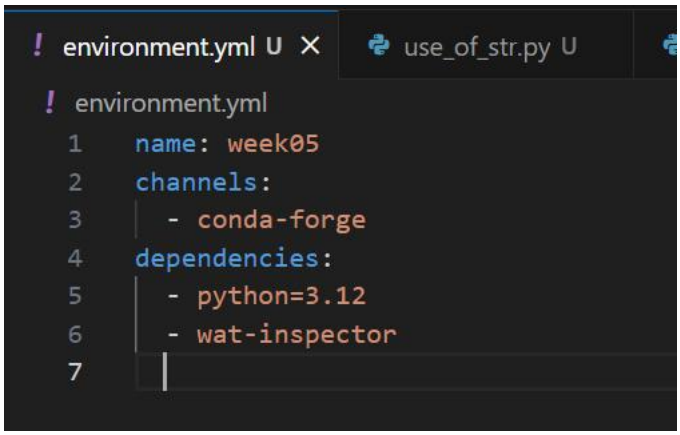
第五周：对 Python 的基本概念进行学习，灵活运用

- 建立的环项目环境

1. 确保自己建立的文件夹中含有 environment.yml 的文件（文件内包含基础配置）

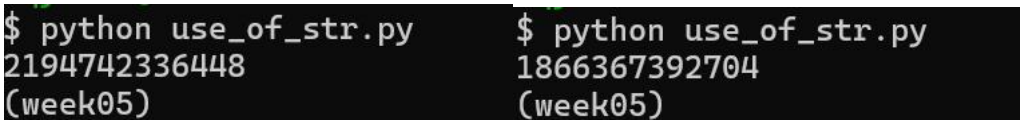


2. 终端运行 conda env create，进行相关软件下载
3. 查询目前有的 Python 环境:conda env list
4. 终端运行 conda activate week05（第二次打开可直接进行这一步）

Python 对象类型 (type)，包括**字符串 (str)、字节串 (bytes)、整数 (int)、浮点数 (float)、布尔值 (bool)、列表 (list)、字典 (dict)、元组 (tuple)、集合 (set)**。建立 use_of_str.py 或者 use_of_bytes.py.进行想过基本概念的验证

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)



每次相同的内容每次运行后显示的 id 不同，但是在同一次运行中即使改变 a 的内容，但是 id 都不变

字符串在电脑里会有缓存（即使 b=”nice’,再使 y 等于 id(b)，其最终显示的 a 和 b 的 id 都是一样的）

type() -- 对象的类型

isinstance() -- 判断对象是否属于某个 (或某些) 类型（可以同时判断多个）

dir() -- 对象所支持的属性 (attributes) 的名称列表

str() -- 对象 print 时要显示在终端的字符串，只有字符串能够被 print 出来

print() 函数将表达式 (expression) 输出到终端，

可以利用 assert 语句查验某个表达式是 false ，就会直接报错退出，在报错之后的命令就都不显示；若某个表达式是 true，就无事发生，程序继续向后运行（assert 可以用来做一些检查）

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

可以调用 breakpoint() 函数暂停程序运行，进入 pdb 调试 (debug) 模式，可以在调试器中 import wat,再 wat /s，来查看 s 的相关信息

（或者需要输入 python -m pdb use_of_str.py 进入调试器）

use_of_str.py > ...

```python
a = [3, 6, 9, 8]
x = id(a)
print(x)
print("isinstance(a,;list):", isinstance(a, list))
print("dir(a)", dir(a))
try:
    assert isinstance(a, str)
except AssertionError:
    print("type error")
print("nice")
```

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），相关的笔记顺手记在 use_of_xx 的文件中一般包括以下途径:

- 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)
-

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):

输出结果：

- 对数学运算符 (+、-、*、**、/、//、%、@) 有没有支持
- 如何判断相等 (==)
- 对于比较运算符 (>、<、>=、<=) 有没有支持
- 字符串的大小比较搜索 ASCII Table
- 什么值被当作 True，什么值被当作 False(当出现 False 时，assert 会报错）长度不为 0 的字符串默认为 Ture
- 是否可迭代 (iterable)，如何做迭代 (可迭代的可以做 for 循环)→可迭代输入 iter()就可以输入迭代器中
- 是否支持返回长度 (len)
- 是否 (如何) 支持提取操作 (subscription) ([] 运算符)
- 拥有哪些常用方法 (method) 可供调用 (() 运算符：首字母大写：capitalize()

图中在调试器中查看常用函数

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

```
value: 'seventeen right here'
type: str
len: 20

Public attributes:
  def capitalize() # Return a capitalized version of the string.…
  def casefold() # Return a version of the string suitable for ca
seless comparisons.
  def center(width, fillchar=' ', /) # Return a centered string o
f length width.…
  def count(…) # S.count(sub[, start[, end]]) -> int…
  def encode(encoding='utf-8', errors='strict') # Encode the stri
ng using the codec registered for encoding.…
  def endswith(…) # S.endswith(suffix[, start[, end]]) -> bool…
  def expandtabs(tabsize=8) # Return a copy where all tab charact
ers are expanded using spaces.…
  def find(…) # S.find(sub[, start[, end]]) -> int…
  def format(…) # S.format(*args, **kwargs) -> str…
  def format_map(…) # S.format_map(mapping) -> str…
  def index(…) # S.index(sub[, start[, end]]) -> int…
  def isalnum() # Return True if the string is an alpha-numeric s
tring, False otherwise.…
  def isalpha() # Return True if the string is an alphabetic stri
```

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

字节：bytes

查看某一路径是否存在：

```
(Pdb) p p.exists()
False
```

p p.is_file() 或者 p p.is_dir()→判断是否是文件夹

```
value: [
    5,
    6,
    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 fro
m 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 ascendi
ng order and return None.…
```

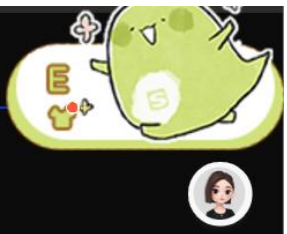
```
value: {
    'apple': 1,
    'boy': 2,
    'cat': 3,
}
type: dict
len: 3

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 dictionary, 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 corresponding value.…
  def popitem() # Remove and return a (key, value) pair as a 2-tu
ple.…
  def setdefault(key, default=None, /) # Insert key with a value
of default if key is not in the dictionary.…
  def update(…) # D.update([E, ]**F) -> None.  Update D from mapp
ing/iterable E and F.…
  def values(…) # D.values() -> an object providing a view on D's
 values
```

```
value: (17, 'seventeen', 5.26)
type: tuple
len: 3

Public attributes:
  def count(value, /) # Return number of occurrences of value.
  def index(value, start=0, stop=9223372036854775807, /) # Return
 first index of value.…
```

可以做集合的并交补，不在乎顺序的时候可以使用



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

value: {1, 5, 25}
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 a
s a new set.…
  def difference_update(…) # Remove all elements of another set f
rom 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 another.
  def isdisjoint(…) # Return True if two sets have a null interse
ction.
  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 memb
er.…
  def symmetric_difference(…) # Return the symmetric difference o
f two sets as a new set.…
  def symmetric_difference_update(…) # Update a set with the symm
```