1.复制第四周 environment.yml 至第五周，创建环境

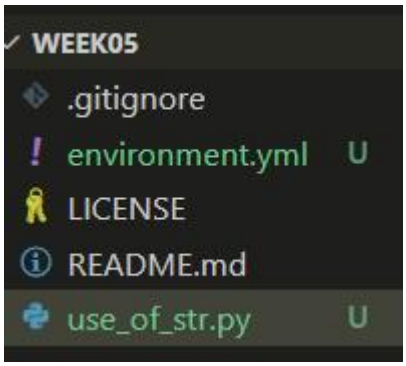
```
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ cat week04/environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ cp week04/environment.yml week05/

(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 86139 197609    91  4月 26 11:08 environment.yml
-rw-r--r-- 1 86139 197609 18805  4月 26 11:03 LICENSE
-rw-r--r-- 1 86139 197609  2239  4月 26 11:03 README.md

(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ cd week05

(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ conda env create
C:\Users\86139\anaconda3\Lib\argparse.py:2006: FutureWarning: `re
5.9. Use `conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
```

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



（1） 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)

（2） 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)

```
use_of_str.py > ...
1   a = "hello"
2   x = id(a)
3   print(x)
```

```
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2565247097888
```

use_of_str.py > ...

```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
```

```
(week05)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2011550456064
2011550454080
[9, 5]
[2, 5]
2011550456064
2011550454080
```

type() -- 返回对象的类型

```python
print(type(a))
```

```
(week05)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2501985835264
2501985833280
[9, 5]
[2, 5]
2501985835264
2501985833280
<class 'list'>
```

isinstance() -- 判断对象是否属于某个 (或某些) 类型

```python
print("isinstance(a,str):", isinstance(a, str))
```

```
isinstance(a,str): False
```

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

```python
print("dir(a):", dir(a))
```

```
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '_
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
```

str() -- 返回对象 print 时要显示在终端的字符串

（3）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

```
(week05)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python
Python 3.12.10 | packaged by conda-forge | (main, Apr 10 2025, 22:08:16) [MSC v.1943 64 bit (AMD64)] on win3
Type "help", "copyright", "credits" or "license" for more information.
>>> print(32)
32
>>> print(str(32))
32
```

（4）可以利用 assert 语句查验某个表达式 (expression) 为真，否则报错 (AssertionError) 退出

```
17  assert isinstance(a, list)
18  print("goodbye")
```

```
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2766523406592
2766523404608
[9, 5]
[2, 5]
2766523406592
2766523404608
<class 'list'>
isinstance(a,str): False
isinstance(a,str): True
True
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '_
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse'
'sort']
goodbye
```

（5）可以利用 try 语句拦截报错，避免退出，将流程 (flow) 转入 except 语句

```
17  try:
18      assert isinstance(a, list)
19  except AssertionError:
20      print("type error")
21  print("goodbye")
```

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

```
17  try:
18      assert isinstance(a, list)
19  except AssertionError:
20      breakpoint
21      print("type error")
22  print("goodbye")
```

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)

use_of_str.py > ...

```python
print("字面值")
s = "university"
print(s)
print(isinstance(s, str))
assert type(s) is str

print("f-string")
x = "Tom"
s = f"name:{x}"
print(s)

s = "a\tb"
print("TAB", s)

s = "aaa\nbbb"
print("new line", s)
```

```
86139@LAPTOP-J15OR7EU
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
TAB a   b
new line aaa
bbb
```

推导式 (comprehension) (仅限 list、dict、set)
初始化 (init)

```python
print("初始化")
s = str()
print(s)
s = [5,8,2]
print(s)
```

```
86139@LAPTOP-J15OR7EU MI
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
TAB a   b
new line aaa
bbb
xyz
abc
    eee
aaa

初始化

[5, 8, 2]
```

运算值 (operator)

```
25  print("初始化")
26  s = str()
27  print(s)
28  s = str([5, 8, 2])
29  print(s)
30
31  assert str([5, 8, 2]) == "[5,8,2]"
32  assert str(1.1 + 2.2) != "3.3"
33
34  assert str() == ""
35
36  s = "="
37  s = s * 20
38  print(s)
```

```
86139@LAPTOP-J15OR7EU MIN
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
TAB a    b
new line aaa
bbb
xyz
abc
    eee
aaa

初始化

[5, 8, 2]
====================
```

索引值 (subscription)

```
36    s = "="
37    x = id(s)
38    s = s * 20
39    y = id(s)
40    print(s)
41    assert x != y
```

```
43    s = "hello"
44    assert s[3] == "l"
45    assert s[-1] == "o"
46    assert s[:3] == "hel"
47    assert s[4] == s[-1]
48    try:
49        s[5]
50    except IndexError as e:
51        print(e)
```

返回值 (return value of function/method call)

```
53    s = "hello"
54    u = s.upper()
55    print(u)
56    print(s)
57
58    t = "name:{},age{}"
59    print(t)
60    t1 = t.format("Jack", 21)
61    print(t1)
```

```
HELLO
hello
name:{},age{}
name:Jack,age21
```

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

```
63    s1 = "abc"
64    s2 = "ghi"
65    s = s1 + s2
66    assert s == "abcghi"
67    print(s2 + s1)
68
69    try:
70        print(s2 - s1)
71    except TypeError as e:
72        print(e)
```

如何判断相等 (==)

```
s = "=*="
s = s * 10
print(s)

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

assert s == "aaaa"
```

```
ghiabc
unsupported operand type(s) for -: 'str' and 'str'
=*==*==*==*==*==*==*==*==*=
unsupported operand type(s) for /: 'str' and 'int'
assert s == "aaaa"
       ^^^^^^^^^^
```

对于比较运算符 (>、<、>=、<=) 有没有支持

```
print("abc" > "ABC")     True
print("123" > "abcd")    False
print("9" > ",")         True
print("9" < ":")         True
print("book" < "box")    True
print("book" < "{")      True
```

什么值被当作 True，什么值被当作 False

```
assert "book"
assert ""
```

```
Traceback (most recent call last):
  File "C:\Users\86139\repo\week05\use_of_str.py", line 92, in <module>
    assert ""
           ^^
AssertionError
```

是否可迭代 (iterable)，如何做迭代 (for 循环)

```
assert "book"
assert not ""

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

```
<str_ascii_iterator object at 0x000001454C7CD120>
```

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

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

```
b
o
o
k
4
```

【breakpoint（）】

是否支持返回长度 (len)

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

```
b
o
o
k
4
```

是否 (如何) 支持提取操作 (subscription) ([] 运算符)

```
s = "book"
assert s[1:3] == "oo"

s = "the book of why took nooo"
print(s.capitalize())
print(s)
print(s.count("oo") == 3)
```

```
The book of why took nooo
the book of why took nooo
True
```

拥有哪些常用方法 (method) 可供调用 (() 运算符)

```
s = "the book of why took nooo"
print(s.capitalize())
print(s)
print(s.count("oo") == 3)

print("abc123".isalnum())
print("abc123 ".isalnum())
print("abc123".isidentifier())

q = ["rose", "jack", "bob"]
print(";".join(q))
s = "rose:jack:bob"
print(s.split(":"))
assert s.partition(":") == ("rose", ":", "jack:bob")
```

```
True
True
False
True
rose;jack;bob
['rose', 'jack', 'bob']
```

5.字节串

use_of_bytes.py > ...

```python
from pathlib import Path

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

p = Path("C:\\Users\\86139\\anaconda3\\envs\\week05\\python.exe")
s = p.read_bytes()
print(len(s))

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

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

s = "你好"
b1 = s.encode("utf-8")
print(b1)
b2 = s.encode("gbk")
print(b2)

s = "abc你好😊"
print(s)
b = s.encode()
```

```
$ python use_of_bytes.py
b'hello'
104
93184
110
b'\xe4\xbd\xa0\xe5\xa5\xbd'
b'\xc4\xe3\xba\xc3'
abc你好😊
```

6.整数

```
(Pdb) import wat
(Pdb) wat /x
```

```
value: 5
type: int

Public attributes:
  denominator: int = 1
  imag: int = 0
  numerator: int = 5
  real: int = 5

  def as_integer_ratio() # Return a pair of integers, whose ratio is
 equal to the original int.…
  def bit_count() # Number of ones in the binary representation of t
he absolute value of self.…
  def bit_length() # Number of bits necessary to represent self in b
inary.…
  def conjugate(…) # Returns self, the complex conjugate of any int.
  def from_bytes(bytes, byteorder='big', *, signed=False) # Return t
he integer represented by the given array of bytes.…
  def is_integer() # Returns True. Exists for duck type compatibilit
y with float.is_integer.
  def to_bytes(length=1, byteorder='big', *, signed=False) # Return
an array of bytes representing an integer.…
```

```
(Pdb) p x
5
(Pdb) for i in x:
*** IndentationError: expected an indented block af
(Pdb) for i in x: print(i)
*** TypeError: 'int' object is not iterable
(Pdb) p iter(x)
*** TypeError: 'int' object is not iterable
(Pdb) p len(x)
*** TypeError: object of type 'int' has no len()
(Pdb) p x[0]
*** TypeError: 'int' object is not subscriptable
(Pdb) wat / x
*** NameError: name 'wat' is not defined
```

7.浮点数

use_of_float.py > ...

```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)

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)
```

```
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.41827782285672055
nan
False
False
False
0.0314
True
False
True
-inf
```

8.布尔值

use_of_bool.py > ...

```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
```

9.列表

use_of_list.py > ...

```python

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

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

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

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(f"{b=}")
a[0] = 9
print(a)
print(b)

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

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

10.字典

use_of _dict.py > ...

```python
d = {"a": 1, "bb": 5, "cat": 3}
print(d)
print(type(d))

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()
```

```
[EOF]
(Pdb) wat / d
```

```
value: {
    'a': 1,
    'bb': 5,
    'cat': 3,
}
type: dict
len: 3

Public attributes:
  def clear(…) # D.clear() -> None.  Remove all items from D.
  def copy(…) # D.copy() -> a shallow copy of D
  def fromkeys(iterable, value=None, /) # Create a new dictionary with keys from
 iterable and values set to value.
  def get(key, default=None, /) # Return the value for key if key is in the dict
ionary, else default.
  def items(…) # D.items() -> a set-like object providing a view on D's items
  def keys(…) # D.keys() -> a set-like object providing a view on D's keys
  def pop(…) # D.pop(k[,d]) -> v, remove specified key and return the correspond
ing value.…
  def popitem() # Remove and return a (key, value) pair as a 2-tuple.…
  def setdefault(key, default=None, /) # Insert key with a value of default if k
ey is not in the dictionary.…
  def update(…) # D.update([E, ]**F) -> None.  Update D from mapping/iterable E
and F.…
  def values(…) # D.values() -> an object providing a view on D's values
```

11.元组

use_of_tuple.py > ...

```python
t = (1, "a", 3.14)
print(t)
print(type(t))

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

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

d = {}
d["abc"] = 5
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])

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

12.集合

use_of_set.py > ...

```python
s = {1, 4, 7}
print(s)
print(type(s))

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

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

s = (5, 2, 1, 2, 2, 1)
print(s)
print(2 in s)
print(3 in s)

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

13. Path

use_of_path.py > ...

```python
from pathlib import Path
from pprint import pprint

p = Path(".")
print(p)
print(p.exists())
print(p.absolute())
pprint(list(p.iterdir()))
p = Path("./data1")
print(p.exists())
p.mkdir(exist_ok=True)
print(p.exists())
print(p.is_dir())
p = Path(".")
p2 = p / "README.md"
print(p2)
p3 = p2.absolute()
print(p3)
```

14.Datetime

use_of_datetime.py > ...

```python
from datetime import date, datetime

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"
d1 = datetime.strftime(s1, "%Y-%m-%d")
d2 = datetime.strftime(s2, "%Y-%m-%d")
print(d1)
print(d2)
```