1 注释

1.1 单行注释

在Python中，#表示注释，作用于一行

```
# 这是一个单行注释
# 打印hello world
print("hello world")
```

1.2 多行注释

''' '''或者""" """表示区间注释，在三引号之间的所有内容都被注释

```
'''
这是一个多行注释，用三个单引号
这是一个多行注释，用三个单引号
这是一个多行注释，用三个单引号
'''

"""
这是一个多行注释，用三个双引号
这是一个多行注释，用三个双引号
这是一个多行注释，用三个双引号
"""
```

2 变量与赋值

1. 在使用变量之前，需要对其先赋值
2. 变量名可以包括字母、数字、下划线，但变量名不能以数字开头
3. Python变量名是大小写敏感的，foo != Foo

【例子】

```
teacher = "□□马的程序□□"
print(teacher) # □□马的程序□□
teacher = "□□马的程序□□"
print(teacher) # □□马的程序□□

first = 2
second = 3
third = first + second print(third) # 5

myTeacher = "□□马的程序□□"
yourTeacher = "□□马的程序□□"
ourTeacher = myTeacher + yourTeacher print(ourTeacher) # □□马的程序□□□□马的程序□□
```

3 运算符

3.1 算术运算符

操作符	名称	示例
+	加	1+1
-	减	2-1
*	乘	3*4
/	除	3/4
//	整除	3//4
%	取余	3%4
**	幂	2**3

【例子】

```
print(3 % 2) # 1
print(11 / 3) # 3.6666666666666665
print(11 // 3) # 3
print(2 ** 3) # 8
```

3.2 比较运算符

操作符	名称	示例
>	大于	2>1
>=	大于等于	4>=2
<	小于	1<2
<=	小于等于	2<=5
==	等于	3==3
!=	不等于	3!=5

【例子】

```
print(1 > 3) # False
print(2 < 3) # True
print(1 == 1) # True
print(1 != 1) # False
```

3.3 逻辑运算符

操作符	名称	示例
and	与	(2>1) and (3>7)
or	或	(1>3) or (2<9)
not	非	not(2>1)

【例子】

```
print((2>1) and (3>7)) #False
```

```
print((1>3) or (2<9)) #True
print(not(2>1)) #False
```

3.4 位运算符

操作符	名称	示例
~	按位取反	~4
&	按位与	4&5
|	按位或	4|5
^	按位异或	4^5
<<	左移	4<<2
>>	右移	4>>2

3.5 三元运算符

```
__ if __ else __
```

【例子】

```
x, y = 4, 5
small = x if x < y else y
print(small) # 4
```

3.6 其他运算符

操作符	名称	示例
is	是	‘hello’ is ‘hello’
not is	不是	3 is not 5
in	存在	5 in [1, 2, 3, 4, 5]
not in	不存在	2 not in [1, 2, 3, 4, 5]

【例1】

```
letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
if 'A' in letters:
  print('A' + ' exists')
if 'h' not in letters:
  print('h' + ' not exists')

# A exists
# h not exists
```

【例2】（比较的两个变量均指向不可变类型）

```
a = "hello"
b = "hello"
print(a is b, a == b)
# True True
```

【例3】（比较的两个变量均指向可变类型）

```
a = ["hello"]
b = ["hello"]
print(a is b, a == b)
# False True
```

【注意】

1. is，is not 对比的是两个变量的内存地址
2. ==，!= 对比的是两个变量的值

即：

1. 假如比较的两个变量，指向的都是地址不可变的类型（str等），那么is，is not 和 ==，!= 是完全等价的。
2. 假如比较的两个变量，指向的都是地址可变的类型（list，dict，tuple等），则两者是有区别的。

可变类型和不可变类型

Python的每个对象都分为可变和不可变，主要的核心类型中，数字、字符串、元组是不可变的，列表、字典是可变的。

对不可变类型的变量重新赋值，实际上是重新创建一个不可变类型的对象，并将原来的变量重新指向新创建的对象（如果没有其他变量引用原有对象的话（即引用计数为0），原有对象就会被回收）。

3.7 运算符的优先级

1. 一元运算符优于二元运算符。如正负号。
2. 先算术运算，后移位运算，最后位运算。如：1 << 3 + 2 & 7等价于(1 << (3 + 2)) & 7
3. 逻辑运算最后结合

【例子】

```
print(-3 ** 2) # -9
print(3 ** -2) # 0.1111111111111111
print(-3 * 2 + 5 / -2 - 4) # -12.5
print(3 < 4 and 4 < 5) # True
```

4 数据类型与转换

类型	名称	示例
int	整型	10，-34
float	浮点型	3.1492，11.11
bool	布尔型	True，False

4.1 整型int

```
a = 1031
print(a, type(a))

# 1031 <class 'int'>
```

通过print可以看出a的值是1031，以及类（class）是int。

Python里面万物皆对象（object），整型也不例外，只要是对象，就有相应的属性（attribute）和方法（methods）。

对它们有个大概印象就可以了，具体怎么用，需要哪些参数（argument），还需要查文档。看个bit_length的例子。

```
b = dir(int)
print(b)

# ['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
# '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__',
# '__float__', '__floor__', '__floordiv__', '__format__', '__ge__',
# '__getattribute__', '__getnewargs__', '__gt__', '__hash__',
# '__index__', '__init__', '__init_subclass__', '__int__', '__invert__',
# '__le__', '__lshift__', '__lt__', '__mod__', '__mul__', '__ne__',
# '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__',
# '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__',
# '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__',
# '__round__', '__rpow__', '__rrshift__', '__rshift__', '__rsub__',
# '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__', '__str__',
# '__sub__', '__subclasshook__', '__truediv__', '__trunc__', '__xor__',
# 'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag',
# 'numerator', 'real', 'to_bytes']
```

【例子】找到一个整数的二进制表示，再返回其长度。

```
a = 1031
print(bin(a)) # 0b10000000111
print(a.bit_length()) # 11
```

4.2 浮点型float

【例子】

```
print(1, type(1))
# 1 <class 'int'>

print(1., type(1.))
# 1.0 <class 'float'>

a = 0.00000023
b = 2.3e-7
print(a) # 2.3e-07
print(b) # 2.3e-07
```

有时候我们想保留浮点型的小数点后n位。可以用`decimal`包里的`Decimal`对象和`getcontext()`方法来实现。

```
import decimal
from decimal import Decimal
```

Python里面有很多用途广泛的包（package），用什么你就引入（import）什么。包也是对象，也可以用上面提到的`dir`来查看其属性和方法。比如`getcontext()`显示了`Decimal`对象的默认精度值是28位（`prec=28`），展示如下：

```
a = decimal.getcontext()
print(a)

# Context(prec=28, rounding=ROUND_HALF_EVEN, Emin=-999999, Emax=999999, capitals=1, clamp=0, flags= [], traps=[InvalidOperation, DivisionByZero, Overflow])

#1/3保留28位
b = Decimal(1) / Decimal(3)
print(b)

# 0.3333333333333333333333333333

# 1/3保留4位
decimal.getcontext().prec = 4
c = Decimal(1) / Decimal(3)
print(c)

# 0.3333
```

4.3 布尔型

布尔（boolean）型变量只能取两个值，`True`和`False`。当把布尔变量用在数字运算中，用1表示True，0表示False。

【例子】

```
print(True + True)
print(True + False)
print(True * False)

# 2
# 1
# 0
```

除了直接给变量赋值`True`和`False`，还可以用bool(X)来创建变量，其中`X`可以是：

1. 基本类型：整型、浮点型、布尔型
2. 容器类型：字符、元组、列表、字典和集合

`bool`作用在基本类型变量：`X`只要不是整型0、浮点型0.0，bool(X)就是`True`，否则就是`False`。

【例子】

```
print(type(0), bool(0), bool(1))
# <class 'int'> False True

print(type(10.31), bool(0.00), bool(10.31))
# <class 'float'> False True

print(type(True), bool(False), bool(True))
```

```
# <class 'bool'> False True
```

bool作用在容器类型变量：x只要不是空的变量，bool(X)就是True，否则就是False。

【例子】

```
print(type(''), bool(''), bool('python'))
# <class 'str'> False True

print(type(()), bool(()), bool((10,)))
# <class 'tuple'> False True

print(type([]), bool([]), bool([1, 2]))
# <class 'list'> False True

print(type({}), bool({}), bool({'a': 1, 'b': 2}))
# <class 'dict'> False True

print(type(set()), bool(set()), bool({1, 2}))
# <class 'set'> False True
```

4.4 获取类型信息

第一种方法：type(object)

【示例】

```
print(type(1)) # <class 'int'>
print(type(5.2)) # <class 'float'>
print(type(True)) # <class 'bool'>
print(type('5.2')) # <class 'str'>
```

第二种方法：isinstance(object, classinfo)

【示例】

```
print(isinstance(1, int)) # True
print(isinstance(5.2, float)) # True
print(isinstance(True, bool)) # True
print(isinstance('5.2', str)) # True
```

注：

1. type()不会认为子类是一种父类类型，不考虑继承关系
2. isinstance()会认为子类是一种父类类型，考虑继承关系

如果要判断两个类型是否相同推荐用isinstance()

4.5 类型转换

1. 转换为整型：int(x, base = 10)
2. 转换为字符串：str(object='')
3. 转换为浮点型：float(x)

【示例】

```
print(int('520')) # 520
print(int(520.52)) # 520
print(float('520.52')) # 520.52
print(float(520)) # 520.0
print(str(10 + 10)) # 20
print(str(10.1 + 5.2)) # 15.3
```

5 print()函数

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

1. 将对象以字符串表示的方式格式输出到流文件对象file里。其中所有非关键字参数都按str()方式进行转换为字符串输出；
2. 关键字参数sep是实现分隔符，比如多个参数输出时想要输出中间的分隔字符；
3. 关键字参数end是输出结束时的字符，默认是换行符\n；
4. 关键字参数file是定义流输出的文件，可以是标准的系统输出sys.stdout，也可以重定义为别的文件；
5. 关键字参数flush是立即把内容输出到流文件，不作缓存

【示例】没有参数时，每次输出后都会换行。

```
shoplist = ['apple', 'mango', 'carrot', 'banana'] print("This is printed without 'end'and 'sep'.")
for item in shoplist:
  print(item)

# This is printed without 'end'and 'sep'.
# apple
# mango
# carrot
# banana
```

【示例】每次输出结束都用end设置的参数&结尾，并没有默认换行。

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
print("This is printed with 'end='&''.")
for item in shoplist:
  print(item, end='&')
print('hello world')

# This is printed with 'end='&''.

# apple&mango&carrot&banana&hello world
```

【示例】item值与another string两个值之间用sep设置的参数&分割。由于end参数没有设置，因此默认是输出后换行，即end参数的默认值为\n。

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
print("This is printed with 'sep='&''.")
for item in shoplist:
  print(item, 'another string', sep='&')

# This is printed with 'sep='&''.
```

```
# apple&another string
# mango&another string
# carrot&another string
# banana&another string
```

6 习题

1. 怎样对python中的代码进行注释？

   ```
   # 单行注释
   """
   多行注释
   多行注释
   """
   '''
   多行注释
   多行注释
   '''
   ```

2. python有哪些运算符，这些运算符的优先级是怎么样的？

   运算符：

   - 算术运算符（+, -, *, /, //, %, **）
   - 比较运算符（>, >=, <, <=, ==, !=）
   - 逻辑运算符（and, or, not）
   - 位运算符（~, &, |, ^, <<, >>）
   - 其他运算符（is, not is, in, not in）

   优先级：

   - 一元运算符优于二元运算符。
   - 先算术运算，后移位运算，最后位运算。
   - 逻辑运算最后结合。

3. pythonz`is`，`is not`与`==`，`!=`的区别是什么？

   - 假如比较的两个变量，指向的都是地址不可变的类型（str等），那么is，is not 和 ==，!= 是完全等价的。
   - 假如比较的两个变量，指向的都是地址可变的类型（list，dict，tuple等），则两者是有区别的。

4. python中包含哪些数据类型？这些数据类型之间如何转换？

   数据类型：

   - int、float、bool、str

   数据类型的转换：

   - 转换为整型：int(x, base = 10)
   - 转换为字符串：str(object='')
   - 转换为浮点型：float(x)