

# 暨南大学本科实验报告专用纸

课程名称：Python 程序设计实验

实验项目名称：序列型数据类型 指导教师：林聪

实验项目编号：实验 04 实验项目类型：验证性实验

实验地点：实 C302

学生姓名：陈旭天 学号：2021100733

学院：智能科学与工程学院 专业：人工智能

实验时间：2022 年 3 月 2 日下午--2022 年 3 月 2 日下午



# 1. 实验目的

通过自拟案例，初始化并测试各类的序列型数据类型。自行查阅相关材料，了解数据类型 Numpy ndarray，完成思考题。

# 2. 实验原理

# 3. 实验结果

1. 初始化并测试: 列表嵌套, 集合, 字典/OrderedDict, 函数 len()

初始化并测试: 列表嵌套, 集合, 字典/OrderedDict, 函数 len()

```
a = [ [1,2,3],[4,5,6]]
for i in a:
    for n in i:
        print(str(i))
```

0.2s

[1, 2, 3]  
[1, 2, 3]  
[1, 2, 3]  
[4, 5, 6]  
[4, 5, 6]  
[4, 5, 6]

```
a = [ [1,2,3],[4,5,6]]
c = len(a)
print(c)
```




0.4s

2

```
b = (1,2,3,4)
type(b)
```

0.6s

tuple

Name	Type	Size	Value
 a	list	2	[[1, 2, 3], [4, 5, 6]]
b	tuple	4	(1, 2, 3, 4)
c	int		4
 dic	dict	2	{'chen': 'xutian', 'qiu': 'tianai'}
 i	list	3	[4, 5, 6]
n	int		6

使用 jupyter 的变量查看器可以很容易的知道以下变量的类型和大小，在实际操作中应该使用 type () 和 len()函数

2. 练习列表的相关方法 : .append(), del, pop(); 字典的相关方法: .keys(), .values(), .items().

```
list1 = [1,2,3]
```

[19] ✓ 0.2s

list1	list	3	[1, 2, 3]
n	int		6

```
list1 = [1,2,3]  
list1.append(4)
```

list1	list	4	[1, 2, 3, 4]
n	int		6

```
list1 = [1,2,3]  
list1.append(4)  
del list1[3]
```

[3] ✓ 0.3s

list1	list	3	[1, 2, 3]
n	int		6

```
list1 = [1,2,3]  
list1.append(4)  
del list1[3]  
list1.pop()
```

✓ 0.4s

list1	list	2	[1, 2]
n	int		6

```
dict1 = {"chen": "3.24", "mao": "3.61", "hu": "3.91"}  
dict1.keys()  
# dict1.values()
```

[27] ✓ 0.3s

```
... dict_keys(['chen', 'mao', 'hu'])
```

```
dict1 = {"chen": "3.24", "mao": "3.61", "hu": "3.91"}
dict1.keys()
dict1.values()
[28] ✓ 0.5s
... dict_values(['3.24', '3.61', '3.91'])
```

```
dict1 = {"chen": "3.24", "mao": "3.61", "hu": "3.91"}
dict1.keys()
dict1.values()
dict1.items()
✓ 0.4s
dict_items([('chen', '3.24'), ('mao', '3.61'), ('hu', '3.91')])
```

字典的一些方法使用

3. 安装并了解 Numpy 库中的 ndarray 数据类型. 通过下面语句初始化一个 ndarray 对象: `arr = numpy.array([[1,2,3],[4,5,6]])`

测试 `arr.ndim`, `arr.shape`, `arr.size`, `arr.dtype` 等属性, 并了解其代表什么意思.

可以运用在实验 01 中学习到的虚拟环境的相关知识, 激活虚拟环境, 在虚拟环境创建的过程中会自带 numpy 包的下载, 如图, 可以直接使用

```
(base) [chenxutian@mu01 ~]$ python
Python 3.9.7 (default, Sep 16 2021, 13:09:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> arr = numpy.array([[1,2,3],[4,5,6]])
>>> arr.ndim
2
>>> arr.shape
(2, 3)
>>> arr.size
6
>>> arr.dtype
dtype('int64')
>>> █
```

4. 随机初始化一个元素为整数的  $N \times N$  二维数组 (方阵), 例如: `arr = numpy.random.randint(low=0, high=10, size=(10,10), dtype='uint8')` 编写一个函数 `compute_det()`, 实现数组 `arr` 所代表的数学矩阵的行列式/determinant 的计算. 函数的实现可通过拉普拉斯定理代数余子式相关算法(其他方法也行). 函数返回该行列式值. 利用 `numpy.linalg.det()` 检验结果是否正确.

```

[5, 0, 5, 4, 5, 4, 9, 2, 7, 7]], dtype=uint8)
>>> arr = numpy.random.randint(low=0, high=10, size=(10,10), dtype='uint8')
>>> arr
array([[0, 1, 9, 9, 4, 5, 8, 9, 0, 3],
       [0, 3, 9, 4, 4, 8, 6, 6, 0, 2],
       [9, 4, 0, 2, 8, 8, 9, 4, 6, 7],
       [4, 4, 3, 8, 1, 2, 8, 7, 4, 5],
       [5, 6, 3, 9, 8, 0, 4, 2, 7, 3],
       [3, 2, 3, 5, 9, 9, 6, 0, 5, 0],
       [6, 2, 9, 6, 4, 2, 0, 6, 1, 9],
       [0, 8, 6, 7, 1, 2, 0, 4, 7, 7],
       [6, 0, 9, 4, 9, 5, 3, 5, 1, 9],
       [6, 5, 2, 9, 3, 9, 7, 3, 2, 7]], dtype=uint8)

```

首先生成随机矩阵，分别用 numpy 方法和自定义函数计算矩阵行列式的值

```

>>> numpy.linalg.det(arr)
122130959.99999958
>>> def get_det(mat):
...     #为了节省空间，直接在输入的行列式上进行了化简，而没有使用copy
...     mat = mat.astype('float')
...     n = mat.shape[0]
...     res = 1
...     #遍历列
...     for col in range(n):
...         row = col
...         res *= mat[row][col]
...         #寻找不是0的位置row
...         while mat[row][col] == 0 and row < n - 1:
...             row += 1
...         #化简mat[row,col]下面的每一元素为0
...         for i in range(row + 1, n):
...             if mat[i][col] == 0:
...                 pass
...             else:
...                 k = - mat[i][col] / mat[row][col]
...                 for j in range(col, n):
...                     mat[i][j] += mat[row][j] * k
...     return res
>>> get_det(arr)
122130960.00000021

```

检验发现，在误差允许范围内，自定义函数可以成功计算矩阵的行列式的值

#### 4. 思考题

1. 什么是 ndarray 类型？

• NumPy 最重要的一个特点是其 N 维数组对象 **ndarray**，它是一系列同类型数据的集合，以 0 下标为开始进行集合中元素的索引。

**ndarray** 对象是用于存放同类型元素的多维数组。

**ndarray** 中的每个元素在内存中都有相同存储大小的区域。

**ndarray** 内部由以下内容组成：

- 一个指向数据（内存或内存映射文件中的一块数据）的指针。
- 数据类型或 **dtype**，描述在数组中的固定大小值的格子。

- 一个表示数组形状 (**shape**) 的元组, 表示各维度大小的元组。
- 一个跨度元组 (**stride**), 其中的整数指的是为了前进到当前维度下一个元素需要"跨过"的字节数。

2. 如果一个 list 存储 1-100 的整数与一个 ndarray 向量中存储 1-100 有哪些区别?

在 list 中存储是以列表形式存储元素, 相对无序

Ndarray 中存储的形式是集合, 集合的每个元素是列表, 从而实现一个数据集合的存储, ndarray 还会提供一个指向该数据地址的指针, 以及表示行、列大小的元组

2. 比较 list 和 ndarray 各有哪些相对优势?

3. 什么是惰性序列(lazy sequence)? 举出两个例子.

Python 的惰性序列多数指 iterator, 具有惰性计算特点的序列称为惰性序列。

Python 的 iterator 是一个惰性序列, 意思是表达式和变量绑定后不会立即进行求值, 而是当你用到其中某些元素的时候才去求某元素对的值。惰性是指, 你不主动去遍历它, 就不会计算其中元素的值。

在 python 中, 可以尝试使用 itertools 模块实现惰性序列

4. 惰性序列有哪些好处?

对于很大数量的迭代, 惰性序列可以在特定的值出现时才进行运算, 可以很大程度的提升计算机的计算效率和节省性能空间减少内存消耗

5. 给定一个迭代器, 变量名 iter\_A,, 执行 list(iter\_A)后, iter\_A 是否能够再用于迭代? 为什么

a	list_iterator		<list_iterator object at 0x0000018B8E7E6160>
b	list	3	[1, 2, 3]

```
list2 = [1,2,3]
a = iter(list2)
b = list(a)
✓ 0.5s
```

```
list2 = [1,2,3]
a = iter(list2)
a = list(a)
✓ 0.4s
```

a

list

3

[1, 2, 3]

可以用来迭代

[..\Python 语言\实验 04.html](#)