#### 1.python基本数据类型

**重点**：掌握常用的python基本数据类型，数据类型之间的转换，掌握python中的基础运算符

上篇文章我们提出了"变量 = 值"的目标，今天我们看一下什么是一个合法的值。首先是python中的基本数据类型，就是一个合法的值。

##### 1.1.数字类型（Number）

###### 1.1.1.整形，int

表示整数

int1 = 10  
int2 = -10

###### 1.1.2.浮点数，float

表示带有小数的数值

float1 = 3.14  
float2 = -1.23

###### 1.1.3复数，complex

表示复数

complex1 = 3+4j  
complex2 = 5-6j

##### 1.2.序列类型（Sequence Type）

###### 1.2.1字符串，str

表示文本数据

str1 = 'huahai2022'  
str2 = 'zhangyang2022'

###### 1.2.2.列表，list

表示有序的元素集合

list1 = [1, 2, 3, 4, 5]  
list2 = ["dog", "cat", "bird", "fish", "horse"]

###### 1.2.3.元组，tuple

表示不可变的有序元素集合

tuple1 = (1, 2, 3, 4, 5)  
tuple2 = ("dog", "cat", "bird", "fish")

##### 1.3.映射类型（mapping Type）

###### 1.3.1.字典，dict

表示键值对的集合

dict1 = {'a': 1, 'b': 2}  
dict2 = {'b': 3, 4: "huahai2022"}

##### 1.4.集合类型（Set Type）

###### 1.4.1.集合，set

表示无序且元素唯一的集合

set1 = {1, 2, 3}  
set2 = {3, 4, 5, 5}  
print(set2)

###### 1.4.2.冻结集合，frozenset（不常用）

表示不可变的集合类型

frozenset1 = frozenset([2, 3, 4, 4])  
print(frozenset1)

##### 1.5.布尔类型（boolean Type）

###### 1.5.1布尔

标识逻辑值True或者False

isTure = True  
isFalse = False

#### 2.python基本数据类型的应用

##### 2.1.整形

int1 = 10 #默认是十进制  
int2 = 0b1010 #二进制，0b  
int3 = 0o763 #八进制，0o  
int4 = 0xABCD #十六进制，0x

可以用其它进制来表示整数。

##### 2.2.浮点型

float1 = 0.0  
float2 = 3.3e10 #科学计数法，相当于10的10次方  
float3 = -21.9

##### 2.3.复数

complex1 = 3.14j  
complex2 = 2e10-2.1j

##### 2.4.字符串

str1 = "huahai2022"

以上面的字符串为例，它是由一个一个字符组成的，我们可以任意拼接这些字符来组成字符串，如何取出字符串的字符呢？我们使用字符串的索引，如下图所示。

| h | u | a | h | a | i | 2 | 0 | 2 | 2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

#取出前三个字符  
print(str1[0:3])  
#将字符串重复两遍  
print(str1\*2)  
#将字符串后面加上"2024"  
print(str1 + "2024")  
#将字符串后面加上""2024""  
print(str1 + ""2024"") ?

我们可以看到这里会报错，原因是python解释器把""当作引号，而不是字符串的一部分，如何避免“尴尬”呢？转移字符登场，顾名思义，转义字符就是给字符变一个含义。

###### 2.4.1.常见的转义字符

\\ - 反斜杠字符 \。  
print(r"hu\\ahai2022")  
\' - 单引号字符 '。  
print(r"\'huahai2022\'")  
\" - 双引号字符 "。  
print(r"\"huahai2022\"")  
\n - 新行（换行）字符。  
print(r"\nhuahai2022")  
\r - 回车字符。  
print(r"\rhuahai2022")  
\t - 制表符。  
print(r"\thuahai2022")

但是如果我不想让转义字符转移，比如说我就想print("\rhuahai2022")的结果是\rhuahai2022，这个时候只需要将字符串前面加上r即可。print(r"\rhuahai2022")。r标识Raw。

###### 2.4.2.字符串运算符

1. 字符串连接（+）：  
 使用加号运算符可以将两个字符串连接在一起，生成一个新的字符串。  
 str1 = "Hello"  
 str2 = "World"  
 result = str1 + " " + str2  
 print(result) # 输出: "Hello World"  
  
2. 重复字符串（\*）：  
 使用乘号运算符可以将一个字符串重复多次。  
 str1 = "Hello"  
 repeatedStr = str1 \* 3  
 print(repeatedStr) # 输出: "HelloHelloHello"  
  
3. 切片（[]）：  
 使用方括号运算符可以从字符串中提取子字符串，通过指定起始索引和结束索引进行切片操作。  
 myString = "Hello World"  
 subString = myString[6:11]  
 print(subString) # 输出: "World"  
  
4. 成员运算符（in、not in）：  
 使用`in`和`not in`运算符可以判断一个字符串是否包含另一个子字符串。  
 myString = "Hello World"  
 print("Hello" in myString) # 输出: True  
 print("Python" not in myString) # 输出: True  
  
5. 比较运算符（==、!=、<、>、<=、>=）：  
 使用比较运算符可以比较两个字符串的大小关系（基于字典顺序）或判断两个字符串是否相等。  
 str1 = "apple"  
 str2 = "banana"  
 print(str1 < str2) # 输出: True  
 print(str1 == "Apple") # 输出: False  
  
  
6. 格式化运算符（%）：  
 使用百分号运算符可以将数据插入到格式化字符串中，通过占位符指定插入的位置和数据类型。  
 name = "huahai2022"  
 age = 23  
 result = "My name is %s and I am %d years old." % (name, age)  
 print(result) # 输出: "My name is huahai2022 and I am 23 years old."

###### 2.4.3.格式化字符串

格式化字符串时可以使用一些格式化符号来指定要插入的数据的类型、宽度、精度

1. `%s`：字符串格式化符号，用于插入字符串。  
name = "huahai2022"  
result = "My name is %s." % name  
print(result) # 输出: "My name is huahai2022."  
  
2. `%d`：整数格式化符号，用于插入整数。  
age = 23  
result = "I am %d years old." % age  
print(result) # 输出: "I am 19 years old."  
  
3. `%f`：浮点数格式化符号，用于插入浮点数。  
pi = 3.14159  
result = "The value of pi is %f" % pi  
print(result) # 输出: "The value of pi is 3.141590."  
  
4. `%.<precision>f`：带有精度的浮点数格式化符号，用于指定浮点数的小数位数。  
pi = 3.14159  
result = "The value of pi is %.2f." % pi  
print(result) # 输出: "The value of pi is 3.14."  
  
5. `%x` 和 `%X`：十六进制整数格式化符号，用于插入十六进制整数。  
number = 16  
result = "The hexadecimal representation of %d is %x." % (number, number)  
print(result) # 输出: "The hexadecimal representation of 16 is 10."  
  
6. `%o`：八进制整数格式化符号，用于插入八进制整数。  
number = 8  
result = "The octal representation of %d is %o." % (number, number)  
print(result) # 输出: "The octal representation of 8 is 10."  
  
7.更简单的格式化字符串，如果我们不想判断插入的数据类型是什么，我们可以直接使用f"{插入的数据}"  
pi = 3.14159  
result = f"The value of pi is {pi:.2f}."  
print(result) # 输出: "The value of pi is 3.14."  
  
name = "huahai2022"  
age = 19  
result = f"My name is {name} and I am {age} years old."  
print(result) # 输出: "My name is huahai2022 and I am 19 years old."

###### 2.4.4.常用的内置方法

大小写转换、去除空格、分割和连接字符串、替换子字符串等

1. `len()`: 返回字符串的长度。  
my\_string = "Hello World"  
length = len(my\_string)  
print(length) # 输出: 11  
  
2. `lower()`: 将字符串转换为小写。  
my\_string = "Hello World"  
lowercase\_string = my\_string.lower()  
print(lowercase\_string) # 输出: "hello world"  
  
3. `upper()`: 将字符串转换为大写。  
my\_string = "Hello World"  
uppercase\_string = my\_string.upper()  
print(uppercase\_string) # 输出: "HELLO WORLD"  
  
4. `capitalize()`: 将字符串的第一个字符转换为大写，其余字符转换为小写。  
my\_string = "hello world"  
capitalized\_string = my\_string.capitalize()  
print(capitalized\_string) # 输出: "Hello world"  
  
5. `title()`: 将字符串中每个单词的首字母转换为大写。  
my\_string = "hello world"  
titlecased\_string = my\_string.title()  
print(titlecased\_string) # 输出: "Hello World"  
  
6. `strip()`: 去除字符串两端的空白字符。  
my\_string = " Hello World "  
stripped\_string = my\_string.strip()  
print(stripped\_string) # 输出: "Hello World"  
  
7. `split()`: 将字符串分割成一个列表，使用指定的分隔符。  
my\_string = "Hello,World"  
split\_string = my\_string.split(",")  
print(split\_string) # 输出: ["Hello", "World"]  
  
8. `join()`: 将一个可迭代对象的字符串元素连接起来，使用指定的分隔符。  
my\_list = ["Hello", "World"]  
joined\_string = ",".join(my\_list)  
print(joined\_string) # 输出: "Hello,World"  
  
9. `replace()`: 将字符串中的指定子字符串替换为另一个子字符串。  
my\_string = "Hello World"  
new\_string = my\_string.replace("World", "Python")  
print(new\_string) # 输出: "Hello Python"

##### 2.5.列表

list1 = ["huahai2022", 2024, "JJBond", 3.14]

以上面列表为例，里面可以装字符串或者数字。类比字符串我们如何取里面的值呢。相比你一定想到了用索引。

| huahai2022 | 2024 | JJBond | 3.14 |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| -4 | -3 | -2 | -1 |

这样我们就可以访问列表中的值了

#输出huahai2022  
print(list1[0])  
#输出huahai2022中的2022  
print(list1[0][0:6])

列表是python中常见的一种数据结构，数据结构一般都要提供增删改查的功能，列表也不例外。怎么往列表中添加删除数据呢？

下面是列表一些常用的函数来统计列表内容。

| 函数 | 介绍 | 举例 | 备注 |
| --- | --- | --- | --- |
| len(list) | 展示列表中元素个数 | len(list1) |  |
| max(list) | 输出列表中的最大值 | max(list1) | 列表中的元素必须属于同一数据类型，或者不同数据类型定义了比较方法 |
| min(list) | 输出列表中的最大值 | min(list2) | 列表中的元素必须属于同一数据类型，或者不同数据类型定义了比较方法 |

下面是列表的方法用来对列表中的内容进行操作。

| 方法 | 介绍 | 举例 | 备注 |
| --- | --- | --- | --- |
| list.append(obj) | 将对象添加到列表末尾 | list1.append(520) | 增 |
| list.insert(index,obj) | 将对象添加到index位置 | list1.insert(1,"boboji") | 增 |
| list.extend(list)或者list+list | 将列表添加到原列表末尾 | list1.extend(list1) | 增 |
| list.remove(obj) | 移除列表中第一个obj对象 | list1.remove("huahai2022") | 删 |
| list.pop([index]) | 移除index位置处的元素 | list1.pop([1:2]) | 删 |
| list.count(obj) | 查找列表中obj出现的次数 | list1.count("huahai2022") | 查 |
| list.index(obj) | 查找列表中obj的索引 | list1.index("huahai2022") | 查 |
| list.sort(key=None,reverse=True) | 对原列表进行排序 | list1.sort(reverse=False) | 改 |
| list.reverse() | 对原列表中的元素进行翻转 | list1.reverse() | 改 |
| list.copy() | 复制列表 | list1.copy() |  |
| list.clear() | 清空列表 | list1.clear() |  |

##### 2.6.元组

tuple1 = ("huahai2022", "ronaldo", "kunkun", 888, "Jack")

可以看到元组使用()来定义，列表使用[]来定义，它们之间最大的区别就是元组中的元素是不可修改的，而列表中的元素是可以修改的。

同样的，元组也存在索引。

| "huahai2022" | "ronaldo" | "kunkun" | 888 | "Jack" |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| -5 | -4 | -3 | -2 | -1 |

#显示元组第一个元素  
print(tuple1[0])  
#修改元组第一个元素为“huahai”  
? 不允许

常见的函数及其作用

| 函数 | 介绍 | 举例 | 备注 |
| --- | --- | --- | --- |
| len(tuple) | 统计元组中的元素个数 | len(tuple1) | 5 |
| max(tuple) | 给出元组中的元素最大值 | max(tuple1) | 列表中的元素必须属于同一数据类型，或者不同数据类型定义了比较方法 |
| min(tuple) | 给出元组中的元素最小值 | min(tuple1) | 列表中的元素必须属于同一数据类型，或者不同数据类型定义了比较方法 |

**对元组的误解**：元组中的元素是不可变的，不代表元组是不可变的。

tuple1 = (1, 2, 3, 4, 5)  
print(id(tuple1))  
tuple1 = (1,2)  
print(id(tuple1))

##### 2.7.字典

myDict = {'name': 'huahai2022', 'likes': "code", 'age': 21}

字典是由键值对(key-value)组成的。

字典的特性：

* 键必须是唯一的，且不可修改，只能删除或者增加；而值不一定是唯一的，且只可以修改。
* 键可以是数字，字符串或者元组，但不能是列表，键是不可变的。

#向字典中添加新的键值对  
myDict['job'] = "code"  
#修改字典中的值  
myDict['age'] = '22'  
#删除键值对  
del myDict['age']  
#统计字典中的键值对  
print(len(myDict))  
#打印字典中的键  
print(myDict.keys())  
#打印字典中的值  
print(myDict.values())  
#判断键是否在myDict中  
print("gender" in myDict)  
#返回键值对对象  
print(myDict.items())  
#获取指定键的值，如果不存在，返回false  
print(myDict.get("height",default=false))

遍历字典的方式

myDict = {'name': 'huahai2022', 'likes': "code", 'age': 21}  
for key,value in myDict.items():  
 print("key:",key, "value", value)

myDict = {'name': 'huahai2022', 'likes': "code", 'age': 21}  
for key in myDict.keys():  
 print("key:",key, "value", myDict[key])

##### 2.8.集合

Python中的集合（set）是一个无序的不重复元素集。集合是可变的，可以添加或删除元素。集合不支持索引，因为它们是无序的。集合主要用于消除重复元素。

#创建集合  
mySet = {1, 2, 3, 4, 5}  
#创建空集合（不能使用{}，因为它会创建一个空字典）  
emptySet = set()  
#添加元素  
mySet.add(6)  
#移除元素  
mySet.remove(6)  
#判断元素是否在集合中  
print(2 in mySet)  
#遍历集合  
for ele in mySet:  
 print(ele)

集合的并集、交集、差集

a = {1, 2, 3}  
b = {3, 4, 5}  
#并集  
unionSet=a|b #或者a.union(b)  
#交集  
interSet=a&b #或者a.intersection(b)  
#差集  
diffSet=a-b #或者a.difference(b)

#### 3.python基本数据类型的转换

不同数据类型之间怎么进行操作呢？这里就需要进行基本数据类型的转换。分为两种，一种是隐式数据类型转换，一种是显式数据类型转换，顾名思义，隐式数据类型是自动完成的，不需要手动转换，显式数据类型需要我们手动完成。

##### 3.1.隐式数据类型转换（不推荐使用）

常有以下情况：

1.整数和浮点数运算：当整数和浮点数进行运算时，整数会隐式转换为浮点数  
int1 = 5  
float1 = 3.14  
result = int1 + float1  
print(result) # 输出: 8.14  
2.整数和布尔值进行运算，布尔值隐式转换为整数，false表示0，true表示1  
int1 = 5  
isTrue = True  
result = int1 + isTrue  
print(result) # 输出: 6

##### 3.2.显式数据类型转换

在Python中，可以使用一些内置函数进行基本数据类型的转换。  
  
1. 转换为整数（int）类型：str-->int  
str1 = "10"  
int1 = int(str1)  
print(int1)   
  
2. 转换为浮点数（float）类型：str-->float  
num\_str = "3.14"  
num\_float = float(num\_str)  
print(num\_float)   
  
3. 转换为字符串（str）类型：int-->str  
num\_int = 42  
num\_str = str(num\_int)  
print(num\_str)   
  
4. 转换为布尔（bool）类型：int-->bool  
zero = 0  
nonzero = 10  
zero\_bool = bool(zero)  
nonzero\_bool = bool(nonzero)  
print(zero\_bool) # 输出: False  
print(nonzero\_bool) # 输出: True  
  
5. 转换为列表（list）类型：tuple-->list  
my\_tuple = (1, 2, 3)  
my\_list = list(my\_tuple)  
print(my\_list) # 输出: [1, 2, 3]  
  
  
6. 转换为元组（tuple）类型：list-->tuple  
my\_list = [1, 2, 3]  
my\_tuple = tuple(my\_list)  
print(my\_tuple) # 输出: (1, 2, 3)  
  
  
7. 转换为集合（set）类型：list-->set  
my\_list = [1, 2, 2, 3, 3, 3]  
my\_set = set(my\_list)  
print(my\_set) # 输出: {1, 2, 3}  
  
  
8. 转换为字典（dict）类型：list-->dict  
my\_list = [("name", "Alice"), ("age", 25)]  
my\_dict = dict(my\_list)  
print(my\_dict) # 输出: {"name": "Alice", "age": 25}

#### 4.python运算符

我们掌握了值得一些操作，但是两个值之间可以进行哪些运算呢？让我们一起了解一下python运算符。主要可以分为算术运算符，比较运算符

##### 4.1.算术运算符

`+`：加法  
x = 5  
y = 3  
result = x + y  
print(result) # 输出: 8  
  
  
`-`：减法  
x = 5  
y = 3  
result = x - y  
print(result) # 输出: 2  
  
`\*`：乘法  
x = 5  
y = 3  
result = x \* y  
print(result) # 输出: 15  
  
`/`：除法（返回浮点数）  
x = 5  
y = 3  
result = x / y  
print(result) # 输出: 1.6666666666666667  
  
`//`：除法（返回整数部分）  
x = 5  
y = 3  
result = x // y  
print(result) # 输出: 1  
  
`%`：取模（返回余数）  
x = 5  
y = 3  
result = x % y  
print(result) # 输出: 2  
  
`\*\*`：幂运算  
x = 2  
y = 3  
result = x \*\* y  
print(result) # 输出: 8

##### 4.2.比较运算符

`==`：等于  
x = 5  
y = 3  
result = x == y #(运算符优先级)  
print(result) # 输出: False  
  
`!=`：不等于  
x = 5  
y = 3  
result = x != y  
print(result) # 输出: True  
  
`>`：大于  
x = 5  
y = 3  
result = x > y  
print(result) # 输出: True  
  
  
`<`：小于  
x = 5  
y = 3  
result = x < y  
print(result) # 输出: False  
  
`>=`：大于等于  
x = 5  
y = 3  
result = x >= y  
print(result) # 输出: True  
  
`<=`：小于等于  
x = 5  
y = 3  
result = x <= y  
print(result) # 输出: False

##### 4.3.逻辑运算符

`and`：与  
x = True  
y = False  
result = x and y  
print(result) # 输出: False  
  
`or`：或  
x = True  
y = False  
result = x or y  
print(result) # 输出: True  
  
`not`：非  
x = True  
result = not x  
print(result) # 输出: False

##### 4.4.赋值运算符

赋值运算符：  
`=`：赋值  
x = 5  
  
  
`+=`：加法赋值  
x = 5  
x += 2 # 等同于 x = x + 2  
print(x) # 输出: 7  
  
`-= `：减法赋值  
x = 5  
x -= 2 # 等同于 x = x - 2  
print(x) # 输出: 3  
  
`\*= `：乘法赋值  
x = 5  
x \*= 2 # 等同于 x = x \* 2  
print(x) # 输出: 10  
  
`/= `：除法赋值  
x = 5  
x /= 2 # 等同于 x = x / 2  
print(x) # 输出: 2.5  
  
`//= `：除法赋值（整数部分）  
x = 5  
x //= 2 # 等同于 x = x // 2  
print(x) # 输出: 2  
  
`%= `：取模赋值  
x = 5  
x %= 2 # 等同于 x = x % 2  
print(x) # 输出: 1  
  
`\*\*= `：幂运算赋值  
x = 2  
x \*\*= 3 # 等同于 x = x \*\* 3  
print(x) # 输出: 8

##### 4.5.位运算符

针对二进制

1. `&`：按位与  
x = 5 # 二进制表示为 0101  
y = 3 # 二进制表示为 0011  
result = x & y  
print(result) # 输出: 1 (二进制表示为 0001)  
  
2. `|`：按位或  
x = 5 # 二进制表示为 0101  
y = 3 # 二进制表示为 0011  
result = x | y  
print(result) # 输出: 7 (二进制表示为 0111)  
  
3. `^`：按位异或  
x = 5 # 二进制表示为 0101  
y = 3 # 二进制表示为 0011  
result = x ^ y  
print(result) # 输出: 6 (二进制表示为 0110)  
  
4. `~`：按位取反  
x = 5 # 二进制表示为 0101  
result = ~x  
print(result) # 输出: -6 (二进制表示为 -0110)  
  
5. `<<`：左移  
x = 5 # 二进制表示为 0101  
result = x << 2  
print(result) # 输出: 20 (二进制表示为 10100)  
  
6. `>>`：右移  
x = 5 # 二进制表示为 0101  
result = x >> 1  
print(result) # 输出: 2 (二进制表示为 0010)

##### 4.6.运算符优先级

1. 括号：`()`，括号中的表达式具有最高的优先级，可以用于控制运算的顺序。  
  
2. 幂运算：`\*\*`，幂运算具有高于其他算术运算符的优先级。  
  
3. 一元正号和一元负号：`+`、`-`，用于表示正数和负数。  
  
4. 乘法、除法、取模和整除运算：`\*`、`/`、`%`、`//`，它们具有相同的优先级，并按照从左到右的顺序进行计算。  
  
5. 加法和减法运算：`+`、`-`，它们具有相同的优先级，并按照从左到右的顺序进行计算。  
  
6. 位运算符：`&`、`|`、`^`、`<<`、`>>`，它们具有相同的优先级，但低于算术运算符。  
  
7. 比较运算符：`==`、`!=`、`>`、`<`、`>=`、`<=`，比较运算符的优先级低于算术和位运算符。  
  
8. 逻辑运算符：`and`、`or`、`not`，逻辑运算符的优先级低于比较运算符。  
  
9. 赋值运算符：`=`、`+=`、`-=`、`\*=`、`/=` 等，赋值运算符的优先级最低。

对于拿不准优先级的运算符，建议多加小括号。