

人工智能实验

2022年春季



课程信息

- 课程网站
 - “超算习堂”教学平台: <https://easyhpc.net/course/143>
 - 加入课程邀请码: 2077
 - 请在超算习堂的账号信息中, 完善好真实姓名和学号
- 课程Q群: 827840726



实验课程要求

实验课程内容：

- 由助教讲解实验内容
- 验收前一次的实验内容（包括公式推导、代码解释、现场运行代码产生结果等）
- 会进行考勤

实验课程要求：

- 实验需要一定的数学基础以及编程基础（公式的推导以及代码的实现）
- 编程语言使用Python/C++/Java
 - 若使用Python，不能使用现有算法高级库（除非助教特别说明），否则扣分。
- 禁止抄袭（代码和实验报告都禁止抄袭，若被发现后果严重）



实验课程安排（暂定）

周次	课上	课下	重要时间节点
1	Python程序设计基础 I	实验1-1	
2	Python程序设计基础 II	实验1-2	
3	归结推理	实验2-1	实验1提交
4	知识图谱 I	实验2-2	
5	知识图谱 II	实验2-3	
6	盲目搜索	实验3-1	实验2提交
7	启发式搜索	实验3-2	
8	博弈树搜索	期中项目	实验3提交
9	高级搜索算法		
10	/		
11	项目1验收		



实验课程安排（暂定）

周次	课上	课下	重要时间节点
12	贝叶斯网络	实验4	项目1提交
13	机器学习 I	实验5-1	实验4提交
14	机器学习 II	实验5-2	
15	机器学习 III	期末项目	实验5提交
16	智能规划（实验6）		实验6提交
17	答疑		
18	项目2验收		
19	项目2验收		项目2提交

实验成绩计算方法（暂定）：

总成绩（100%） = 考勤与课堂表现（10%） + 实验（50%） + 期中项目（15%） + 期末项目（25%）



实验报告要求

- 实验报告可使用Word/Markdown/Latex等撰写，以pdf格式提交，可参考课程网站（超算习堂）中的模板与实验报告编写建议，应包含如下内容：
 - (1) 算法原理：用**自己的话**解释一下自己对算法/模型的理解（不可复制PPT和网上文档内容）
 - (2) 伪代码：伪代码或者流程图（注意简洁规范清晰，包含关键步骤）
 - (3) 关键代码展示：可截图或贴文本并对每个模块进行解释，包括代码+**注释**
 - (4) 创新点&优化：如果有的话，**分点**列出自己的创新点（加分项）
 - (5) 实验结果展示：基础算法的结果&(4)中对应分点优化后的算法结果+**分析**
 - (6) 思考题：PPT上写的思考题（如有）一般需要在报告最后写出解答
 - (7) 参考资料：参考的文献、博客、网上资源等需规范引用，否则涉嫌抄袭



实验提交

- 提交到课程网站（超算习堂）中对应的课程作业，并**注意网站上公布的截止日期**
- 提交格式：提交一个命名为“学号_姓名拼音.zip”的压缩包，压缩文件下包含三部分：code文件夹、result文件夹和实验报告pdf文件
 - 实验报告是pdf格式，命名为：学号_姓名拼音.pdf
 - **code**文件夹：存放实验代码，一般有**多个代码文件**的话需要有readme
 - **result**文件夹：存放上述提到的结果文件（不是每次实验都需要交result，**如果没有要求提交结果，则不需要result文件夹**）
 - “学号_姓名拼音”样例：20*****_wangxiaoming
- 如果需要更新提交的版本，则在后面加_v1，_v2。如第一版是“学号_姓名拼音.zip”，第二版是“学号_姓名拼音_v1.zip”，依此类推

Python程序设计基础

麦成源

maichy7@mail2.sysu.edu.cn

2022.02.17



参考资料与建议阅读

- 《Python编程：从入门到实践》
- 《人工智能（第3版）》附录A
- 超算习堂-在线实训
- <https://www.runoob.com/python3/python3-tutorial.html>



目录

- 1 初识Python
- 2 简单数据类型
- 3 控制结构
- 4 复杂数据结构与操作
- 5 函数与类
- 6 文件与异常
- 7 模块与库
- 8 总结

初识Python

Python 3



什么是Python

Life is short, you need Python.
by Bruce Eckel

- 一种（解释型）编程语言
 - 开发效率高：清晰简洁的语法结构，更贴近自然语言；开发生态好…
 - 运行效率慢：语句需实时解释；变量的数据类型是动态的…^[1]
- 优越的AI生态：有很多可以在AI中使用的库
 - 数据分析与计算：numpy、scipy、pandas
 - 机器学习：scikit-learn
 - 深度学习：pytorch、tensorflow、keras
 - 特定应用领域（如文本挖掘）：gensim
 - …

[1] 为什么 Python 这么慢？ <https://zhuanlan.zhihu.com/p/47795989>



安装： Windows

- 1. 访问Python官网： <https://www.python.org/>， 下载安装包；
- 2. 使用安装包进行安装， 并在运行时勾选： Add Python to PATH；
 - 否则， 请配置环境变量
- 3. 安装完成， 打开终端输入如下命令， 验证是否成功安装：
 - `python --version`
- ▲. 若无文本编辑器或IDE， 建议安装。
 - 文本编辑器， 如Geany、 Sublime Text；
 - IDE， 如Pycharm、 VS Code；



Hello World程序与运行

- Hello World程序

```
print("Hello World!")
```

Hello World!

- 注意：一行为一条语句，而不是分号分隔
- 由Python解释器运行
 - 命令行运行：直接运行语句
 - 命令行运行：运行文件
 - 文本编辑器或IDE运行



1-1.py

Hello World程序： 注释

- 井号（#）注释单行
- 三个单/双引号注释多行

```
# My first Python program

'''
a Hello World program
'''

'''
print a Hello World message
'''

print("Hello World!")
```



Hello World程序： 变量

- 用一个变量存储字符串"Hello World!"

```
message = "Hello World!"  
print(message)
```

- Python是动态类型语言， 变量不需要声明类型
- 变量名
 - 变量名只能包括字母、数字和下划线；
 - 变量名不能以数字开头， 不能包含空格；
 - Python关键字和函数名最好不要用作变量名。



1-2.py

Hello World程序： 输出

- 多条消息的输出

```
message_1 = "Hello World!"  
message_2 = 2022  
print(message_1, message_2)  
print(message_1, message_2, sep="AI", end="SYSU")
```

```
Hello World! 2022  
Hello World!AI2022SYSU
```

- print函数

- 输入参数为要打印的对象；
- 可接收一个或多个参数；
- end参数，默认值为"\n"（即print后默认换行）；
- sep参数，默认值为" "（即多个输出内容之间，默认由空格分开）。



1-3.py

Hello World程序： 用户输入

- 用一个变量存储字符串"Hello World!"

```
message_1 = "Hello!"  
message_2 = input("Please enter a message:\n")  
print("Greeting:", message_1, message_2)
```

Greeting: Hello! SYSU

- input函数
 - 输入参数（可选）：提示字符串
 - 返回值：字符串



初识Python：总结

- Python基本概念、安装与配置
- Hello World程序与运行
- 注释
- 变量
- 输出函数print()
- 输入函数input()



练习：初识Python

- 在你的电脑上配置你的Python开发环境，并运行Hello World程序
- 编写一个程序，该程序打印用户输入的内容

简单数据类型



简单/基本数据类型

- 数字
 - 整数
 - 浮点数
 - 布尔值: **T**True / **F**False (注意大写)
 - int(True)返回1, int(False)返回0
- 字符串
- *空值:

```
a = None
```



数字： 整数

- 加 (+)、减 (-)、乘 (*)、除 (/)、整除 (//)、幂 (**)、模 (%)

```
>>> 2 + 3
5
>>> 3 - 2
1
>>> 2 * 3
6
```

```
>>> 3 / 2
1.5
>>> 3 // 2
1
>>> 4 / 2
2.0
>>> 4 // 2
2
```

```
>>> 3 ** 2
9
>>> 3 ** 3
27
>>> 10 ** 6
1000000
>>> 5 % 3
2
```

```
>>> 2 + 3 * 4
14
>>> (2 + 3) * 4
20
```

向下取整



数字：浮点数

- 加 (+)、减 (-)、乘 (*)、除 (/)、整除 (//)、幂 (**)
- 但要注意的是，结果包含的小数位数可能是不确定的：

```
>>> 0.2 + 0.1
0.30000000000000004
>>> 3 * 0.1
0.30000000000000004
```

- 保留k位小数四舍五入：round(x, k)

```
>>> 5 / 3
1.6666666666666667
>>> 5 // 3
1
>>> round(5 / 3)
2
>>> round(5 / 3, 2)
1.67
```




数字：函数

- 绝对值
 - `abs(a)`
- 最大值、最小值
 - `max(a, b)`
 - `min(a, b)`
- `math`模块：用“`import math`”导入
 - `math.floor(a)`
 - `math.ceil(a)`
 - ...

```
>>> a = -1.5
>>> b = 3.25
>>>
>>> abs(a)
1.5
>>> max(a, b)
3.25
>>> min(a, b)
-1.5
>>>
>>> import math
>>> math.floor(a)
-2
>>> math.ceil(a)
-1
```



i += 1

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

注意：Python中没有类似于“++”的运算符！



2-1.py

字符串

- 字符串就是一系列字符。
- 在Python中，用引号括起的都是字符串，其中的引号可以是单引号也可以是双引号；
 - 这种灵活性能让你在字符串中包含引号或撇号，而无需使用转义字符。

```
s1 = "I told my friend, \"Python is my favorite language!\""
s2 = 'I told my friend, "Python is my favorite language!'"
print(s1 == s2)
```

True



2-2.py

字符串： 拼接

- 用加号 (+) 实现两个字符串的拼接

```
first_name = "Xiaoming"  
last_name = "Wang"  
name = first_name + " " + last_name  
print(name)  
print("Hello, " + name + "!")
```

```
Xiaoming Wang  
Hello, Xiaoming Wang!
```

- 用乘号 (*) 实现重复自拼接

```
s = "haha"  
print(s * 5)
```

```
hahahahahahahahaha
```



2-2.py

字符串： 方法

- 大小写

```
name = "xiaoMing wang"  
print(name.title()) # 每个单词的首字母转化为大写  
print(name.lower()) # 所有字母转化为小写  
print(name.upper()) # 所有字母转化为大写
```

```
Xiaoming Wang  
xiaoming wang  
XIAOMING WANG
```

- 删除空白（空格、换行、制表符）

```
name = " \txiaoMing wang\n"  
print(name.strip()) # 删除字符串前后的空白字符  
print(name.rstrip()) # 删除字符串后面的空白字符  
print(name.lstrip()) # 删除字符串前面的空白字符
```

```
xiaoMing wang  
        xiaoMing wang  
xiaoMing wang
```



2-2.py

字符串： 方法

- 分割

```
sentence = "Life is short, you need Python."  
print(sentence.split())  
print(sentence.split(","))
```

```
['Life', 'is', 'short,', 'you', 'need', 'Python.']  
['Life is short', ' you need Python.']
```

- 以输入的符号为界，分割字符串，得到“列表”

- 替换

```
sentence = "Life is short, you need Python."  
print(sentence.replace("h", "XD"))  
print(sentence.replace("short", "long").replace("Python", "C++"))
```

```
Life is sXDort, you need PytXDon.  
Life is long, you need C++.
```



类型转换

- 格式: `datatype()`
 - `int()`、`float()`、`str()`...
- 例:

```
print(5 // 3) # 1  
print(int(5 / 3)) # 1
```

```
import random  
  
num = random.random()  
message = "random number: " + str(num)  
print(message)
```

- ▲通过`import`导入其它模块/库
- ▲如果直接将数值和字符串相加会导致出错!



简单数据类型：总结

- 数字
 - 包括：整数、浮点数
 - 运算：加 (+)、减 (-)、乘 (*)、除 (/)、整除 (//)、幂 (**) ...
- 字符串
 - 拼接 (+)
 - 方法：大小写、删除空白、分割、替换...
- 类型转换
- 模块的导入 (import)
- 布尔值、空值



练习：简单数据类型

1. 编写4个表达式，它们分别使用加法、减法、乘法和除法运算，但结果都是数字8。你应使用print语句输出，例如：

```
print(5 + 3)
```

2. 找一句你喜欢的名人名言，将这个名人的姓名和他的名言打印出来。其中，名人的姓名存储在变量famous_person中，消息存储在变量message中。输出应类似于下面这样（包括引号）：

```
Albert Einstein once said, "A person who never made a mistake never tried anything new."
```

3. 先用lstrip()、rstrip()和strip()分别处理s中存储的机构名，观察输出结果；之后，用变量保存strip()处理后的名称，并分别以小写、大写和首字母大写的方式显示：

```
s = "\n\tSchool of Computer Science and Engineering \n"
```

控制结构



控制结构

- 分支结构: if
- 循环结构: while
- 循环结构: for



控制结构： 分支结构

```
if condition_A:  
    do something  
elif condition_B:  
    do something  
elif condition_C:  
    do something  
else:  
    do something
```

注意：

- 是elif， 而不是else if;
- if和elif后的条件不用括号包裹;
- if、 elif和else最后加冒号;
- 每个分支内部的代码缩进



3-1.py

- if

```
age = 19
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
```

- if-else

```
age = 17
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
else:
    print("Sorry, you are too young to vote.")
    print("Please register to vote when 18!")
```

- if-elif-else

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 5
elif age < 65:
    price = 10
else: # elif age >= 65:
    price = 5
print("Your admission cost is $" + str(price) + ".")
```

- 为了清晰和明确，最后的 else 也可改为 elif age >= 65
- 变量price虽然在缩进块内定义，但走出循环后依然可用



代码块与缩进

- 在C++中，代码块由花括号 ({...}) 包裹；
- 在Python中，代码块由缩进控制，Python根据缩进来判断代码行与前一个代码行的关系；
- 编写Python代码时要小心严谨地进行缩进，避免发生缩进错误：
 - 缩进量要统一（例如统一用4个空格）；
 - 分支/循环结束后的一行，记得删除一次缩进；
 - ...



比较运算符

a=10, b=20

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 True。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。	(a < b) 返回 True。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 True。



逻辑运算符

Python语言支持逻辑运算符， 以下假设变量 a 为 10， b为 20：

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False， x and y 返回 x 的值， 否则返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是 True， 它返回 x 的值， 否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True， 返回 False 。 如果 x 为 False， 它返回 True。	not (a and b) 返回 False



while循环

- while循环不断地运行，直到指定的条件不满足为止。

```
while condition:  
    do something
```

- while循环中的特殊语句：
 - break
 - continue



while循环：一些典型情况

```
while condition:  
    do something
```

```
while True:  
    do something  
    if condition:  
        break  
    do something
```

```
while condition:  
    do something  
    if condition:  
        continue  
    do something
```



3-2.py

while循环： 例子

```
s = 0
i = 1
while i <= 100:
    s += i
    i += 1
print(s)
```

5050

```
s = 0
i = 1
while True:
    s += i
    i += 1
    if i > 100:
        break
print(s)
```

5050

```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

1
3
5
7
9

```
s = 0
i = 1
flag = True
while flag:
    s += i
    i += 1
    flag = True if i <= 100 else False
print(s)
```

5050



3-2.py

for循环： 初识

```
s = 0
for i in range(100):
    s += i + 1
print(s)
```

5050

```
s = 0
for i in range(1, 101):
    s += i
print(s)
```

5050

```
s = 0
for i in range(100, 0, -1):
    s += i
print(s)
```

5050

```
s = 0
for odd in range(1, 101, 2):
    s += odd
print(s)
```

2500

- The **range** type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in **for** loops.
 - range(stop): 从0到stop-1, 步长为1
 - range(start, stop[, step]): 从start到stop-step, 步长为step, step默认值1



控制结构：总结

- 分支
 - if、if-else、if-elif-else、if-elif
- 循环
 - while循环
 - 初识for循环与range
- 条件判断运算符与布尔表达式
 - 比较运算符
 - 逻辑运算符



练习：控制结构

4. 假设在游戏中刚射杀了一个外星人，请创建一个名为alien_color的变量，并将其设置为'green'、'yellow'或'red'。
 - a) 如果外星人是绿色的，打印一条消息，指出玩家获得了5个点。
 - b) 如果外星人是绿色的，打印一条消息，指出玩家获得了10个点。
 - c) 如果外星人是绿色的，打印一条消息，指出玩家获得了15个点。
5. 编程求出从1到100所有偶数的和
6. 编写一个猜数字游戏的程序：
 - a) 用random模块中的randint()函数生成一个在1到100之间的随机整数作为答案；
 - b) 程序循环读取用户输入的猜测数字，并向用户提示猜测偏大/偏小，直到猜中；
 - c) 用户猜中后，输出一共猜测了多少次。

复杂数据结构与操作



复杂数据类型（数据容器）

- 列表 (list)
- 元组 (tuple)
- 字典 (dict)
- 集合 (set)



列表

- 列表由一系列按特定顺序排列的元素构成
- 回忆：字符串的split()方法

```
sentence = "Life is short, you need Python."  
print(sentence.split())
```

```
['Life', 'is', 'short,', 'you', 'need', 'Python.']
```

- 在Python中，用方括号（[]）来表示列表，并用逗号来分隔其中的元素。

```
bicycles = ["trek", "cannondale", "redline", "specialized"]
```

- 注意：一个列表中的元素可以是不同类型的

```
l = ["abc", 123, 4.5, True, None]
```



4-1.py

列表： 访问元素

```
bicycles = ["trek", "cannondale", "redline", "specialized"]

print(bicycles[0])
print(bicycles[0].title())
print(bicycles[1])
print(bicycles[3])
print(bicycles[-1]) # access the last element in the list
print(bicycles[-3])
message = "My first bicycle was a " + bicycles[0].title() + "."
print(message)
```

```
trek
Trek
cannondale
specialized
specialized
cannondale
My first bicycle was a Trek.
```



4-1.py

列表：修改、添加和删除元素

- 列表创建后，元素可在程序运行过程中动态增删。
- 修改

```
motorcycles = ['honda', 'yamaha', 'suzuki']  
print(motorcycles)  
motorcycles[0] = 'ducati'  
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']  
['ducati', 'yamaha', 'suzuki']
```



4-1.py

列表：修改、添加和删除元素

- 添加

```
motorcycles = []  
motorcycles.append('honda')  
motorcycles.append('yamaha')  
motorcycles.append('suzuki')  
print(motorcycles)  
  
motorcycles.insert(0, 'ducati')  
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']  
['ducati', 'honda', 'yamaha', 'suzuki']
```

- append()方法
 - 在列表末尾添加元素
- insert()方法
 - 在列表中插入元素
 - 第一个参数是插入位置
 - 第二个参数是插入的值



列表：修改、添加和删除元素

4-1.py

- 使用del语句删除元素

```
motorcycles = ['honda', 'yamaha', 'suzuki']  
print(motorcycles)  
del motorcycles[1]  
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']  
['honda', 'suzuki']
```

- 根据值删除元素（remove方法）

```
motorcycles = ['honda', 'yamaha', 'suzuki']  
print(motorcycles)  
motorcycles.remove('yamaha')  
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']  
['honda', 'suzuki']
```

列表：修改、添加和删除

- 使用pop()方法弹出（任何位置的）元素

```
['honda', 'yamaha', 'suzuki']  
The last motorcycle I owned was a Suzuki.  
['honda', 'yamaha']  
['honda', 'yamaha', 'suzuki']  
The second motorcycle I owned was a Yamaha.  
['honda', 'suzuki']
```

```
motorcycles = ['honda', 'yamaha', 'suzuki']  
print(motorcycles)  
last_owned = motorcycles.pop()  
print("The last motorcycle I owned was a " + last_owned.title() + ".")  
print(motorcycles)  
  
motorcycles = ['honda', 'yamaha', 'suzuki']  
print(motorcycles)  
second_owned = motorcycles.pop(1)  
print("The second motorcycle I owned was a " + second_owned.title() + ".")  
print(motorcycles)
```



4-1.py

列表： 长度

- len()函数

```
cars = ['bmw', 'audi', 'toyota', 'subaru']  
print(len(cars))
```

4



4-1.py

列表： 翻转

- reverse()方法

```
cars = ['bmw', 'audi', 'toyota', 'subaru']  
print(cars)  
cars.reverse()  
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']  
['subaru', 'toyota', 'audi', 'bmw']
```

- 注意： reverse()方法做的是原地（in place）操作，即直接对cars永久地修改。
 - 可再次调用reverse()恢复原来的排列顺序。



4-1.py

列表： 排序

- 使用方法sort()对列表进行永久性排序

```
cars = ['bmw', 'audi', 'toyota', 'subaru']  
print(cars)  
cars.sort() # ascending  
print(cars)  
cars.sort(reverse=True) # descending  
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']  
['audi', 'bmw', 'subaru', 'toyota']  
['toyota', 'subaru', 'bmw', 'audi']
```

- sort()方法做的是原地（in place）操作，即直接对cars永久地修改。
 - 且无法恢复原来的排列顺序。



4-1.py

列表： 排序

- 使用函数sorted()对列表进行永久性排序

```
cars = ['bmw', 'audi', 'toyota', 'subaru']  
print(cars)  
cars_ascending = sorted(cars)  
cars_descending = sorted(cars,  
reverse=True)  
print(cars_ascending)  
print(cars_descending)  
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']  
['audi', 'bmw', 'subaru', 'toyota']  
['toyota', 'subaru', 'bmw', 'audi']  
['bmw', 'audi', 'toyota', 'subaru']
```

- sorted()函数不会对输入的cars产生副作用（side effect），即不影响其原始排列顺序



列表：切片

4-1.py

- 切片：从列表中“切”出一段子列表。格式为：list[start: stop(: step)]

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print(players[0:3])
print(players[:3])
print(players[:-2])
print(players[2:4])
print(players[2:])
print(players[-3:])
print(players[::-2])
top_players = players[0:3]
```

```
['charles', 'martina', 'michael']
['charles', 'martina', 'michael']
['charles', 'martina', 'michael']
['michael', 'florence']
['michael', 'florence', 'eli']
['michael', 'florence', 'eli']
['charles', 'michael', 'eli']
```

- 子列表中包含下标从start到stop-step，步长为step的所有元素
 - step默认为1可省略，start默认为0可留空，stop默认为列表长度可留空



4-1.py

列表：赋值与复制

- 赋值：这是你预期的结果吗？

```
my_foods = ['pizza', 'falafel', 'carrot cake']
print('my_foods:', my_foods)
your_foods = my_foods
your_foods[-1] = 'apple'
print('yr_foods:', your_foods)
print('my_foods:', my_foods)
```

```
my_foods: ['pizza', 'falafel', 'carrot cake']
yr_foods: ['pizza', 'falafel', 'apple']
my_foods: ['pizza', 'falafel', 'apple']
```

- 分析 ▲ id()函数用于获取对象内存地址。
▲ 身份运算符：x is y, 类似 id(x) == id(y)

```
print(id(your_foods))
print(id(my_foods))
print(id(your_foods) == id(my_foods))
print(your_foods is my_foods)
```

```
1986166284936
1986166284936
True
True
```



身份运算符

身份运算符用于比较两个对象的**存储单元**

运算符	描述	实例
is	is 是判断两个标识符是不是引用自一个对象	x is y , 类似 id(x) == id(y) , 如果引用的是同一个对象则返回 True, 否则返回 False。
is not	is not 是判断两个标识符是不是引用自不同对象	x is not y , 类似 id(a) != id(b) 。如果引用的不是同一个对象则返回结果 True, 否则返回 False。

▲ id()函数用于获取对象内存地址

▲ is 与 == 区别:

is 用于判断两个变量引用对象是否为同一个(同一块内存空间),

== 用于判断引用变量的值是否相等。

```
>>> a = 123456789
>>> b = 123456788 + 1
>>> a == b
True
>>> a is b
False
```

```
>>> a = 2
>>> b = 1 + 1
>>> a == b
True
>>> a is b
True
```



4-1.py

列表：赋值与复制

- 利用切片复制

```
my_foods = ['pizza', 'falafel', 'carrot cake']
print('my_foods:', my_foods)
your_foods = my_foods[:]
your_foods[-1] = 'apple'
print('yr_foods:', your_foods)
print('my_foods:', my_foods)
```

- 分析 ▲ id()函数用于获取对象内存地址。
▲ 身份运算符：x is y, 类似 id(x) == id(y)

```
print(id(your_foods))
print(id(my_foods))
print(id(your_foods) == id(my_foods))
print(your_foods is my_foods)
```

切片会创建一个新的对象，
分配新的内存空间

```
my_foods: ['pizza', 'falafel', 'carrot cake']
yr_foods: ['pizza', 'falafel', 'apple']
my_foods: ['pizza', 'falafel', 'carrot cake']
```

```
1986166285064
1986166284744
False
False
```



浅复制与深复制

4-1.py

```
import copy
a = [1, 2, 3, 4, ['a', 'b']]

b = a # assign
c = a[:] # slice (shallow copy)
d = copy.copy(a) # shallow copy
e = copy.deepcopy(a) # deep copy

a.append(5)
a[4].append('c')
```

- 利用Python标准库的copy库

```
print( 'a = ', a )
print( 'b = ', b )
print( 'c = ', c )
print( 'd = ', d )
print( 'e = ', e )
```

• 结果

```
a = [1, 2, 3, 4, ['a', 'b', 'c'], 5]
b = [1, 2, 3, 4, ['a', 'b', 'c'], 5]
c = [1, 2, 3, 4, ['a', 'b', 'c']]
d = [1, 2, 3, 4, ['a', 'b', 'c']]
e = [1, 2, 3, 4, ['a', 'b']]
```



列表：for循环遍历

4-2.py

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(magician.title() + ", that was a great trick!")  
    print("I can't wait to see your next trick, " + magician.title() + ".\n")  
print("Thank you, everyone. That was a great magic show!")
```

Alice, that was a great trick!
I can't wait to see your next trick, Alice.

David, that was a great trick!
I can't wait to see your next trick, David.

Carolina, that was a great trick!
I can't wait to see your next trick, Carolina.

Thank you, everyone. That was a great magic show!



4-2.py

列表：数值列表

- 使用range的for循环

```
squares = []  
for value in range(1, 11):  
    squares.append(value**2)  
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- 复习：range类型

- range(stop):
 - 从0到stop-1，步长为1
- range(start, stop[, step]):
 - 从0到stop-step，步长为step，step默认值1



列表： 数值列表

4-2.py

- 使用类型转换函数list()获取数值列表

```
>>> range(1, 11, 2)
range(1, 11, 2)
>>>
>>> list(range(1, 11, 2))
[1, 3, 5, 7, 9]
```

- 对数字列表进行简单统计计算

```
>>> digits = list(range(10))
>>> min(digits)
0
>>> max(digits)
9
>>> sum(digits)
45
```



4-2.py

列表： 列表解析

- 使用range的for循环

```
squares = []  
for value in range(1, 11):  
    squares.append(value**2)  
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- 列表解析

```
squares = [value**2 for value in range(1, 11)]  
  
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```



4-2.py

列表：使用if语句处理列表

- 确定列表不是空的

```
l = []  
if l:  
    print("Not empty")  
else:  
    print("Empty")
```

Empty

- 判断元素在列表中

```
magicians = ['alice', 'david', 'carolina']  
if 'alice' in magicians:  
    print("Hi, Alice!")  
print('zachary' in magicians)
```

Hi, Alice!
False

- 类似的有：0、空列表、空字符串、None等

- 成员运算符：in



成员运算符

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中，如果 x 不在 y 序列中返回 True。

序列：字符串，列表、元组等

```
>>> "l" in "Artificial Intelligence"
True
>>> "ll" in "Artificial Intelligence"
False
>>> "tell" in "Artificial Intelligence"
True
>>> "AI" not in "Artificial Intelligence"
True
```



4-2.py

元组

- 不可变的列表
- 圆括号标识

```
dimensions = (200, 50) #原始元组
for dimension in dimensions:
    print(dimension)
dimensions = (150, 50) #整体修改
for dimension in dimensions:
    print(dimension)
dimensions = list(dimensions)
dimensions[0] = 100 #转为列表修改
dimensions = tuple(dimensions)
for dimension in dimensions:
    print(dimension)
dimensions[0] = 200 #试图直接修改
```

```
200
50
150
50
100
50
Traceback (most recent call last):
  File "4-2.py", line 29, in <module>
    dimensions[0] = 200
TypeError: 'tuple' object does not support item assignment
```



集合

- 集合 (set) 是一个无序的不重复元素序列。
- 其中一种常用场景：元素去重

```
>>> a = [1, 4, 2, 1, 2]
>>> list(set(a))
[1, 2, 4]
```

- 扩展阅读: <https://www.runoob.com/python3/python3-set.html>



字典

4-3.py

- 字典是一系列键-值对 (key-value pair) ， 每个键都与一个值相关联。

```
alien_0 = {'color': 'green', 'points': 5}
```

- 访问字典中的值

```
print(alien_0['color'])  
print(alien_0['points'])
```

```
green  
5
```

- 修改字典中的值

```
alien_0['color'] = 'yellow'  
print(alien_0['color'])
```

```
yellow
```




4-3.py

字典

- 添加键-值对

```
alien_0 = {}  
alien_0['color'] = 'green'  
alien_0['points'] = 5  
print(alien_0)
```

```
{'color': 'green', 'points': 5}
```

- 删除键-值对

```
alien_0 = {'color': 'green', 'points': 5}  
print(alien_0)  
del alien_0['points']  
print(alien_0)
```

```
{'color': 'green', 'points': 5}  
{'color': 'green'}
```



字典： 遍历

```
favorite_languages = {  
    'jen': 'python',  
    'sarah': 'c',  
    'edward': 'ruby',  
    'phil': 'python',  
}
```

- 遍历所有的键-值对

```
for name, language in favorite_languages.items():  
    print(name.title() + "'s favorite language is " + language.title() + ".")
```

Jen's favorite language is Python.
Sarah's favorite language is C.
Edward's favorite language is Ruby.
Phil's favorite language is Python.

- 遍历字典中的所有键

```
for name in favorite_languages.keys():  
    print(name.title())
```

Jen
Sarah
Edward
Phil

- 遍历字典中的所有值

```
for language in favorite_languages.values():  
    print(language.title())
```

Python
C
Ruby
Python



字典： 遍历

```
favorite_languages = {  
    'jen': 'python',  
    'sarah': 'c',  
    'edward': 'ruby',  
    'phil': 'python',  
}
```

Edward's favorite language is Ruby.
Jen's favorite language is Python.
Phil's favorite language is Python.
Sarah's favorite language is C.

- 按顺序遍历字典中的所有键

```
for name in sorted(favorite_languages.keys()):  
    language = favorite_languages[name]  
    print(name.title() + "'s favorite language is " + language.title() + ".")
```

- 虽然py3.6遍历顺序与存储/添加顺序相同，但这不被保证。^[1]
- 因此，我们可以利用sorted规定遍历顺序

- 遍历字典中的所有不重复值

```
for language in set(favorite_languages.values()):  
    print(language.title())
```

Ruby
C
Python

[1] <https://exp.newsmth.net/topic/article/e8a1d1ca4cffec5c61b42f4debb665e7>



字典： 嵌套

- 字典列表
 - 列表中的元素，是字典
- 在字典中存储列表
 - 字典中的值，是列表
- 在字典中存储字典
 - 字典中的值，是字典



总结

- 列表 (list)
 - 元素的访问、修改、添加和删除
 - 列表的操作：长度、翻转、排序、切片与复制
 - 列表的遍历 (for)，数字列表，列表解析
 - 使用if语句处理列表
- 元组 (tuple)、集合 (set)
- 字典 (dict)
 - 访问、修改、添加和删除
 - 遍历



练习：复杂数据结构与操作

6. 编程求出1到100的和（你能用一行代码完成吗？）
7. 实现矩阵乘法的程序（使用二维列表存储矩阵）
8. 创建一个名为cities的字典，其中将三个城市名用作键；对于每座城市，都创建一个字典，并在其中包含该城市所属的国家、人口约数以及一个有关该城市的事实。在表示每座城市的字典中，应包含country、population和fact等键。将每座城市的名字以及有关它们的信息都打印出来

实验任务



实验1-1 最短路径算法

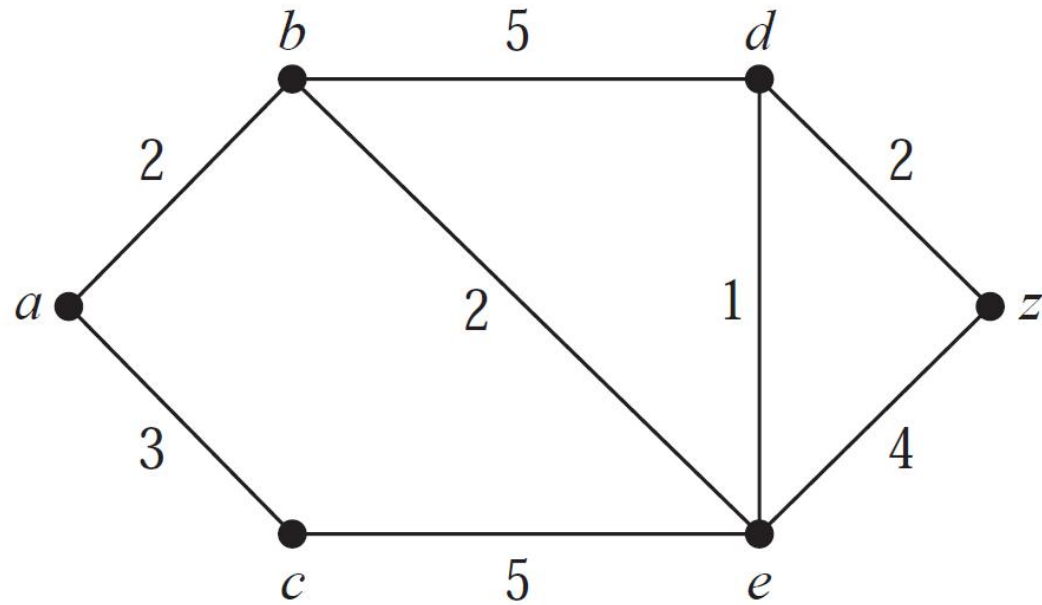
- 给定无向图，及图上两个节点，求其最短路径及长度
- 要求：使用Python实现，至少实现Dijkstra算法
- 输入（统一格式便于验收）
 - 第1行：节点数m 边数n（中间用空格隔开，下同）；
 - 第2行到第n+1行是边的信息，每行是：节点1名称 节点2名称 边权；
 - 第n+2行开始可接受循环输入，每行是：起始节点名称 目标节点名称。
- 输出（格式不限）
 - 最短路径及其长度。



实验1-1 最短路径算法

- 样例

```
6 8  
a b 2  
a c 3  
b d 5  
b e 2  
c e 5  
d e 1  
d z 2  
e z 4  
a z
```



Thanks!

附录



Python代码风格规范

- 自行了解：PEP 8



Python之禅

- 在Python命令行中输入import this并回车，发现Python的隐藏彩蛋