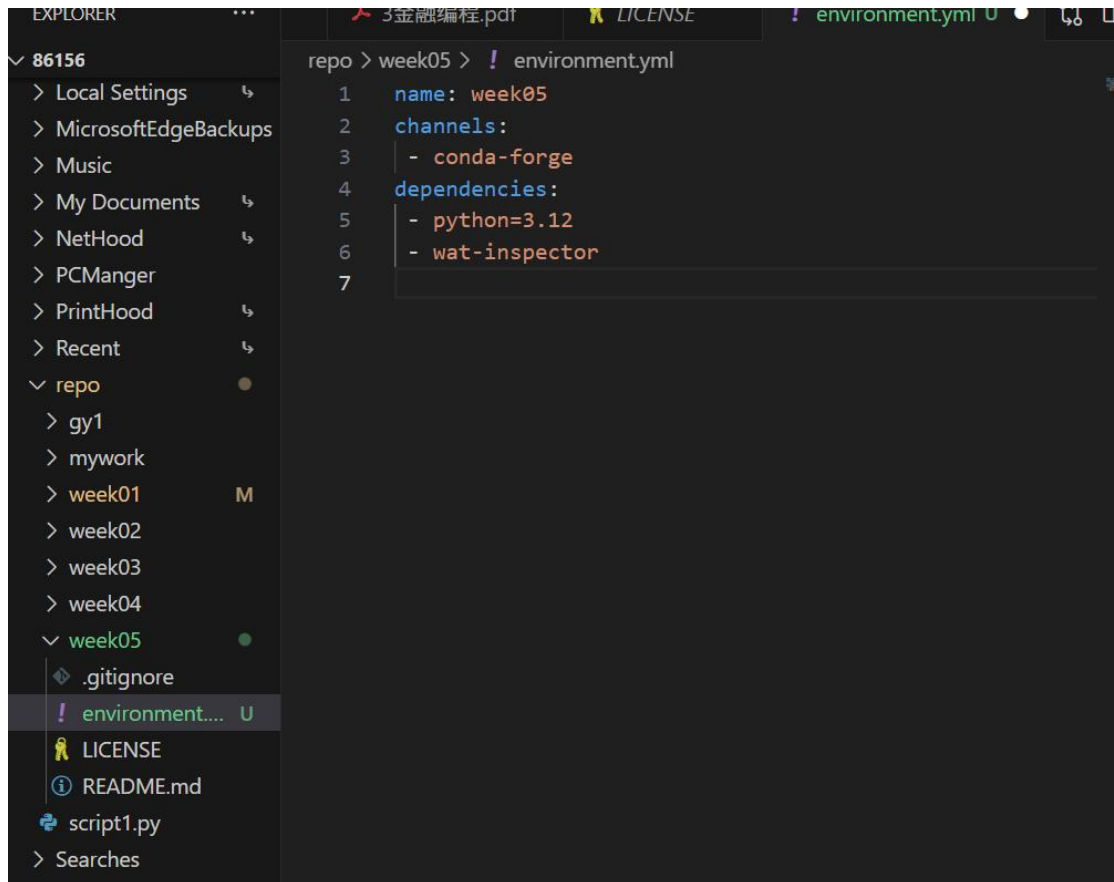
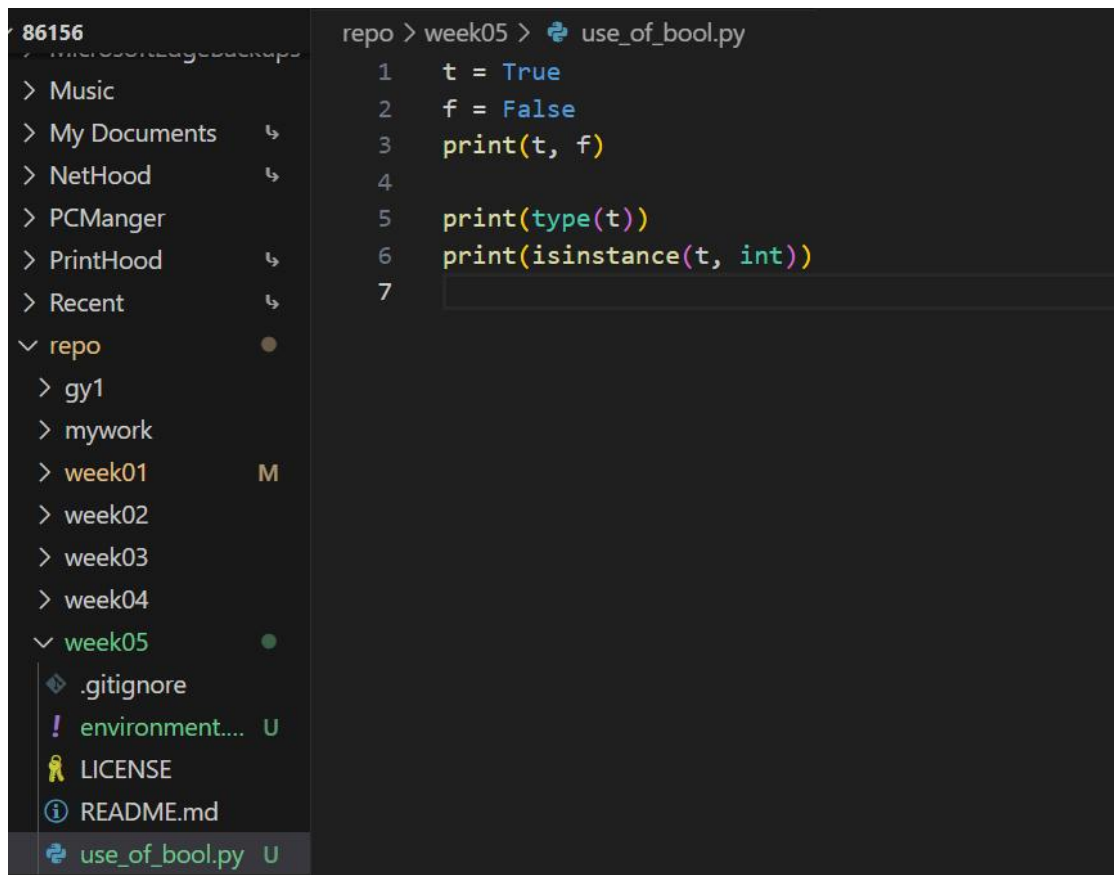


第五周



```
EXPLORER
86156
  > Local Settings
  > MicrosoftEdgeBackups
  > Music
  > My Documents
  > NetHood
  > PCManger
  > PrintHood
  > Recent
  > repo
  > gy1
  > mywork
  > week01
  > week02
  > week03
  > week04
  > week05
    .gitignore
    ! environment.... U
    LICENSE
    README.md
    script1.py
  > Searches

repo > week05 > ! environment.yml
1  name: week05
2  channels:
3    - conda-forge
4  dependencies:
5    - python=3.12
6    - wat-inspector
7
```



The image shows a code editor interface. On the left is a file explorer with a tree view. The root is '86156', which contains several folders like 'Music', 'My Documents', 'NetHood', 'PCManger', 'PrintHood', 'Recent', and 'repo'. The 'repo' folder is expanded, showing subfolders 'gy1', 'mywork', 'week01', 'week02', 'week03', 'week04', and 'week05'. The 'week05' folder is selected and expanded, showing files: '.gitignore', 'environment.... U', 'LICENSE', 'README.md', and 'use_of_bool.py U'. On the right, the 'use_of_bool.py' file is open, showing the following Python code:

```
repo > week05 > use_of_bool.py
1  t = True
2  f = False
3  print(t, f)
4
5  print(type(t))
6  print(isinstance(t, int))
7
```

常用内置函数

在学习和操作这些对象类型时，Python 提供了一系列内置函数帮助我们进行对象检视和操作：

`id()`：返回对象在虚拟内存中的地址（正整数），若 `id(a)==id(b)`，则 `a is b` 成立，`is` 用于判断两个对象是否为同一内存地址。

`type()`：返回对象的类型，例如 `type(1)` 返回 `<class 'int'>`。

`isinstance()`：用于判断对象是否属于某个（或某些）类型，如 `isinstance(1,int)` 返回 `True`。

`dir()`：返回对象所支持的属性（attributes）的名称列表，通过它可以查看对象可用的方法和属性。

`str()`：将对象转换为字符串形式，返回对象 `print` 时要显示在终端的字符串。

`print()`：将表达式输出到终端，方便查看结果。

`assert`：用于查验某个表达式为真，若表达式为 `False`，则抛出 `AssertionError` 异常退出。

`try-except`: 用于拦截报错, 避免程序直接退出, 将流程转入 `except` 语句块进行异常处理。

`breakpoint()`: 暂停程序运行, 进入 `pdb` 调试模式, 方便逐行检查代码和变量状态。

二、各对象类型详细学习

2.1 字符串(str)

字符串是用于表示文本的数据类型, 由 Unicode 字符组成。

2.1.1 获取字符串实例的途径

- 字面值: 使用单引号 `'`、双引号 `"` 或三引号 `'''`、`"""` 定义, 如 `s1='Hello'`, `s2="World"`, 三引号常用于多行字符串, 如 `s3='''This is a multi-line string'''`。此外, f-string 语法可以方便地嵌入表达式, 如 `name="Alice"; print(f"Hello, {name}!")`。
- 初始化: 通过 `str()` 函数进行初始化, 例如 `s=str(123)` 会将整数 123 转换为字符串 `"123"`。
- 运算值: 字符串可以通过 `+` 进行拼接, `*` 进行重复, 如 `"Hello"+" "+"World"` 得到 `"HelloWorld"`, `"Hi"*3` 得到 `"HiHiHi"`。
- 提取值: 通过索引 `[]` 提取单个字符 (索引从 0 开始), 如 `"Hello"[0]` 返回 `"H"`; 也可以通过切片 `[start:stop:step]` 提取子串, 如 `"Hello"[1:4]` 返回 `"ello"`。
- 返回值: 许多字符串方法会返回新的字符串, 例如 `"Hello".upper()` 返回 `"HELLO"`。

2.1.2 字符串的属性

- 数学运算符: 支持 `+` 拼接和 `*` 重复, 不支持减法、乘法等其他数学运算符。
- 相等判断: 使用 `==` 比较字符串内容是否相同, `is` 比较内存地址。
- 比较运算符: 支持 `>`、`<`、`>=`、`<=`, 按照字典序进行比较, 例如 `"abc"<"abd"` 返回 `True`。
- 布尔值: 非空字符串为 `True`, 空字符串 `""` 为 `False`。
- 可迭代性: 字符串是可迭代对象, 可以通过 `for` 循环遍历每个字符, 如 `for char in "Hello": print(char)`。
- 长度: 可以使用 `len()` 函数获取字符串长度, 如 `len("Hello")` 返回 5。
- 常用方法: 包括 `upper()`、`lower()`、`strip()`、`split()`、`replace()`、`find()` 等。例如, `"Hello".strip()` 去除字符串两端的空格, `"a,b,c".split(",")` 将字符串按逗号分割成列表 `["a", "b", "c"]`。

2.2 字节串(bytes)

字节串用于处理二进制数据, 由字节组成。

2.2.1 获取字节串实例的途径

- 字面值：使用前缀 `b` 定义，如 `b1=b'Hello'`，这里的字符必须是 ASCII 字符。
- 初始化：`bytes()` 函数可以根据可迭代对象（元素范围 0-255）创建字节串，如 `bytes([65,66,67])` 得到 `b'ABC'`；`str.encode()` 方法可以将字符串编码为字节串，如 `"你好".encode('utf-8')`。
- 运算值：支持 `+` 拼接和 `*` 重复，与字符串类似。
- 提取值：通过索引获取单个字节，如 `b'Hello'[0]` 返回 72（H 的 ASCII 码）。
- 返回值：字节串方法如 `replace()`、`find()` 会返回新的字节串。

2.2.2 字节串的属性

- 数学运算符：支持 `+` 和 `*`。
- 相等判断：使用 `==` 比较字节内容。
- 比较运算符：按字节值进行比较。
- 布尔值：非空字节串为 `True`，空字节串 `b""` 为 `False`。
- 可迭代性：可迭代，每个元素为字节值。
- 长度：`len()` 函数返回字节数。
- 常用方法：如 `decode()` 将字节串解码为字符串，`startswith()`、`endswith()` 等。

2.3 整数(int)

整数用于表示没有小数部分的数值。

2.3.1 获取整数实例的途径

- 字面值：直接书写，如 `num1=123`，也支持二进制（`0b` 前缀）、八进制（`0o` 前缀）、十六进制（`0x` 前缀）表示，如 `0b101` 表示 5。
- 初始化：`int()` 函数可以将字符串、浮点数等转换为整数，如 `int("123")` 返回 123，`int(3.14)` 向下取整为 3。
- 运算值：通过各种数学运算得到，如 `2+3`、`10-5`、`2*3`、`10/2`（结果为浮点数）、`10//3`（整除，结果为 3）、`10%3`（取余，结果为 1）、`2**3`（幂运算，结果为 8）。
- 返回值：许多数学函数和方法会返回整数结果，如 `abs(-5)` 返回 5。

2.3.2 整数的属性

- 数学运算符：支持 `+`、`-`、`*`、`**`、`/`、`//`、`%` 等。
- 相等判断：使用 `==` 判断数值是否相等。
- 比较运算符：支持 `>`、`<`、`>=`、`<=`。
- 布尔值：非零整数为 `True`，0 为 `False`。
- 可迭代性：整数本身不可迭代。
- 长度：没有长度概念，不支持 `len()` 函数。

- 常用方法：如 `bit_length()` 返回表示该整数所需的最少位数。

2.4 浮点数(float)

浮点数用于表示带有小数部分的数值。

2.4.1 获取浮点数实例的途径

- 字面值：直接书写，如 `num=3.14`，也支持科学计数法，如 `1.2e3` 表示 `1200.0`。
- 初始化：`float()` 函数可以将字符串、整数等转换为浮点数，如 `float("3.14")` 返回 `3.14`，`float(3)` 返回 `3.0`。
- 运算值：与整数类似的数学运算，结果通常为浮点数，如 `2.0+3.0`、`10.0/3.0`。
- 返回值：许多数学函数返回浮点数，如 `math.sqrt(4)` 返回 `2.0`（需导入 `math` 模块）。

2.4.2 浮点数的属性

- 数学运算符：支持各种数学运算。
- 相等判断：由于浮点数存在精度问题，直接使用 `==` 比较可能不准确，建议使用 `math.isclose()` 函数（需导入 `math` 模块）进行近似相等比较。
- 比较运算符：支持 `>`、`<`、`>=`、`<=`。
- 布尔值：非零浮点数为 `True`，`0.0` 为 `False`。
- 可迭代性：浮点数本身不可迭代。
- 长度：不支持 `len()` 函数。
- 常用方法：如 `as_integer_ratio()` 返回分数形式表示浮点数。

2.5 布尔值(bool)

布尔值只有两个取值：`True` 和 `False`，用于逻辑判断。

2.5.1 获取布尔值实例的途径

- 字面值：直接使用 `True` 和 `False`。
- 运算值：通过逻辑运算（`and`、`or`、`not`）和比较运算（`==`、`>`、`<` 等）得到，如 `3>2` 返回 `True`，`not True` 返回 `False`。
- 初始化：`bool()` 函数可以将其他类型转换为布尔值，空值（如 `""`、`[]`、`{}`、`0`、`None`）转换为 `False`，其他值转换为 `True`。

2.5.2 布尔值的属性

- 数学运算符：支持 `and`、`or`、`not` 逻辑运算。
- 相等判断：使用 `==` 判断是否为 `True` 或 `False`。
- 比较运算符：布尔值本身不支持大小比较。

- 布尔值: True 就是 True, False 就是 False。
- 可迭代性: 布尔值本身不可迭代。
- 长度: 不支持 len() 函数。
- 常用方法: 无特殊常用方法。

2.6 列表(list)

列表是一种有序、可变的数据集合, 可以存储不同类型的元素。

2.6.1 获取列表实例的途径

- 字面值: 使用方括号 [] 定义, 如 `lst=[1,2,"Hello"]`。
- 推导式: 通过列表推导式生成列表, 如 `[i for i in range(5)]` 生成 `[0,1,2,3,4]`。
- 初始化: `list()` 函数可以将可迭代对象转换为列表, 如 `list("Hello")` 得到 `['H','e','l','l','o']`。
- 运算值: 列表可以通过 + 拼接和 * 重复, 如 `[1,2]+[3,4]` 得到 `[1,2,3,4]`, `[1]*3` 得到 `[1,1,1]`。
- 提取值: 通过索引和切片操作提取元素和子列表, 如 `[1,2,3,4][1:3]` 返回 `[2,3]`。
- 返回值: 许多列表方法返回 None(原地修改列表)或新的列表, 如 `lst.append(5)` 返回 None, `sorted([3,1,2])` 返回 `[1,2,3]`。

2.6.2 列表的属性

- 数学运算符: 支持 + 拼接和 * 重复。
- 相等判断: 使用 `==` 比较列表元素是否一一对应相等。
- 比较运算符: 按元素顺序依次比较, 支持 `>`、`<`、`>=`、`<=`。
- 布尔值: 非空列表为 True, 空列表 [] 为 False。
- 可迭代性: 可迭代, 通过 `for` 循环遍历元素, 如 `for element in [1,2,3]: print(element)`。
- 长度: 使用 `len()` 函数获取列表元素个数。
- 常用方法: `append()`、`extend()`、`insert()`、`remove()`、`pop()`、`index()`、`count()`、`sort()`、`reverse()` 等。

2.7 字典(dict)

字典是一种无序、可变的键值对集合, 通过键来快速查找对应的值。

2.7.1 获取字典实例的途径

- 字面值: 使用花括号 {} 定义, 如 `d={'name':'Alice','age':25}`。
- 推导式: 字典推导式如 `{i:i**2 for i in range(5)}` 生成 `{0:0,1:1,2:4,3:9,4:16}`。

- 初始化：dict() 函数可以通过多种方式创建字典，如 dict([('name', 'Alice'), ('age', 25)]) 或 dict(name='Alice', age=25)。
- 运算值：通过赋值语句添加或修改键值对，如 d['city']='NewYork'。
- 提取值：通过键获取值，如 d['name'] 返回 'Alice'，也可以使用 get() 方法避免键不存在时报错。
- 返回值：字典方法如 keys()、values()、items() 返回可迭代对象，copy() 返回字典的副本。

2.7.2 字典的属性

- 数学运算符：字典不支持数学运算符。
- 相等判断：使用 == 比较键值对是否完全相同。
- 比较运算符：字典不支持大小比较。
- 布尔值：非空字典为 True，空字典 {} 为 False。
- 可迭代性：默认迭代键，可以通过 forkeyind: 遍历键，也可以通过 forkey,valueind.items(): 同时遍历键值对。
- 长度：使用 len() 函数获取键值对的数量。
- 常用方法：get()、update()、pop()、popitem()、clear() 等。

2.8 元组(tuple)

元组是一种有序、不可变的数据集合，适合存储不可修改的数据。

2.8.1 获取元组实例的途径

- 字面值：使用圆括号 () 定义，如 t=(1,2,"Hello")，也可以省略括号，如 t=1,2,"Hello"。
- 初始化：tuple() 函数可以将可迭代对象转换为元组，如 tuple([1,2,3]) 得到 (1,2,3)。
- 运算值：支持 + 拼接和 * 重复，如 (1,2)+(3,4) 得到 (1,2,3,4)，(1,)*3 得到 (1,1,1)（注意单元素元组需加逗号）。
- 提取值：通过索引和切片操作，与列表类似，但不能修改元素。
- 返回值：元组方法较少，如 count() 和 index()。

2.8.2 元组的属性

- 数学运算符：支持 + 和 *。
- 相等判断：使用 == 比较元素是否一一对应相等。
- 比较运算符：按元素顺序依次比较，支持 >、<、>=、<=。
- 布尔值：非空元组为 True，空元组 () 为 False。
- 可迭代性：可迭代，通过 for 循环遍历元素。

- 长度：使用 `len()` 函数获取元组元素个数。
- 常用方法：`count()`、`index()`。

2.9 集合(set)

集合是一种无序、可变、不重复的数据集合，用于去重和集合运算

编辑

分享

补充介绍一下 Python 内置对象类型中的字典

编写一段 Python 代码，演示如何使用 `type` 函数

举例说明 Python 中布尔值的使用场景