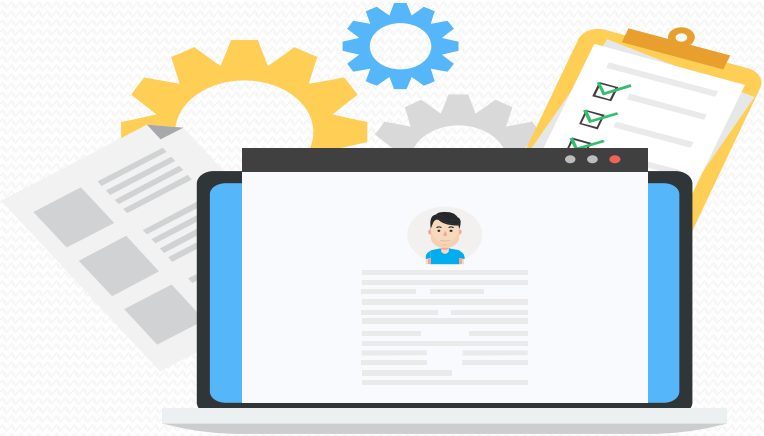
第四章 数据表示

Contents



第2章介绍了基本数据类型（整型、浮点型、复数型、字符串等）和列表类型等。尽管基本数据类型与列表相组合可以表示任意数据类型，但是Python还是将一些特殊形式的数据类型及其相关的方法封装成一些新的类，以方便这些数据类型的操作。这些数据类型包括元组、集合和字典等类型，本章将详细介绍这些新的数据表示类型。

01 元组

4.1 元组

在平面中，表示点的位置（或称矢量）的坐标(x,y)，是一对有序数对，即(x,y)和(y,x)代表不同的点。同样，在n维空间中，点的坐标(x1,x2,…,xn)也是一组有序数列，各个坐标值的位置是固定的。

元组对应着数学上的一组有序数列，例如，包含2个元素x和y的元组表示为(x,y)，这里x和y为元组的元素，且不可改变。可以把元组理解为一个常量数列。

4.1.1 元组定义

元组为由一列有顺序的数据组成的数据类型，元组是只读的。元组间的元素间用“,”号分隔，用圆括号作为元组与其他数据的分界符。空元组和只有一个元素的元组，没有意义，如果非要表示只含一个元素的元组，使用类似于形式“(3,)”表示，即“，”号不能少。

4.1.1 元组定义

```
1    if __name__=='__main__':
2        p1=(3,5)
3        print(f'p1={p1}')
4        p2=(5,7,9)
5        print(f'p2={p2}')
6        print("p2's type is:", type(p2))
7        p3=tuple((3,1,2))
8        print(f'p3={p3}')
```

4.1.1 元组定义

```
9        p4=1,3,5,7
10       print(f'p4={p4}')
11       u=[9,8,5]
12       p5=tuple(u)
13       print(f'p5={p5}')
14       p6=(3,)
15       print(f'p6={p6}')
```

4.1.1 元组定义

```
16      print("p6's type is:", type(p6))
17      p7=()
18      print(f'p7={p7}')
19      print("p7's type is:", type(p7))
20      p8=tuple(range(1,10+1))
21      print(f'p8={p8}')
```

4.1.1 元组定义

第2行“p1=(3,5)”定义元组(3,5)，赋给p1，p1可以表示平面上点的坐标，或平面上的一个矢量。第3行“print(f'p1={p1}')”在屏幕上输出“p1=(3, 5)”，注意，输出元组时，自动添加一对圆括号。

第4行“p2=(5,7,9)”将元组(5,7,9)赋给p2，p2可表示三维空间中的一个点，或三维空间中的一个矢量。第5行“print(f'p2={p2}')”在屏幕上输出“p2=(5, 7, 9)”。第6行“print("p2's type is:", type(p2))”在屏幕上输出p2的类型，得到“p2's type is: <class 'tuple'>”。

4.1.1 元组定义

第2行“p1=(3,5)”定义元组(3,5)，赋给p1，p1可以表示平面上点的坐标，或平面上的一个矢量。第3行“print(f'p1={p1}')”在屏幕上输出“p1=(3, 5)”，注意，输出元组时，自动添加一对圆括号。

第4行“p2=(5,7,9)”将元组(5,7,9)赋给p2，p2可表示三维空间中的一个点，或三维空间中的一个矢量。第5行“print(f'p2={p2}')”在屏幕上输出“p2=(5, 7, 9)”。第6行“print("p2's type is:", type(p2))”在屏幕上输出p2的类型，得到“p2's type is: <class 'tuple'>”。

4.1.1 元组定义

第7行“p3=tuple((3,1,2))”为创建元组的标准方法，这里tuple为元组类型的类名，括号中的参数“(3,1,2)”为创建元组对象时设定的元组中各个元素的值，这里，表示创建一个包含3、1、2三个元素的元组。第8行“print(f'p3={p3}')”在屏幕上输出元组p3，得到“p3=(3, 1, 2)”。

第9行“p4=1,3,5,7”是一种非标准的创建元组的方法，不建议使用。在Python语言中，将这种形式的输入自动设定为元组。第10行“print(f'p4={p4}')”输出元组p4，得到“p4=(1, 3, 5, 7)”。

4.1.1 元组定义

第11行“u=[9,8,5]”得到列表u。第12行“p5=tuple(u)”也是一种标准的创建元组的方法，这里tuple为元组的类名，参数u为列表，将自动转化为元组的形式，赋给p5。

第14行“p6=(3,)”创建仅包含一个元素的元组，这种元组没有意义。注意，这里的“,”号不能省略，因为“(3)”将表示整数3，其中的圆括号只是数据间的分界符，将被忽略。第15行输出元组p6，得到“p6=(3,)”。第16行输出p6的类型，得到“p6's type is: <class 'tuple'>”，可见p6为一个元组。

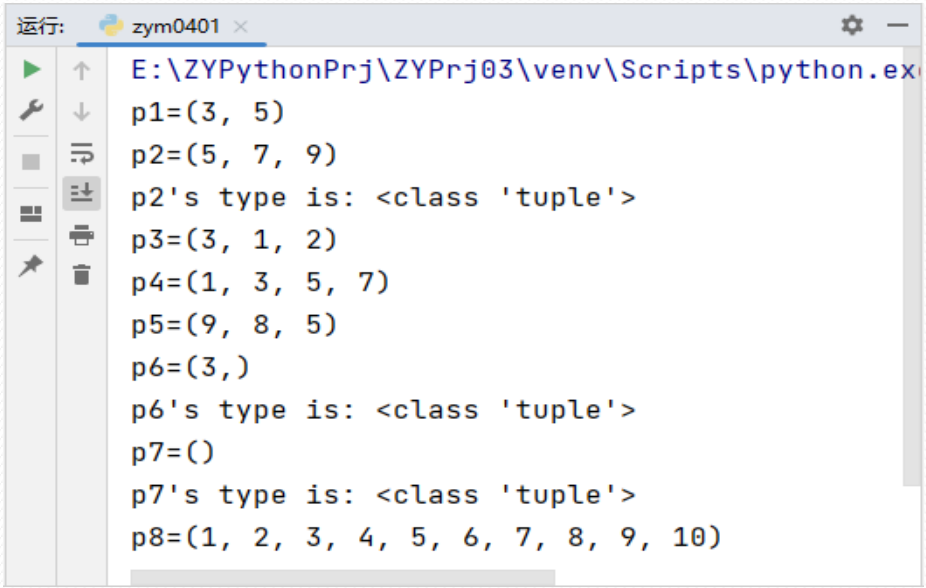
4.1.1 元组定义

第17行“p7=()”生成一个空元组，空元组没有意义。第18行“print(f'p7={p7}')”输出空元组p7。第19行“print("p7's type is:", type(p7))”输出p7的类型，得到“p7's type is: <class 'tuple'>”。

第20行“p8=tuple(range(1,10+1))”由range对象生成元组，这里借助于类tuple的构造方法生成元组p8，。第21行“print(f'p8={p8}')”输出元组p8，得到“p8=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)”。

4.1.1 元组定义

程序执行结果：



模块zym0401执行结果

4.1.1 元组定义

元组是一种复合数据类型，可以保存任意类型的数据，这一点与列表相似。需要特别注意的是，元组中可以包含列表，而且其中的列表元素可以修改，这是因为元组中嵌入的列表仅是占位符，即元组使用了类似于“指针” 或者“标签”的形式“引用”其中嵌入的列表，因此，对元组中列表的操作仍然是对列表的操作，而不是对元组的元素的操作，所以，元组中嵌入的列表中的元素可以修改。

4.1.1 元组定义

- 包含不同数据类型的元组：

```
1    if __name__=='__main__':
2        a=(1,(2,3),[4,5],'Hello World',8,9,3.14,2.71828,1.414)
3        print('a=',a)
4        print("a's type is:",type(a))
5        a[2][1]=a[2][1]+3
6        print('a=', a)
```

4.1.1 元组定义

第2行“a=(1,(2,3),[4,5],'Hello World',8,9,3.14,2.71828,1.414)”定义元组a，其中包含了一个元组(2,3)、一个列表[4,5]、一个字符串“Hello World”、三个整数和三个浮点数。

第3行“print('a=',a)”输出元组a。

第4行“print("a's type is:",type(a))”输出元组a的类型。

第5行“a[2][1]=a[2][1]+3”修改元组中列表“[4,5]”的第1个元）。第6行“print('a=', a)”输出元组a。

4.1.1 元组定义

程序执行结果如下：

模块zym0402

4.1.1 元组定义

上程序段中关于只读元组中的列表可修改的例子，进一步说明Python语言中所有的数据名称均为该数据的“标签”，Python语言中没有“变量”的概念，数据的“标签”指向数据在内存中的首地址。元组是只读的，所以元组中的每个“标签”不能修改，其中的“列表”的标签也不能修改，但是，通过列表的“标签”访问的列表的元素是可能修改的。

4.1.1 元组定义

事实上，元组的作用在于表达类似于空间中点的坐标（或矢量）之类的有序数据，不宜用于其他用途。适合使用列表类型表示的数据，不应使用元组这种类型表示，因此，元组中嵌套列表不是一种好的数据表示。

4.1.2 元组元素访问方法

元组中的元素访问方法与列表中的元素访问方法相同。元组中元素的索引方式有两种：一种为从左向右，索引号从0开始，从左向右递增1；另一种从右向左，最后边的元素（或称元组最后的元素）的索引号为-1，从右向左递减1。可以使用“：”号一次性访问元组的多个元素，有时也称元组的“切片”访问。

4.1.2 元组元素访问方法

```
1    if __name__=='__main__':
2        a=(19,35,7,12,15,29,24,20,30,16,10,21)
3        print('a = ',end='')
4        for e in a:
5            print(e,end='  ')
6        print('\na = ',end='')
7        for i in range(len(a)):
8            print(a[i],end='  ')
```

4.1.2 元组元素访问方法

```
9       print()
10      print(f'a = {a}')
11      print(f'First:{a[0]}, Last:{a[-1]}.')
12      print(a[2:2])
13      print(f'a[1]~a[1]: {a[1:2]}')
14      print(f'a[3]~a[8]: {a[3:8+1]}')
15      print(f'a[3],a[5],a[7]: {a[3:8+1:2]}')
```

4.1.2 元组元素访问方法

```
16      print(f'a[-1]~a[-4]: {a[-1:-4-1:-1]}')
```

第2行“a=(19,35,7,12,15,29,24,20,30,16,10,21)”定义元组a。第3行“print('a = ',end='')”输出“a =”且不换行。

第4~5行为一个for结构，第4行“for e in a:”遍历元组a中的元素，对于每一个元素e，第5行“print(e,end=' ')”输出e的值，后加一个空格。第4~5行说明元素为可迭代型的对象，可以使用for结构逐个访问元组中的元素。

4.1.2 元组元素访问方法

第6行先输出换行符，再输出“a =”。

第7~8行为一个for结构，第7行“for i in range(len(a)):”表示i从0按步长1递增到len(a)-1，此时的每个i可作为元组中元素的索引号；对每个i，执行第8行“print(a[i],end=' ')”输出元组a中的各个元素。可见，元组中元素的访问可以借助于其索引号访问，例如，a[0]为元组a的第0个元素。

第9行“print()”输出一个空白行。

第10行输出元组a，得到“a = (19, 35, 7, 12, 15, 29, 24, 20, 30, 16, 10, 21)”。

4.1.2 元组元素访问方法

第11行“print(f'First:{a[0]}, Last:{a[-1]}.')”输出元组a的首元素和尾元素。

第12行“print(a[2:2])”输出一个空元组。这是因为“2:2”表示索引号从2至1，故将得到一个空元组。因此，使用了“:”访问元组将返回一个元组。

第13行“print(f'a[1]~a[1]: {a[1:2]}')”输出一个元组，只包含元素a[1]。

第14行“print(f'a[3]~a[8]: {a[3:8+1]}')”输出一个元组，包含元素a[3]至a[8]。

4.1.2 元组元素访问方法

第15行“print(f'a[3],a[5],a[7]: {a[3:8+1:2]}')”输出一个元组，包括元素a[3]、a[5]和a[7]。这里的“3:8+1:2”表示索引号从3至8，步长为2。

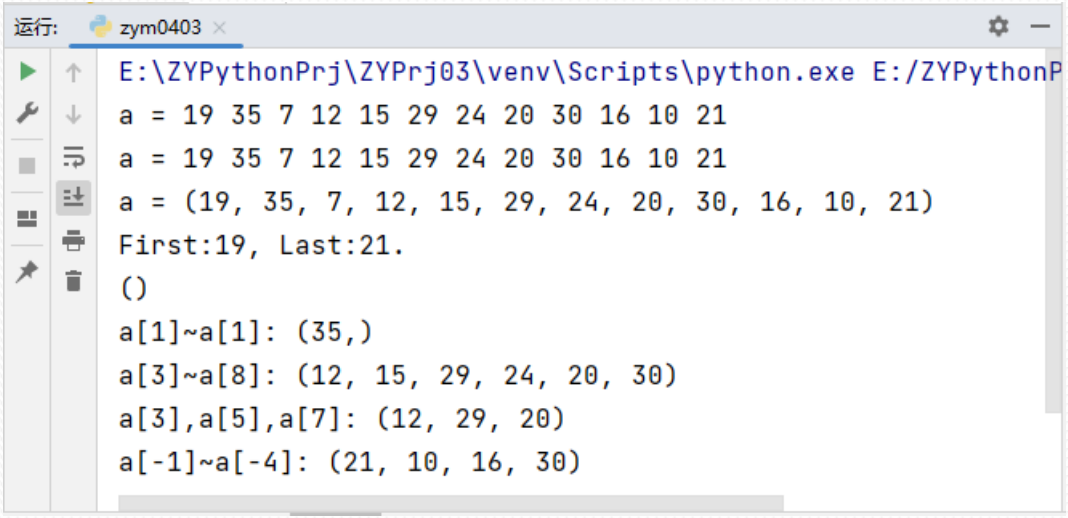
第16行“print(f'a[-1]~a[-4]: {a[-1:-4-1:-1]}')”输出一个元组，包含元素a[-1]至a[-4]。这里的“-1:-4-1:-1”（即“-1:-5:-1”）表示索引号从-1至-4，步长为-1。

4.1.2 元组元素访问方法

使用“:”访问元组的部分内容将返回一个新的元组。“m:n+1:k”表示索引号从m递增到n，步长为k。如果k为1，可以省略。如果m省略，表示索引号从0开始；如果n+1省略表示索引到元组的最后一个元素。因此，a[:]和a的含义相同，均指整个元组a；而a[::2]表示得到一个新的元组，只包含原元组中偶数索引号上的元素。

4.1.2 元组元素访问方法

程序执行结果：



模块zym0403执行结果

4.1.3 元组元素访问方法

元组的内置方法，或称元组类的内置方法，是指元组类内部定义的公有方法。Python语言的内置函数，可以称为全局函数。下面重点介绍表中所示的Python全局函数在元组上的用法，这里设元组a=(3,5,7,9) 。

4.1.3 元组元素访问方法

序号	函数名	含义	示例
1	len	返回元组中的元素个数	len(a)返回 4
2	min	返回元组中最小的元素	min(a)返回 3
3	max	返回元组中最大的元素	max(a)返回 9
4	del	删除整个元组	del(a)删除元组 a
5	tuple	将其他数据类型转化为元组	tuple([3,5,7,9])返回元组 a

4.1.3 元组元素访问方法

```
1    if __name__=='__main__':
2        a=(3,5,7,9)
3        s=len(a)
4        print(f'Length of a: {s}.')
5        print(f'Min and Max of a: {min(a)},{max(a)}')
6        del(a)
7        a=tuple([3,5,7,9])
8        print(a)
```

4.1.3 元组元素访问方法

第2行“a=(3,5,7,9)”定义元组a。第3行“s=len(a)”获得元组的长度4。第4行“print(f'Length of a: {s}.')”输出“Length of a: 4.”。

第5行“print(f'Min and Max of a: {min(a)},{max(a)}')”输出元组a的最小值和最大值，得到“Min and Max of a: 3,9”。

第6行“del(a)”删除元组a。请注意，Python有“垃圾”回收机制，不使用的内存“空间”将被自动回收。

4.1.3 元组元素访问方法

第7行再次对“标签”a赋值，原来的“a”将被自动从内存中清除，因此，一般无需使用del函数。

第7行“a=tuple([3,5,7,9])”得到一个新的元组a。注意，这里的a和第2行的a不是同一个a，尽管其内容相同。

第8行“print(a)”输出元组a，得到“(3, 5, 7, 9)”。

4.1.4 元组应用实例

元组可以视为其元素是常量的特殊“列表”，元组和列表均能“多变量”赋值，所谓的“多变量”赋值例如：

[x,y,z]=[1,2,3] #借助于列表实现x=1、y=2和z=3

(x,y,z)=(1,2,3) 或 a=(x,y,z)=(1,2,3) #借助于元组实现x=1、y=2和z=3

上式可以简写为：

x,y,z=1,2,3 #借助于元组实现x=1、y=2和z=3

4.1.4 元组应用实例

上述的“多变量”赋值，将把x、y和z约束为列表或元组的成员，可以直接使用这些变量，但如果给这些“变量”赋值时，实际上创建一个新的同名“变量”。因此，建议创建只读的“标签”时，可以使用上述的“多变量”赋值，其他情况不建议使用。

“多变量”赋值作为“只读”的“标签”，这种情况最常用于是把元组作为函数的返回值，可以返回多个数据。

4.1.4 元组应用实例

- 例：输入两个正整数，计算这两个整数的最大公约数和最小公倍数

```
1    def mygcdlcm(a,b):
2        if a>b:
3            a,b=b,a
4        if b==0:
5            return (b,a)
6        n=a*b
```

4.1.4 元组应用实例

```
7        while a % b>0:
8            r=a % b
9            a=b
10           b=r
11       n=n//b
12       return (b,n)
13   if __name__=='__main__':
14       c=mygcdlcm(8,22)
15       print(c)
```

4.1.4 元组应用实例

第1~12行为函数mygcdlcm，具有两个参数a和b，使用关键字“def”定义函数。

第2~3行为一个if结构，判断如果a大于b，则将a与b互换。

第4~5行为一个if结构，判断如果b为0，则输出元组(b,a)，此时b为最大公约数，a为最小公倍数。

第6行“n=a*b”将a与b的乘积赋给n。

4.1.4 元组应用实例

第7~10行为一个while循环，这是求两个整数的最大公约数的Euclid算法，此算法依据Euclide定理，即a与b的最大公约数也是b与a % b的最大公约数。用gcd表示求最大公约数的函数，则gcd(a,b)=gcd(b, a %b)。一般地，将a与b中的较大者保存在a中，a与b中的较小者保存在b中，迭代公式(a,b)=(b, a %b)直到得到b能整除a，则b为a与b的最大公约数。这里的“(a,b)=(b, a %b)”表示将b赋给a，同时将a % b赋给b，运算前的a与b和运算后的a与b不占有相同的内存空间。

4.1.4 元组应用实例

第11行“n=n//b”得到的n为a与b的最小公倍数。

第12行“return (b,n)”返回元组，包含了a与b的最大公约数和最小公倍数。

第14行“c=mygcdlcm(8,22)”调用函数mygcdlcm得到8与22的最大公约数和最小公倍数，函数返回值以元组的形式，赋给c。

第15行“print(c)”输出c，得到“(2, 88)”。

02 集合

4.2 集合

Python语言中可定义集合这类数据类型。这里的“集合”是指数学集合论中的集合，即由一组无重复的无序元素组成的集合。在Python语言中，用花括号“{}”括起来且用逗号分隔的一组数据表示。集合中可以嵌套元组，但不能嵌套列表和另一个集合，这是因为集合的元素一定是确定的元素。集合中的元素没有顺序，无法通过索引号访问，集合元素的处理需要通过集合的内置方法实现。

4.2.1 集合定义

集合由一组没有重复的无序元素组成，集合的特点在于其每个元素都是唯一的且确定的。集合中的元素可以取基本类型的常量，也可以取为元组，但不能为列表和集合。下面的程序段展示集合的定义方法。

4.2.1 集合定义

```
1    if __name__=='__main__':
2        a={1,2,2,1,'a',(3,4),'Python','天空',complex(3,5),2.71}
3        print('Set a:',a)
4        b=list(a)
5        print('List b:',b)
6        c=set()
```

4.2.1 集合定义

```
7       print('Empty set:',c)
8       d= set([4, 5, 6, 6, 5])
9       print('Set d:',d)
10       e=set('an apple tree')
11       print('Set e:',e)
12       print('Length of e:',len(e))
```

4.2.1 集合定义

第2行“a={1,2,2,1,'a',(3,4),'Python','天空',complex(3,5),2.71}”定义集合a，其中有重复的元素，在保存时，自动将所有重复的元素仅保留一个；第3行“print('Set a:',a)”输出集合a，得到“Set a: {1, 2, 2.71, (3, 4), '天空', 'a', 'Python', (3+5j)}”，其中所有元素都是唯一的。

第4行“b=list(a)”将集合a转化为列表b；第5行“print('List b:',b)”输出列表b。注意，集合a转化为列表b时，列表b将采用当前的a的元素存储顺序。

4.2.1 集合定义

第6行“c=set()”创建空集合c。由于集合和第4.3节的字典都使用花括号作为定界符，因此，一对空的“{}”将创建一个空字典，而非空集合，可能是因为Python设计人员认为字典比集合更重要。第7行“print('Empty set:',c)”输出空集合c。

第8行“d= set([4, 5, 6, 6, 5])”将一个列表转化为集合，赋给d。第9行“print('Set d:',d)”输出集合d。set是集合类的类名，可以将元组或列表转化为集合。