Lesson 2: 数据类型与基础IO

数据

我们需要考虑的

- 数据的类型
- 数据的流动
- 数据的存储
- 数据的组织

数据类型

你的任何数据在计算机中最终都是一串01,所以我们需要数据类型这一概念来告诉计算机(其实也是我们自己)这些数据的含义

- 20220715的可能含义?
- 1 + 1 与 '1' + '1'?

数据类型也在一定程度上标识了数据应该支持的运算

```

```

基础类型

- 布尔值: 通常由表达式返回,或由其他变量强制转换而来

```
>>> 2 >= 1

True # 返回了表达式的值

>>> 2 <= 1

False

>>> a = 1

>>> b = 1

>>> a == b
```

```
True

>>> not a > b

True

>>> a > b and 2 >= 1

False

>>> a > b or 2 >= 1

True
```

- 整形: 即整数,支持加减乘除、整除、求余

```
>>> 1 + 1

2

>>> 1 / 2 # / 表示普通除法，会返回准确值

0.5

>>> 2 / 2 # 普通除法返回的一定是浮点数

1.0

>>> 1 // 2 # // 表示整数除法

0

>>> -3 // 2 # // 负数的整数除法同样应该向下取整

-2

>>> 5 % 2 # % 表示求余

1

>>> -7 % 3 # 负数的求余同样需要向下取
```

```
2

>>> ((1 + 5) // 3) % 2 + 1 # 不管多么复杂的嵌套，你都只能用小括号来表示计算优先级

1
```

- 浮点: 用于 *模拟* 实数，模拟意味着会有精度的损失，但大多数时候你不需要考虑这个问题

浮点数同样支持加减乘除，但不建议对浮点数进行整除和求余

```
>>> a = 6 / 5 # 赋值操作

>>> b = 7 / 3

>>> a + b # 这是一个表达式

3.533333333333333 # 返回表达式的值
```

- 空值: 不同于0也不同于False，是空，是没有数据的空

通常用于表示该变量没有接收到可用的值

```
>>> a = None

a
```

- 字符串: 处理字符串是python的强项

```
>>> s1 = 'hello' # 你可以用单引号或双引号来标识一个字符串

'hello' # 但引号本身并不属于字符串

>>> s1 = hello

Traceback...
```

```
>>> s2 = '\'world\'' # 当你真的需要一个引号时，在引号前加一个\表示转义

"'world'"

>>> s3 = 'hello' + 'world' # 字符串的加法表示对字符串进行拼接

>>> s3

helloworld

>>> s4 = 'hello' + ',' + 'world' + '!'

>>> s4

'hello,world!'
```

字符串的其他特性

- 索引与切片:我们有时只关心字符串中的一部分(如身份证号的特定位数，姓名中的姓氏)

```
>>> a = 'hello'

>>> a[1] # 索引从0开始!

'e'

>>> a[-1] # 负数表示从后往前

'o'

>>> a[1:3] # 左闭右开!

'el'

>>> a[1:]

'ello'

>>> a[:-1]
```

```
'hell'
```

- 计算长度

```
>>> a = 'hello'

>>> len(a)

5
```

- 拆分: 我们希望将一个字符串拆成多个字符串来分析，我们将在讲列表时讨论split方法

```

```

数据的流动

输出

将程序中的涉及的数据以你需要的形式输出到终端/文件/其他地方中，起到可视化或永久化的效果。我们暂时仅讨论把数据输出到终端

```
# print.py



print('hello world') # 直接输出给定字符串

label = 'shirt'

print(label) # 直接输出变量的值

score = 89

print(f'you got {score}!') # 当你需要根据变量来输出内容时，用f和{}来在输出中插入变量
```

输入

我们希望从文件或键盘缓冲区中读取数据，从而与用户进行交互。我门暂时仅讨论从键盘缓冲区中输入

```
>>> name = input('input your name:')

>>> name

...




>>> score = input('input your score:') # 猜猜此时的score的类型?

>>> score

...

>>> score + 1

Traceback...

### 可见输入数字时，score仍被认为是字符串




>>> score = int(score) # 强制转换为我们需要的类型

>>> score += 1

>>> print(f'score is {score}')

...
```

变量与值

```
# value.py



today = 15 # 初始化变量today,tomorrow中的值

tomorrow = 16

print(f'today is {today}, tomorrow is {tomorrow}\n') # \n用于换行



today = tomorrow # 把储存在tomorrow中的值复制到today中

tomorrow = today + 1 # 把储存在today中的值加1后赋给tomorrow

print(f'today is {today}, tomorrow is {tomorrow}\n')



today = today + 1 # 把today中的值加1后赋给today

tomorrow += 1 # 是上一种写法的简写，类似的运算符有-= %= /= *=

print(f'today is {today}, tomorrow is {tomorrow}\n')
```

- 变量命名原则: 有实际含义或按照特定命名习俗

```
batch_size

learning_rate

num_days

cursor

filter

treenode
```

```
grid

output
```

- 变量赋值原则: 尽量不要改变变量初始化时的类型(空值除外)

```
# bad_example.py



date = '20220715'

date = int(date) + 1



# 可能造成的问题:当程序的其他部分需要用到date中的数据时无法确定其类型
```

- 其他注意点: 注意字符串的引号，注意不要随意覆盖原有变量储存的值

```
# bad_swap.py

a = int(input('input the value of a'))

b = int(input('input the value of b'))

a = b

b = a

print(f'a={a},b={b}')
```

```
# swap.py

a = int(input('input the value of a'))

b = int(input('input the value of b'))
```

```
temp = a

a = b

b = temp

print(f'a={a},b={b}')
```

课后习题

1. 编写一个程序，引导用户输入身份证号码，然后输出用户的出生年、月、日
2. 编写一个程序，引导用户输入两个整数，输出两个整数的积的位数与最低位的值
3. 编写一个程序，实现三个整数的轮换(1->2->3->1)