数情况下列表比元组更强大，元组不可修改的特性

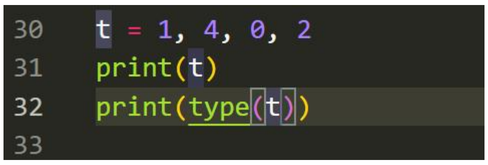

```python
d = {}
d["abc"] = 5
print(d)
```

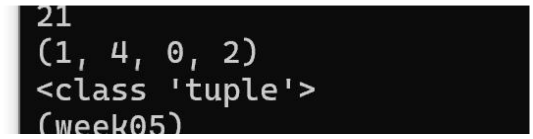

```
'tuple' object does not
{'abc': 5}
(week05)
```

字典中可修改的不可以成为键 key

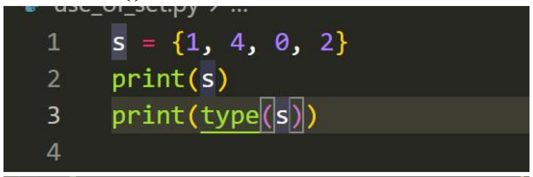

```python
d[7] = 100
q = [3, 1]

try:
    d[q] = 21
except TypeError as e:
    print(e)

t = (3, 1)
d[t] = 21
print(d)
print(d[3, 1])
```

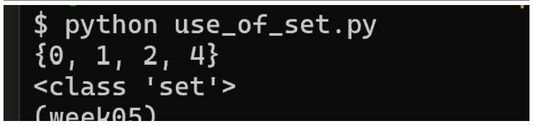

```
{'abc': 5}
unhashable type: 'list'
{'abc': 5, 7: 100, (3, 1): 21}
21
```

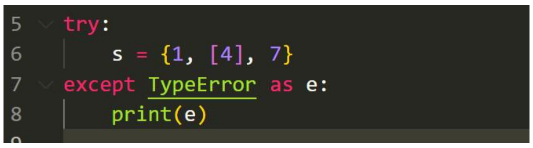

```python
t = 1, 4, 0, 2
print(t)
print(type(t))
```

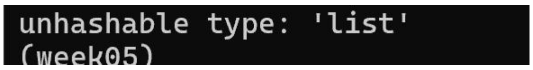

```
21
(1, 4, 0, 2)
<class 'tuple'>
(week05)
```

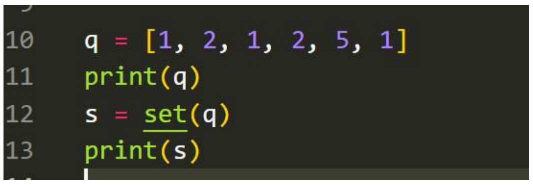
2.set 集合{}

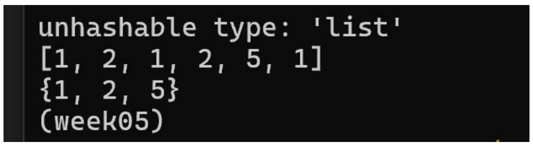

```python
s = {1, 4, 0, 2}
print(s)
print(type(s))
```


```
$ python use_of_set.py
{0, 1, 2, 4}
<class 'set'>
(week05)
```

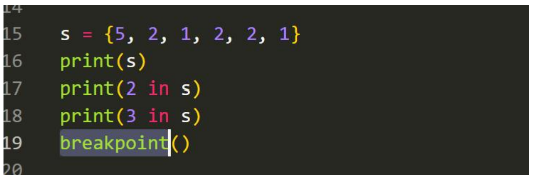

```python
try:
    s = {1, [4], 7}
except TypeError as e:
    print(e)
```

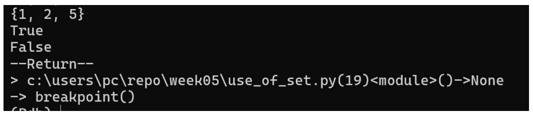

```
unhashable type: 'list'
(week05)
```

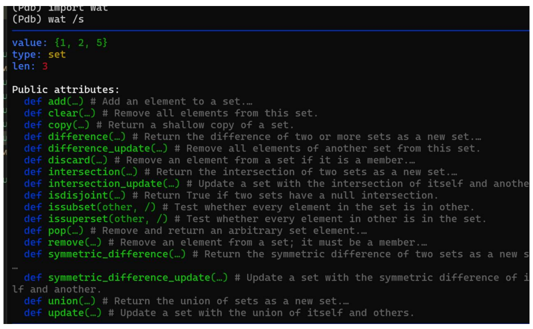

```python
q = [1, 2, 1, 2, 5, 1]
print(q)
s = set(q)
print(s)
```

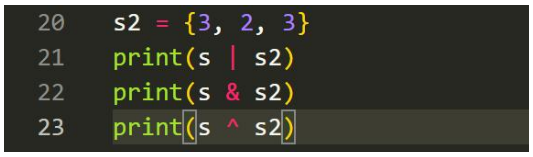

```
unhashable type: 'list'
[1, 2, 1, 2, 5, 1]
{1, 2, 5}
(week05)
```

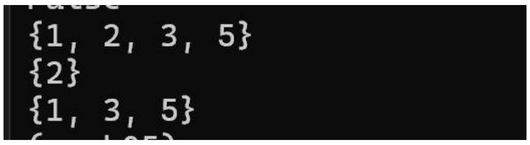
集合可以唯一化


```python
s = {5, 2, 1, 2, 2, 1}
print(s)
print(2 in s)
print(3 in s)
breakpoint()
```


```
{1, 2, 5}
True
False
--Return--
> c:\users\pc\repo\week05\use_of_set.py(19)<module>()->None
-> breakpoint()
```

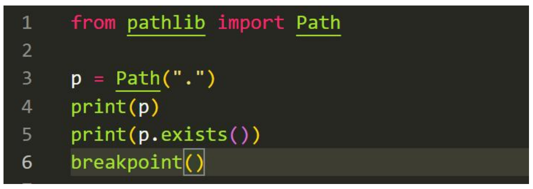

```
(Pdb) import wat
(Pdb) wat /s

value: {1, 2, 5}
type: set
len: 3

Public attributes:
  def add(…) # Add an element to a set.…
  def clear(…) # Remove all elements from this set.
  def copy(…) # Return a shallow copy of a set.
  def difference(…) # Return the difference of two or more sets as a new set.…
  def difference_update(…) # Remove all elements of another set from this set.
  def discard(…) # Remove an element from a set if it is a member.…
  def intersection(…) # Return the intersection of two sets as a new set.…
  def intersection_update(…) # Update a set with the intersection of itself and anothe
  def isdisjoint(…) # Return True if two sets have a null intersection.
  def issubset(other, /) # Test whether every element in the set is in other.
  def issuperset(other, /) # Test whether every element in other is in the set.
  def pop(…) # Remove and return an arbitrary set element.…
  def remove(…) # Remove an element from a set; it must be a member.…
  def symmetric_difference(…) # Return the symmetric difference of two sets as a new s
…
  def symmetric_difference_update(…) # Update a set with the symmetric difference of i
lf and another.
  def union(…) # Return the union of sets as a new set.…
  def update(…) # Update a set with the union of itself and others.
```

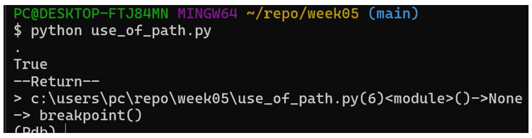

```python
s2 = {3, 2, 3}
print(s | s2)
print(s & s2)
print(s ^ s2)
```


```
False
{1, 2, 3, 5}
{2}
{1, 3, 5}
```

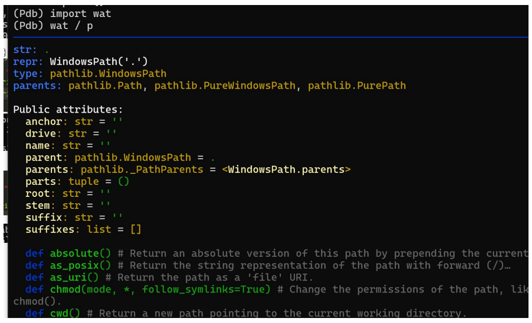
|是并集的意思，&是交集的意思，^是并行差的意思

3.pathlib 模块

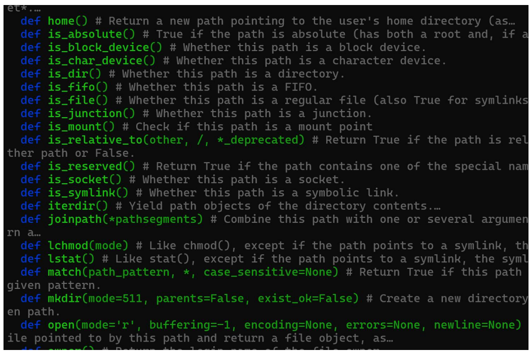

```python
from pathlib import Path

p = Path(".")
print(p)
print(p.exists())
breakpoint()
```



```
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$ python use_of_path.py
.
True
--Return--
> c:\users\pc\repo\week05\use_of_path.py(6)<module>()->None
-> breakpoint()
(Pdb)
```



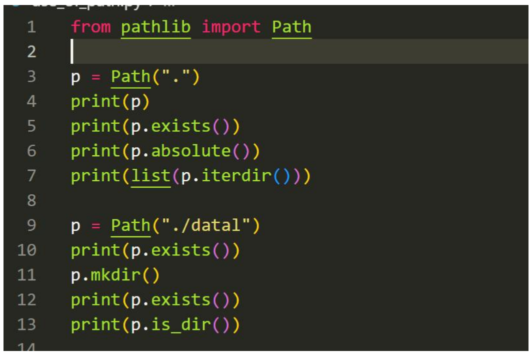
```
(Pdb) import wat
(Pdb) wat / p

str: .
repr: WindowsPath('.')
type: pathlib.WindowsPath
parents: pathlib.Path, pathlib.PureWindowsPath, pathlib.PurePath

Public attributes:
  anchor: str = ''
  drive: str = ''
  name: str = ''
  parent: pathlib.WindowsPath = .
  parents: pathlib._PathParents = <WindowsPath.parents>
  parts: tuple = ()
  root: str = ''
  stem: str = ''
  suffix: str = ''
  suffixes: list = []

  def absolute() # Return an absolute version of this path by prepending the current
  def as_posix() # Return the string representation of the path with forward (/)…
  def as_uri() # Return the path as a 'file' URI.
  def chmod(mode, *, follow_symlinks=True) # Change the permissions of the path, lik
chmod().
  def cwd() # Return a new path pointing to the current working directory.
```



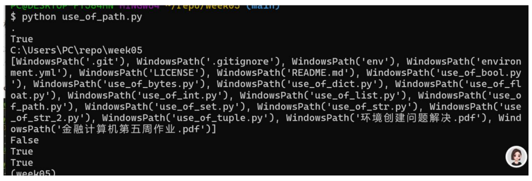
```
et*.…
  def home() # Return a new path pointing to the user's home directory (as…
  def is_absolute() # True if the path is absolute (has both a root and, if a
  def is_block_device() # Whether this path is a block device.
  def is_char_device() # Whether this path is a character device.
  def is_dir() # Whether this path is a directory.
  def is_fifo() # Whether this path is a FIFO.
  def is_file() # Whether this path is a regular file (also True for symlinks
  def is_junction() # Whether this path is a junction.
  def is_mount() # Check if this path is a mount point
  def is_relative_to(other, /, *_deprecated) # Return True if the path is rel
ther path or False.
  def is_reserved() # Return True if the path contains one of the special nam
  def is_socket() # Whether this path is a socket.
  def is_symlink() # Whether this path is a symbolic link.
  def iterdir() # Yield path objects of the directory contents.…
  def joinpath(*pathsegments) # Combine this path with one or several argumen
rn a…
  def lchmod(mode) # Like chmod(), except if the path points to a symlink, th
  def lstat() # Like stat(), except if the path points to a symlink, the syml
  def match(path_pattern, *, case_sensitive=None) # Return True if this path
given pattern.
  def mkdir(mode=511, parents=False, exist_ok=False) # Create a new directory
en path.
  def open(mode='r', buffering=-1, encoding=None, errors=None, newline=None)
ile pointed to by this path and return a file object, as…
```

在 pathlib 模块中，Path 类有 absolute() 方法 。

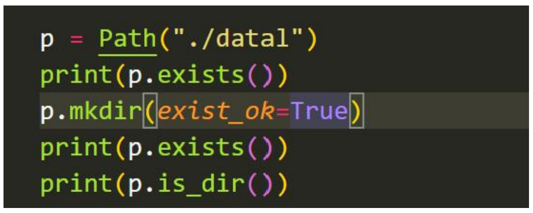
• 功能：将相对路径转换为绝对路径，简单通过追加当前工作目录来实现，不会展开符号链接和解析相对路径标记（如 .. 和 . ） 。比如当前工作目录是 /home/user，有相对路径 test.txt ，Path('test.txt').absolute() 会得到 /home/user/test.txt 。



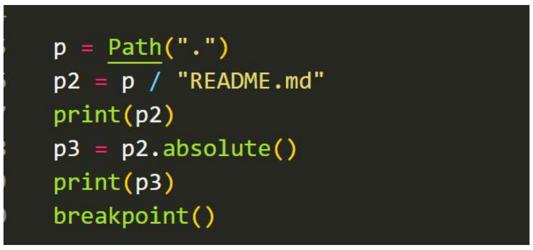
```python
from pathlib import Path

p = Path(".")
print(p)
print(p.exists())
print(p.absolute())
print(list(p.iterdir()))

p = Path("./datal")
print(p.exists())
p.mkdir()
print(p.exists())
print(p.is_dir())
```



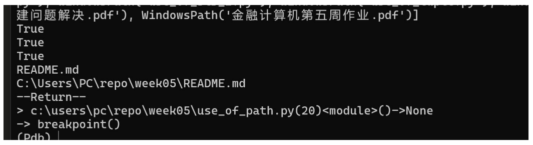
```
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$ python use_of_path.py
.
True
C:\Users\PC\repo\week05
[WindowsPath('.git'), WindowsPath('.gitignore'), WindowsPath('env'), WindowsPath('environ
ment.yml'), WindowsPath('LICENSE'), WindowsPath('README.md'), WindowsPath('use_of_bool.py
'), WindowsPath('use_of_bytes.py'), WindowsPath('use_of_dict.py'), WindowsPath('use_of_fl
oat.py'), WindowsPath('use_of_int.py'), WindowsPath('use_of_list.py'), WindowsPath('use_o
f_path.py'), WindowsPath('use_of_set.py'), WindowsPath('use_of_str.py'), WindowsPath('use
_of_str_2.py'), WindowsPath('use_of_tuple.py'), WindowsPath('环境创建问题解决.pdf'), Wind
owsPath('金融计算机第五周作业.pdf')]
False
True
True
(week05)
```



```python
p = Path("./datal")
print(p.exists())
p.mkdir(exist_ok=True)
print(p.exists())
print(p.is_dir())
```



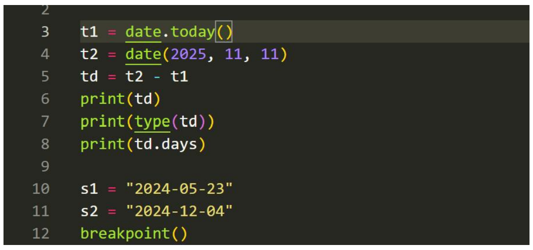
```python
p = Path(".")
p2 = p / "README.md"
print(p2)
p3 = p2.absolute()
print(p3)
breakpoint()
```



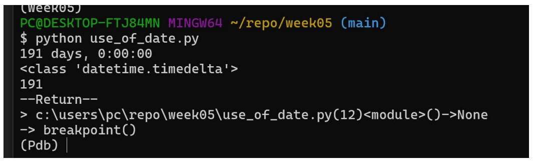
```
建问题解决.pdf'), WindowsPath('金融计算机第五周作业.pdf')]
True
True
True
README.md
C:\Users\PC\repo\week05\README.md
--Return--
> c:\users\pc\repo\week05\use_of_path.py(20)<module>()->None
-> breakpoint()
(Pdb)
```

在前面的 mkdir 后面加上一小段代码，再运行下面的代码，才能够不报错

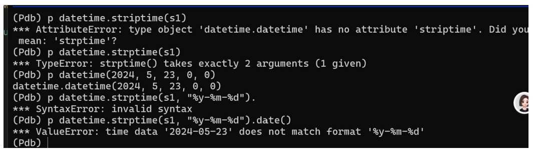
4.datetime 模块



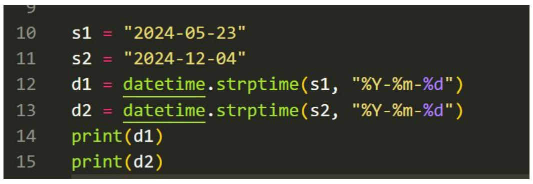
```python
t1 = date.today()
t2 = date(2025, 11, 11)
td = t2 - t1
print(td)
print(type(td))
print(td.days)

s1 = "2024-05-23"
s2 = "2024-12-04"
breakpoint()
```



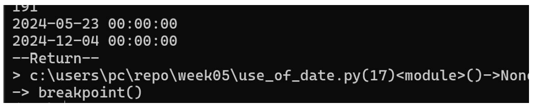
```
(week05)
PC@DESKTOP-FTJ84MN MINGW64 ~/repo/week05 (main)
$ python use_of_date.py
191 days, 0:00:00
<class 'datetime.timedelta'>
191
--Return--
> c:\users\pc\repo\week05\use_of_date.py(12)<module>()->None
-> breakpoint()
(Pdb)
```



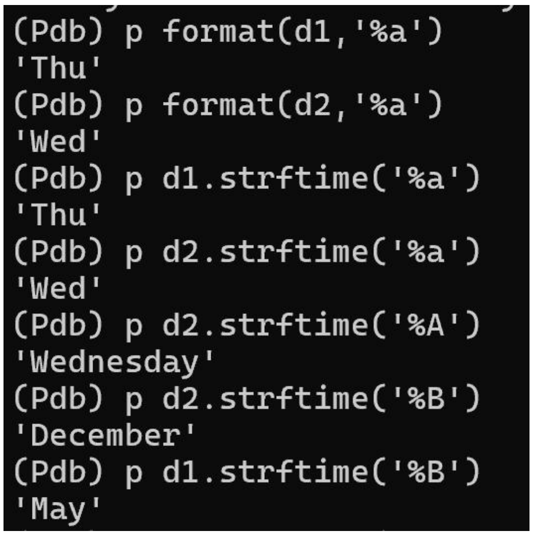
```
(Pdb) p datetime.striptime(s1)
*** AttributeError: type object 'datetime.datetime' has no attribute 'striptime'. Did you
 mean: 'strptime'?
(Pdb) p datetime.strptime(s1)
*** TypeError: strptime() takes exactly 2 arguments (1 given)
(Pdb) p datetime(2024, 5, 23, 0, 0)
datetime.datetime(2024, 5, 23, 0, 0)
(Pdb) p datetime.strptime(s1, "%y-%m-%d").
*** SyntaxError: invalid syntax
(Pdb) p datetime.strptime(s1, "%y-%m-%d").date()
*** ValueError: time data '2024-05-23' does not match format '%y-%m-%d'
(Pdb)
```



```python
s1 = "2024-05-23"
s2 = "2024-12-04"
d1 = datetime.strptime(s1, "%Y-%m-%d")
d2 = datetime.strptime(s2, "%Y-%m-%d")
print(d1)
print(d2)
```



```
191
2024-05-23 00:00:00
2024-12-04 00:00:00
--Return--
> c:\users\pc\repo\week05\use_of_date.py(17)<module>()->None
-> breakpoint()
```



```
(Pdb) p format(d1,'%a')
'Thu'
(Pdb) p format(d2,'%a')
'Wed'
(Pdb) p d1.strftime('%a')
'Thu'
(Pdb) p d2.strftime('%a')
'Wed'
(Pdb) p d2.strftime('%A')
'Wednesday'
(Pdb) p d2.strftime('%B')
'December'
(Pdb) p d1.strftime('%B')
'May'
```