人工智能课程

Python

目录

CONTENTS

01 数据类型-数字

数据类型—数字型

数字型：在Python中，数字型数据包括以下几种：

1. 整型(int)：包括正整数、负整数和0，在程序中的表达与数学上一模一样。
2. 浮点型(float)：其实就是小数。
3. 布尔型(bool)：特殊的整形，只有True和False，True为1，False为0。
4. 复数(complex)：与数学上的表达一致，分为实部和虚部，虚部用j或J表示，多用于科学计算。

整数的特性与范围

- Python中的整数支持各种运算，如加法、减法、乘法、除法等，还可以进行位运算等。
- 小整数池是Python解释器为了提高效率和节省内存而实现的一种优化策略。在Python中，整数是不可变的对象，一旦创建，其值不能被修改。由于一些小的整数在很多程序中用到的频率很高，因此解释器会在启动时预先创建这些小整数对象，以避免频繁地创建和销毁小整数对象所带来的性能问题和内存浪费，其范围为[-5, 256]。
- Python中的没有固定的取值范围，因为它是动态类型的，并且其大小是根据需要自动调整的。这意味着Python的整数可以非常大或非常小，只受限于可用内存。

浮点数

浮点数的存储方式：事实上直到20世纪80年代，还是计算机厂商各自为战，每家都在设计自己的浮点数存储规则，彼此之间并不兼容。直到1985年，IEEE754标准问世，浮点数的存储问题才有了一个通用的工业标准。

IEEE 754标准

浮点数的表示

IEEE 754标准规定了浮点数的表示方法，包括符号位、阶码和尾数位。

浮点数的存储

存储float类型数据需要64个bit，其中最高位是符号位，然后11位是阶码，剩余的52位是尾数位。

特殊值

IEEE 754标准还规定了一些特殊值，如无穷大、无穷小和NaN（不是数字）。

布尔型数据的概念

定义

只有True和False两个值，分别表示真和假。

运算

布尔型数据可以进行逻辑运算，如and、or、not等，用于判断多个条件的真假。

应用

布尔型数据在Python编程中广泛应用于条件判断、循环控制等场景，是编程中不可或缺的数据类型。

布尔型数据的短路求值

短路求值原理

在Python中，布尔型数据的短路求值是一种特殊的逻辑运算规则，当逻辑表达式的某个部分已经能够确定整个表达式的值时，就不再计算表达式的其余部分。

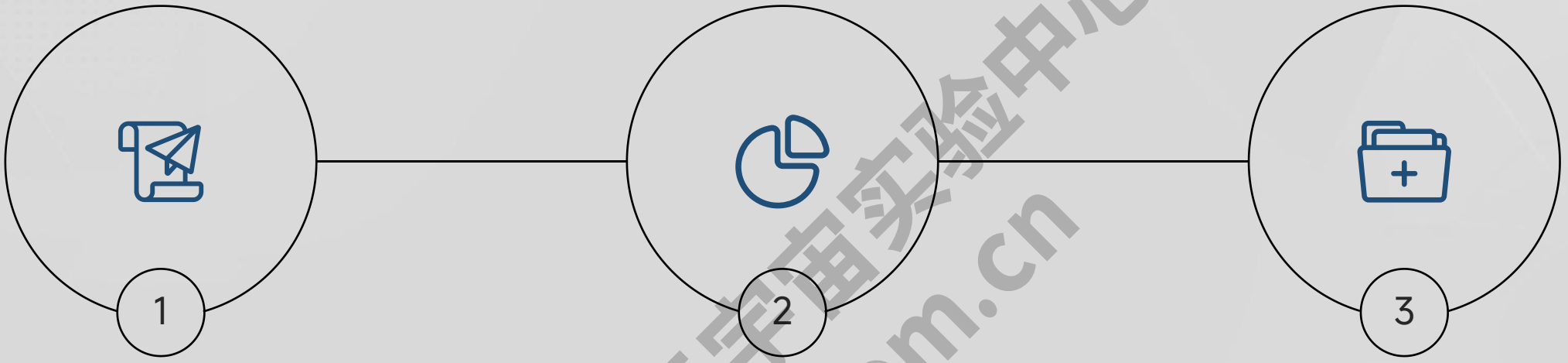
短路求值的应用

在Python编程中，短路求值多用于逻辑判断中，可以用于简化代码和提高效率。

短路求值的示例

计算表达式“a and b”的结果，如果a是False，那么b就不再进行计算，而是直接返回False。

复数的表示形式



实部与虚部

复数是由实部和虚部组成的，其中实部是复数的一部分，虚部是复数的另一部分。

复数的表示方法

复数可以用多种方法表示，例如用“a+bi”表示，其中a是实部，b是虚部，i是虚数单位。

复数的运算

复数可以进行加法、减法、乘法和除法等运算，这些运算遵循一定的规则，

复数的应用场景

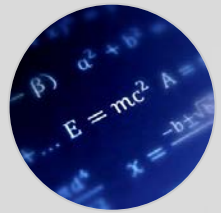
在物理学、工程学等领域，复数被广泛用于描述和计算涉及旋转、振动等现象的问题。

科学计算

在信号处理领域，复数被用于分析、处理和传输信号，如音频、视频等。

信号处理

在数学建模中，复数被用于构建和求解各种数学问题，如微分方程、积分方程等。



数学建模

02 数据类型-字符串

数据类型—字符串

定义

由引号引起来的一系列数字、字母、符号及中文的组合，并且定义好的字符串是不可修改的。

创建

使用一对单引号或双引号可以创建一个只有一行的字符串。

使用一对三引号可以创建一个很多行的字符串

注意

引号不能混合使用，且引号只是字符串的一个表达方式，本身不属于字符串内容。

数据类型—字符串

转义字符：是一种特殊字符，用于表示无法直接表示的字符，以反斜杠“\”开头。

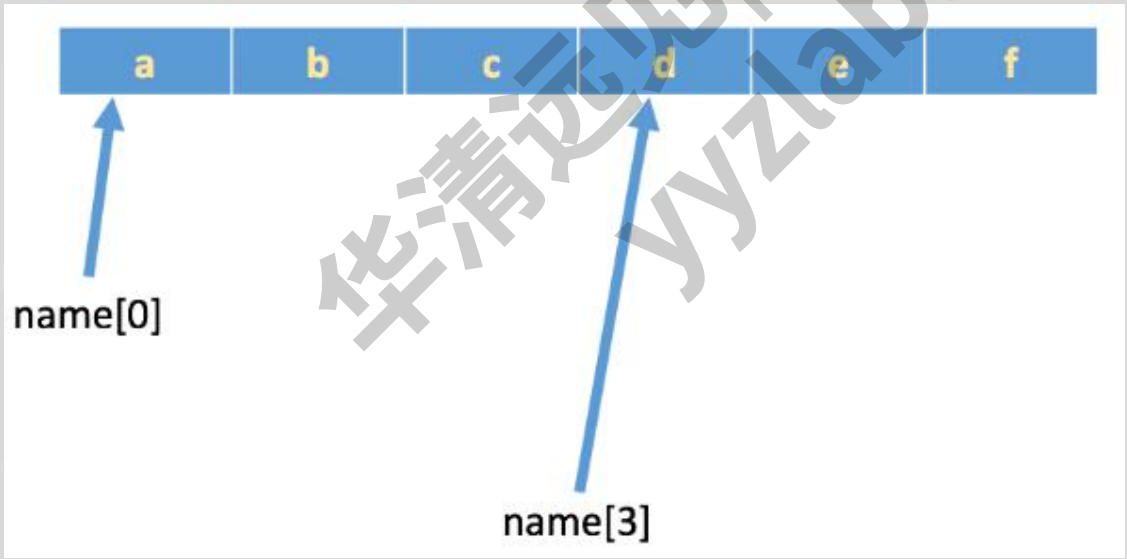
常用的转义字符：

- 换行符：\n，用于实现换行。
- 制表符：\t，相当于一个Tab。
- 回车符：\r，将光标移至当前行的开头。
- 反斜杠：\\，将反斜杠本身转义，使反斜杠本身成为一个普通字符。
- 单引号与双引号：\’和\”，将单引号与双引号转义，使其不再是字符串的标识，而是仅仅只是一个单引号或双引号。

数据类型—字符串

字符串的访问与操作：下标访问与切片访问。

下标访问：所谓的下标，其实就是编号，通过编号就可以找到对应的字符，下标可按照从左至右的顺序开始计算，也可以按照从右至左的顺序开始计算，但是访问的时候下标不能超出范围。



数据类型—字符串

切片访问与下标访问类似，都是通过字符串的下标进行的，不同的是，下标访问每次只能访问到单个字符，切片访问可以一次访问到多个字符，其访问方式为：

字符串名[初始位置：终止位置：步长]

访问时，包括初始位置不包括终止位置，且步长默认为1。如果没有给出初始位置，默认初始位置为开始位置；如果没有给出终止位置，默认终止位置为字符串结束位置，此时访问时包括终止位置。

数据类型—字符串

两个字符串的比较：

- 字符串的比较是逐个字符比较，从字符串的第一个字符开始比较，如果两个字符的Unicode编码值相同，则继续比较下一个字符，直到所有字符都比较完毕。

数据类型—字符串

- Unicode编码是一种用于表示文本字符的标准编码系统。Unicode编码支持包括拉丁字母、数字、符号、各种语言的字母、特殊字符等在内的超过110万个广泛字符集。
- Unicode编码的一个优点是可以使用一个统一的编码来表示所有的字符和符号，因此可以避免因为不同编码方案之间的兼容性问题而导致的乱码问题。
- 常用的表示Unicode编码的方式是使用UTF-8编码。UTF-8是一种变长编码，可以表示Unicode字符集中的任何字符，并且兼容ASCII编码。

数据类型—字符串

字符串的“加法”和“乘法”

在Python中，使用加号（+）可以将两个字符串拼接起来，并返回一个新的拼接好的字符串。

使用乘号（*）乘以一个整数n，可以将某个字符串复制n次，并返回一个新的复制n次的字符串。

数据类型—字符串

查询函数

函数	描述
find()	检测字符串是否包含指定字符，如果是则返回开始的索引值，否则返回-1
index()	检测字符串是否包含指定字符，如果是则返回开始的索引值，否则报错
rfind()	从右向左，检测字符串是否包含指定字符，如果是则返回开始的索引值，否则返回-1
rindex()	从右向左，检测字符串是否包含指定字符，如果是则返回开始的索引值，否则报错

数据类型—字符串

转换函数

函数	描述
lower()	将字符串转换为小写
upper()	将字符串转换为大写
title()	将字符串中每个单词的首字母大写

数据类型—字符串

判断函数

函数	描述
startswith()	如果字符串以obj开头，则返回True，否则返回False
endswith()	如果字符串以obj结尾，则返回True，否则返回False
isspace()	如果字符串只包含空格则返回True，否则返回False
isalnum()	如果字符串都是字母或数字则返回True，否则返回False
isdigit()	如果字符串都是数字则返回True，否则返回False
isalpha()	如果字符串都是字母则返回True，否则返回False

数据类型—字符串

分割函数

函数	描述
partition()	将字符串根据参数分割为三部分
rpartition()	从右向左，将字符串根据参数分割为三部分
split()	将字符串根据参数进行分割，且可以指定分割的次数
splitlines()	按照\n分割，返回一个列表

数据类型—字符串

其他函数

函数	描述
count()	统计子字符串出现的次数
join()	将序列中的元素连接成一个新的字符串
replace()	替换指定的字符串，并且可以指定替换的次数
capitalize()	将字符串的首字母大写
len()	返回字符串的长度值

03 数据类型-列表

数据类型—列表

列表：是一种有序的元素集合，用于存储一组有序的数据，可以包含任意数量的元素，并且每个元素可以是不同的数据类型。与字符串不同的是，列表里的元素是可以修改的。

列表使用方括号[]来表示，其定义方式为：

列表名称 = [元素1，元素2，...，元素n]

数据类型—列表

列表的访问：下标访问和切片访问。

下标访问：与字符串的索引一样，列表索引从左到右从0开始，从右到左从-1开始。注意访问时不能超出下标范围，其访问方式为：

列表名[下标]

数据类型—列表

切片访问：与下标访问一样的是都是使用下标来访问，不同的是切片访问可以一次访问多个元素，其访问方式为：

列表名字[初始位置：终止位置：步长]

访问时，包括初始位置不包括终止位置，且步长默认为1。如果没有给出初始位置，默认初始位置为开始位置；如果没有给出终止位置，默认终止位置为列表结束位置，此时访问时包括终止位置。

数据类型—列表

列表的“加法”和“乘法”：

在Python中，使用加号（+）可以将两个列表拼接起来，并返回一个新的拼接好的列表。

使用乘号（*）乘以一个整数n，可以将某个列表复制n次，并返回一个新的拼接好的列表。

数据类型—列表

列表元素的增加

函数	描述
append()	向列表的尾部添加元素
insert()	向列表的指定位置添加元素
extend()	将另一个列表的所有元素添加到本列表的后面

数据类型—列表

列表元素的删除

函数	描述
remove()	删除指定的元素
pop()	用于移除列表的一个元素，可以指定下标，并将该元素返回。
clear()	删除列表中的所有元素
del	指定下标时删除对应的元素，不指定时删除整个列表对象。

数据类型—列表

列表元素的修改：

使用下标或切片的方式对列表中的单个元素或多个元素进行修改。

列表名[索引] = 新的元素内容

list_name[0] = ‘lisi’
list_name[0:2] = [‘zhangsan’, ‘lisi’]

数据类型—列表

列表元素的查找：

所谓的查找，就是查看某元素是否存在于该列表里，这里有三个方法：

- count()：返回列表中某个元素的数量
- 使用in关键字查找，如果存在就返回True，否则返回False
- 使用not in关键字查找，如果不存在就返回True，否则返回False

数据类型—列表

列表的其他常用操作：

- len()：获取列表中元素的个数
- reverse()：反转列表中的元素
- sort()：对列表元素进行排序
- copy()：对列表的拷贝

数据类型—列表

列表的嵌套：在Python中，是允许列表进行嵌套的，也就是说，一个列表里的所有的元素都是列表，并且其访问方式与普通列表的访问有一些差异。在嵌套列表中，需要指定每一层的索引才能访问内层列表的元素，嵌套了多少层的列表，就需要指定多少层的索引。

数据类型—列表

浅拷贝与深拷贝：

- 浅拷贝创建了一个新的对象，该对象与原始对象的内容相同，但是内部的可变元素是原始对象中可变元素的引用，而不是新的独立对象。
- 深拷贝创建了一个新的对象，该对象及其内部的所有元素都是原始对象中元素的副本，而不是引用。换句话说，深拷贝创建了一个完全独立的新对象，其中的所有元素都是原始对象中元素的副本。

数据类型—列表

总的来说，浅拷贝在实际编程中更常见，因为它在大多数情况下能够满足需求，同时也更高效。深拷贝则通常在需要完全独立副本的情况下使用，但由于性能和内存消耗的原因，使用相对较少。

数据类型—列表

列表推导式：是一种简洁的方式来创建或修改列表的方法，它允许你使用单行的代码从一个已有的可迭代对象（如列表、元组、集合、字典等）中快速构建新的列表，其语法如下：

```
new_list = [expression for item in iterable if condition]
```

- expression 是对 item 的操作或表达式，用于生成新的列表元素。
- item 是可迭代对象中的元素，例如列表中的元素。
- iterable 是用于迭代的可迭代对象，例如列表、元组、集合等。
- condition 是一个可选的条件，用于过滤元素。

其中，if condition如果没有必要可以不使用。

数据类型—列表

列表推导式也可以进行嵌套，其语法如下：

```
[expression for item1 in iterable
                for item2 in iterable
                        ...
                    for itemN in iterable
]
```

可以达到多层for循环的作用，其中外层循环在前面，内层循环在里面。

数据类型—列表

加上条件后，嵌套的列表表达式的语法如下：

```
[expression for item1 in iterable1 if condition1
                for item2 in iterable2 if condition2
                        ...
                 for itemN in iterableN if conditionN
]
```

数据类型—列表

总的来说，虽然列表推导式能够以一行代码完成多行代码的任务，且执行效率比for循环更高，但是由于语法的晦涩难懂，可能会导致代码的后期阅读与维护有难度，包括后面的元组、字典、集合等等，虽然都有推导式，但是要根据实际情况来决定是否使用。

04 数据类型-元组

数据类型—元组

元组：与列表非常的相似，不同之处在于元组的元素不可修改。

元组使用圆括号()来表示，其定义方式为：

元组名字 = (元素1，元素2，...，元素n)

数据类型—元组

元组元素的访问：下标访问与切片访问

下标访问：与字符串、列表的方式相同，根据索引进行访问，索引从左到右从0开始，从右到左从-1开始，访问时不能超过下标范围，访问方式为：

元组名称[下标]

数据类型—元组

切片访问：与下标访问一样的是都是使用下标来访问，不同的是切片访问可以一次访问多个元素，其访问方式为：

元组名字[初始位置：终止位置：步长]

访问时，包括初始位置不包括终止位置，且步长默认为1。如果没有给出初始位置，默认初始位置为开始位置；如果没有给出终止位置，默认终止位置为元组结束位置，此时访问时包括终止位置。

数据类型—元组

元组的“加法”和“乘法”：

在Python中，使用加号（+）可以将两个元组拼接起来，并返回一个新的拼接好的元组。

使用乘号（*）乘以一个整数n，可以将某个元组复制n次，并返回一个新的拼接好的元组。

数据类型—元组

元组的常用操作：

- len()：获取元组中元素的个数
- max()：返回元组中元素最大值
- min()：返回元组中元素最小值
- 使用in或not in查找某元素是否存在于元组中
- del：删除元组

数据类型—元组

元组的推导式语法如下：

```
（expression for item1 in iterable1 if condition1
                for item2 in iterable2 if condition2
                    ...
                for itemN in iterableN if conditionN
)
```

与列表不同的地方就是列表使用[]，元组使用()，且生成的对象是一个生成器对象，需要进行强转或通过for循环进行查看元素。

序列

字符串、列表和元组的一些共同点：

- 都可以通过索引（下标）获取每一个元素。
- 第一个元素的索引值从左到右都是以0开始，从右到左以-1开始。
- 都可以通过切片的方法获取一个范围。

它们有一个共同的名字，叫做序列。

序列

序列（Sequence）是一个抽象概念，它指的是可以按照索引位置访问元素的有序数据结构。标准序列类型包括列表（list）、元组（tuple）和字符串（str），其中列表叫做可变序列，元组和字符串叫做不可变序列。

序列的操作

min()和max()函数：用于统计序列中的最小值和最大值，根据传入的序列和参数的不同有不同的结果：

- 不传入参数，直接使用。
- 传入key参数，该参数接收一个函数，该函数会用于序列中的每个元素以确定如何比较元素。
- 默认值：传入default参数，当输入的是空列表时，会打印default参数的内容。

序列的操作

len()函数：用来计算序列的长度或者元素的个数，对于32位系统最大为$2^{31}-1$，对于64位系统最大位$2^{63}-1$。

sum()函数：求序列元素的和，可通过控制start参数来决定求和的起始值，相当于在原有序列的和的基础上又加上了一个元素。

序列的操作

sorted()：用于对序列进行排序，与列表的sort函数不同的地方在于，该函数会返回一个全新的列表，原有的序列不会改变。

reversed()：用于对序列进行翻转，与列表的reverse函数不同的地方在于，该函数会返回一个迭代器，需通过强转或for循环来观看元素。

序列的操作

all()：用于判断序列中的所有元素是否都为真，返回的是布尔值。

any()：用于判断序列中的某个元素是否为真，返回的是布尔值。

序列的操作

enumerate()：用于将一个可遍历的数据对象（如列表、元组或字符串）中的下标与元素组合起来，该函数返回的是枚举对象，是一个包含元组的序列，需要通过强转或for循环进行查看。

zip()：用于将多个可迭代对象（例如列表、元组等）中对应位置的元素打包成一个元组，然后返回由这些元组组成的迭代器，在需要同时处理多个序列的情况下非常有用，同样的也需要通过强转或for循环进行查看元素。

序列的操作

map()：用于对可迭代对象中的每个元素应用一个指定的函数，然后返回一个包含所有函数调用结果的迭代器，也是需要使用强转或for循环进行查看元素。

filter()：会根据提供的函数对指定的可迭代对象的每个元素进行运算，并将运算结果为真的元素以迭代器的形式返回，需要使用强转或for循环进行查看元素。

05 数据类型-集合

数据类型—集合

集合：集合是一个无序的不重复元素序列，分为可变集合与不可变集合。

可变集合的元素在定义好之后是不可修改的，但集合本身是可以增加、删除元素的，这意味着集合的元素只能是数字、字符串及元组，并且每个元素在集合中只会出现一次。

集合使用花括号{}表示，集合的创建方法为：

集合名称 = {元素1，元素2，...，元素n}

注意：集合在python3.7及以后的版本变成有序的了，但这个有序仅仅只是因为底层的存储方式改变而造成的“偶然”的有序性，并不能作为我们的依赖。

数据类型—集合

可变集合元素的添加：

- add()：一次添加单个元素，比如一个字符串、数值型数据。
- update()：一次添加多个元素，比如一个字符串、列表、元组等。

数据类型—集合

可变集合元素的删除：

- remove()：删除指定的元素，如果不存在，就会报错
- discard()：删除指定的元素，如果不存在，不会报错
- pop()：删除第一个元素，如果集合是空的，会报错

数据类型—集合

可变集合元素的查找：

- in：使用in关键字查找某元素是否存在于集合中。
- not in：使用not in关键字查找某元素是否不在集合中。

数据类型—集合

部分其他的操作：

- len()：计算集合里元素的个数
- set()：生成一个集合
- copy()：返回集合的浅拷贝
- clear()：清空集合
- intersection()：求两个集合的交集
- union()：求两个集合的并集
- issubset()：求两个集合是不是子集关系
- issuperset()：求两个集合是不是父集关系

数据类型—集合

集合的推导式：

{

expression1 for item1 in iterable if condition1

}

数据类型—集合

不可变集合：不仅集合的元素不可变，集合的元素的个数也不可变。

使用frozenset()函数进行创建，且之前的一些增删等修改集合的操作都是用不了的，但可以使用求交集、子集等操作。

06 数据类型-字典

数据类型—字典

字典：字典也是一种无序的数据类型，与其他的数据类型不同的是，字典的数据是以键值对的方式进行存储的，键值对是可以进行修改的。

字典也用花括号{}表示，与集合的区别就是花括号{}里的内容以键值对的形式存在，其定义如下：

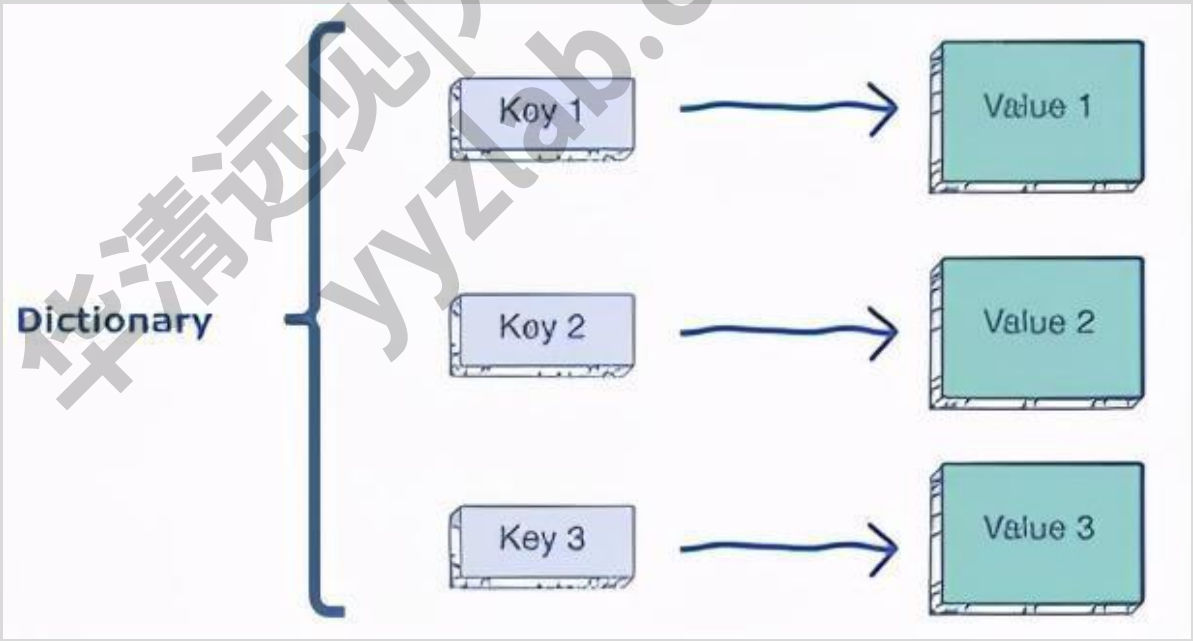
字典名 =｛键1：值1，键2：值2，键3：值3，……｝

注意：字典在python3.7及以后的版本变成有序的了，但为了程序的兼容性，建议在写程序时不要利用这个字典的顺序。

数据类型—字典

键值对的特性：

- 一个字典中，键是唯一的，且键和值是一一对应的，如果定义了相同的键，那么新定义的值会覆盖原值。
- 键必须不可变，可以用数字、字符串或元组，但不能用列表，值可以是任何数据类型。



数据类型—字典

字典的访问：只能通过唯一的键来访问对应的值，其访问方式为：

字典名[键名]

通过这种方法就可以得到该键所对应的值，如果字典中没有这个键就会报错。

数据类型—字典

键值对的添加：

- 可以直接添加。
- 也可以使用update()方法将两个字典合并为一个。

数据类型—字典

键值对的删除

函数	描述
pop()	参数是要删除的键名，并将其对应的值删除。
clear()	删除字典中所有键值对，使字典成为空字典
popitem()	随机弹出一个键值对，在python3.7及以后变为弹出最后一个添加的键值对。
del	用于删除字典

数据类型—字典

修改键值对的值：

- 对于修改单个键值对，可以通过访问键名来修改对应的值。
- 对于修改多个键值对，可以使用update()通过覆盖原有的键值对来修改。

数据类型—字典

查找键值对：

- 直接通过键名查找，类似于字典的访问，如果键不存在就报错
- 使用get()函数来查找，与直接通过键名不同的是，使用get()函数查找时，如果键不存在可以指定要返回的提示语。
- 使用in或not in关键字来查找某键是否存在。
- 使用setdefault()函数查找，如果存在就返回对应的值，不存在就创建一个新的键值对，值通过参数获取。

数据类型—字典

字典的特殊操作

函数	描述
len	返回字典的键值对个数
copy	对字典进行浅拷贝
keys	返回字典中所有的key值
values	返回字典中所有的value值
items	返回字典中所有的键值对

数据类型—字典

字典的推导式：

```
{
key_expression1: value_expression1 for item1 in iterable if condition1
key_expression2: value_expression2 for item2 in iterable if condition2
                                ...
key_expressionN: value_expressionN for itemN in iterable if conditionN
}
```

科技赋能·智引未来

技术领先
品质保障

超多干货
实时更新

海量视频
贴身学习