成像算法基础

Python基础

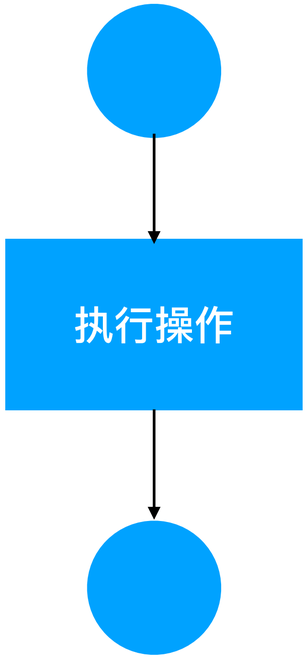
高思成像科技
Goss Imaging Technology

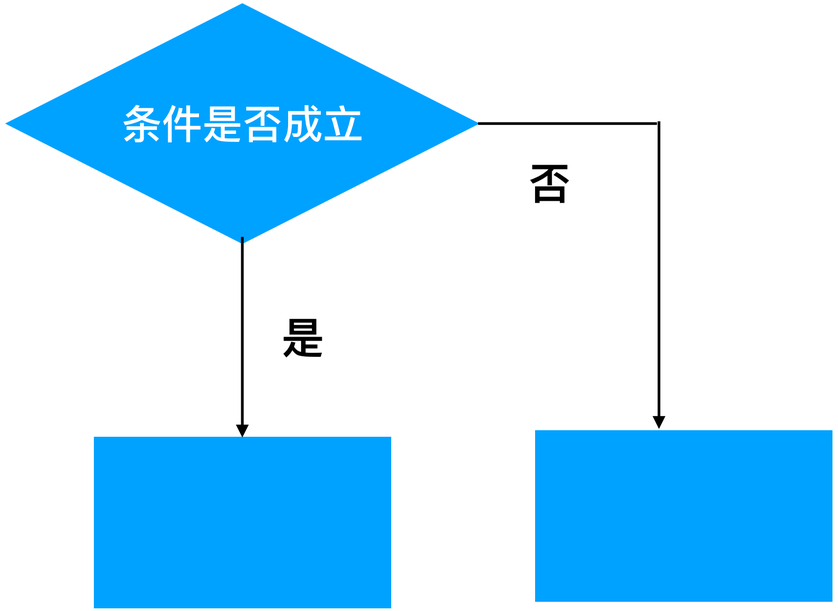
尽快开始你的程序

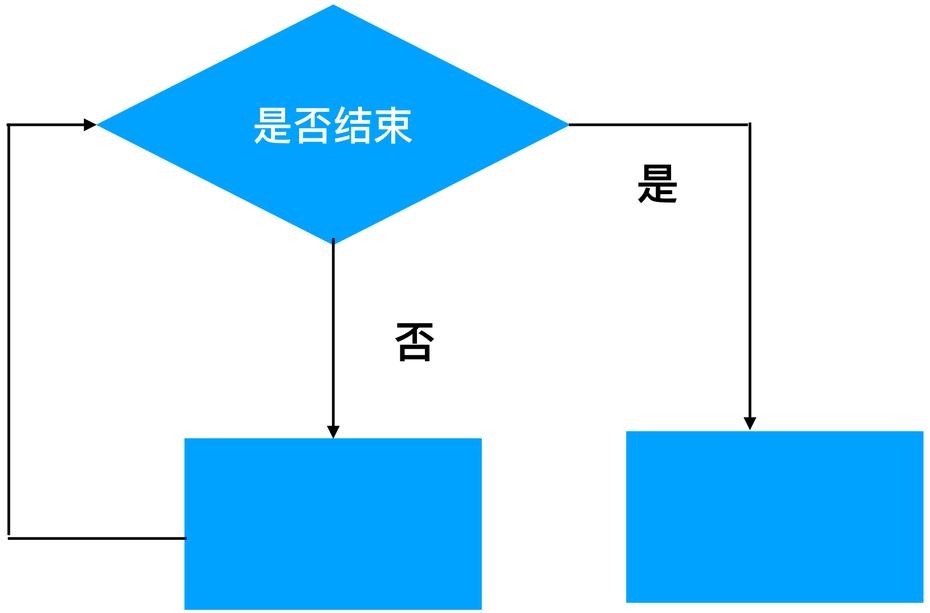
主要内容

- 流程图伪代码
- 基本编程元素
- 脚本函数和包
- Numpy

伪代码到程序







第一个脚本

- 1 建立项目和脚本
- 2 运行
- 3 Debug,查看断点
- 4 步进,查看系统函数的代码
- 5 查看

行和缩进

学习 Python 与其他语言最大的区别就是，Python 的代码块不使用大括号 `{}` 来控制逻辑范围，函数以及其他逻辑判断。python 最具特色的就是用缩进来区分模块和逻辑。缩进的空白数量是可变的，但是所有代码块语句必须包含相同的缩进空白数量，这个必须严格执行

```
if True:
    print ("True")
else:
    print (“False”)
```

如果错误

幸运的时候

```
IndentationError: unindent does not match any outer indentation level
```

不幸的时候
甚至执行逻辑错误

变量

Python 中的变量赋值不需要类型声明。每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。等号（=）用来给变量赋值。
在内存中存储的数据可以有多种类型。
例如，一个人的年龄可以用数字来存储，他的名字可以用字符来存储。Python 定义了一些标准类型，用于存储各种类型的数据。Python有五个标准的数据类型：

- Numbers（数字）
- String（字符串）
- List（列表）
- Tuple（元组）
- Dictionary（字典）

```
counter = 100   # 赋值整型变量
miles = 1000.0  #浮点型
name = "John"  #字符串

print(counter,miles,name)
```

Python 保留字符

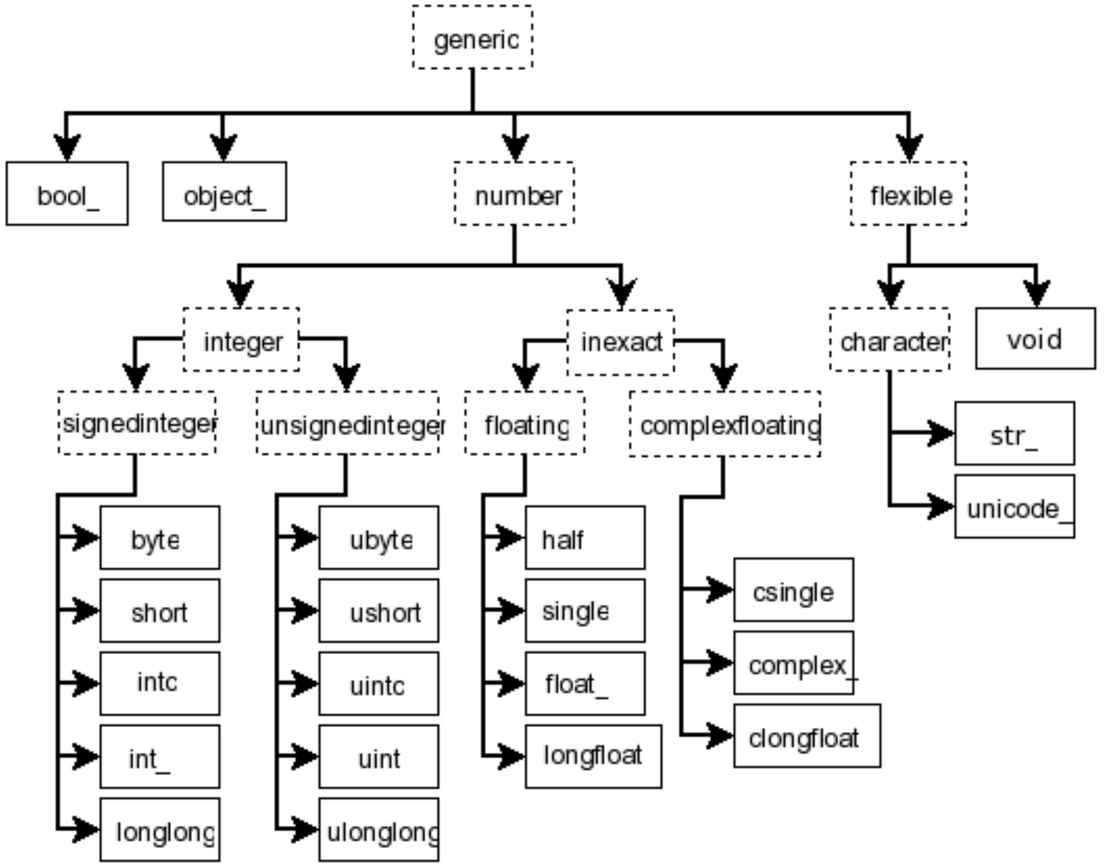
def	if	return	and	exec	not
del	import	try	assert	finally	or
elif	in	while	break	for	pass
else	is	with	class	from	print
except	lambda	yield	continue	global	raise

数字类型

Python支持四种不同的数字类型：

- int（有符号整型）
- long（长整型[也可以代表八进制和十六进制]）
- float（浮点型）
- complex（复数）

在NumPy中，有24种新的基本Python类型来描述不同类型的标量。这些类型描述符主要基于CPython编写的C语言中可用的类型，其他几种类型与Python的类型兼容。



N维数组,列表,元组

Python没有默认数组需要numpy库支持,只有列表(list)和元组(tuple)；

list是列表，list中的元素的数据类型可以不一样。数组中的元素的数据类型必须一样.列表中的元素可以是多种类型

元组一旦创建不可改变，元组不能追加(append)元素，弹出(pop)元素.相当于只读的list

ndarray：全称（n-dimensional array object）是储存单一数据类型的多维数组。

```
tuple = ( 'runoob', 786 , 2.23, 'john', 70.2 )
list = [ 'runoob', 786 , 2.23, 'john', 70.2 ]
#tuple[2] = 1000    # 元组中是非法应用
list[2] = 1000     # 列表中是合法应用
print(tuple)              # 输出完整元组
print(tuple[0])           # 输出元组的第一个元素
print(tuple[1:3])         # 输出第二个至第四个（不包含）的元素
print(tuple[2:])          # 输出从第三个开始至列表末尾的所有元素
print(list[0])            # 输出列表的第一个元素
print(list[1:3])          # 输出第二个至第三个元素
print(list[2:])           # 输出从第三个开始至列表末尾的所有元素
```

高思成像科技
Goss Imaging Technology

二维列表

列表也可以是多维的

```
nums = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [3, 4, 7]]
total = 0
for i in nums:
    for j in i:
        total += j
print(total)
```

列表和元组可以套在一起

```
#元组列表
tuple_list=[(1,2),(3,4)]
tuple_list[0]=(1,2)     #合法
#tuple_list[0][0]=1     #非法
```

Python 字典

字典(dictionary)是除列表以外python之中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。
两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。
字典用"{ }"标识。字典由索引(key)和它对应的值value组成。

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
print(dict['one']) # 输出键为 'one' 的值
print(dict[2])  # 输出键为 2 的值
print(tinydict)  # 输出完整的字典
print(tinydict.keys())  # 输出所有键
print(tinydict.values())  # 输出所有值
```

Python模块

Python 模块(Module)，是一个 Python 文件，以 .py 结尾，包含了 Python 对象定义和Python语句。模块让你能够有逻辑地组织你的 Python 代码段。把相关的代码分配到一个模块里能让你的代码更好用，更易懂。模块能定义函数，类和变量，模块里也能包含可执行的代码。

模块的使用是通过import

复杂的模块也可以由多个文件组成,这个时候需要通过包的形式来管理

```
包名/
    __init__.py
    primitive/
        __init__.py
        line.py
        fill.py
        text.py
    formats/
        __init__.py
        png.py
        jpg.py
```

引用模块

Python 模块(Module)，是一个 Python 文件，以 .py 结尾，包含了 Python 对象定义和Python语句。

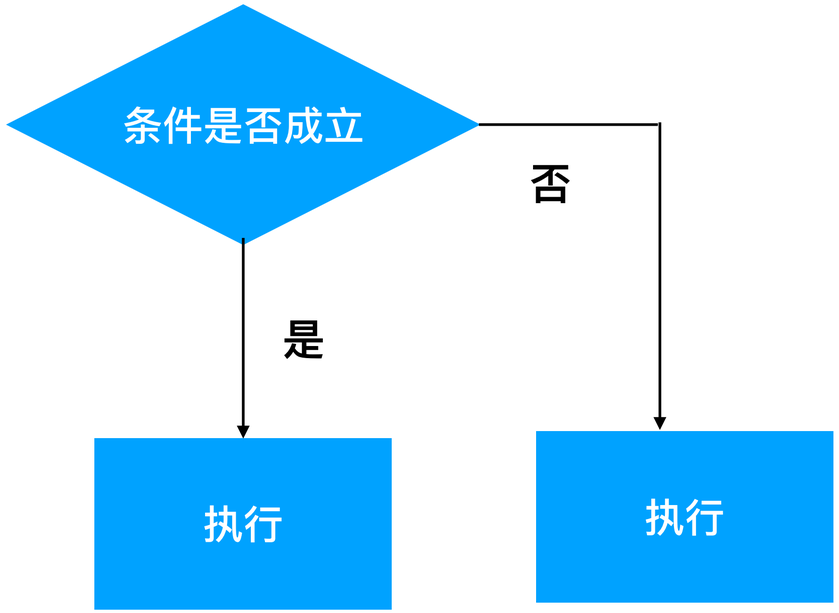
- import module1
- import module1 as name
- from module1 import function
- from module1 import function as name
- from module1 import *

判断语句

```
if 判断条件:
    执行语句……
else:
    执行语句……
```

```
flag = False
name = 'luren'
if name == 'python':          # 判断变量是否为 python
    flag = True               # 条件成立时设置标志为真
    print('welcome boss')     # 并输出欢迎信息
else:
    print(name)               # 条件不成立时输出变量名称
```

```
num = 5
if num == 3:          # 判断num的值
    print('boss')
elif num == 2:
    print('user')
elif num == 1:
    print('worker')
elif num < 0:         # 值小于零时输出
    print('error')
else:
    print('roadman')     # 条件均不成立时输出
```



高思成像科技
Goss Imaging Technology

for循环

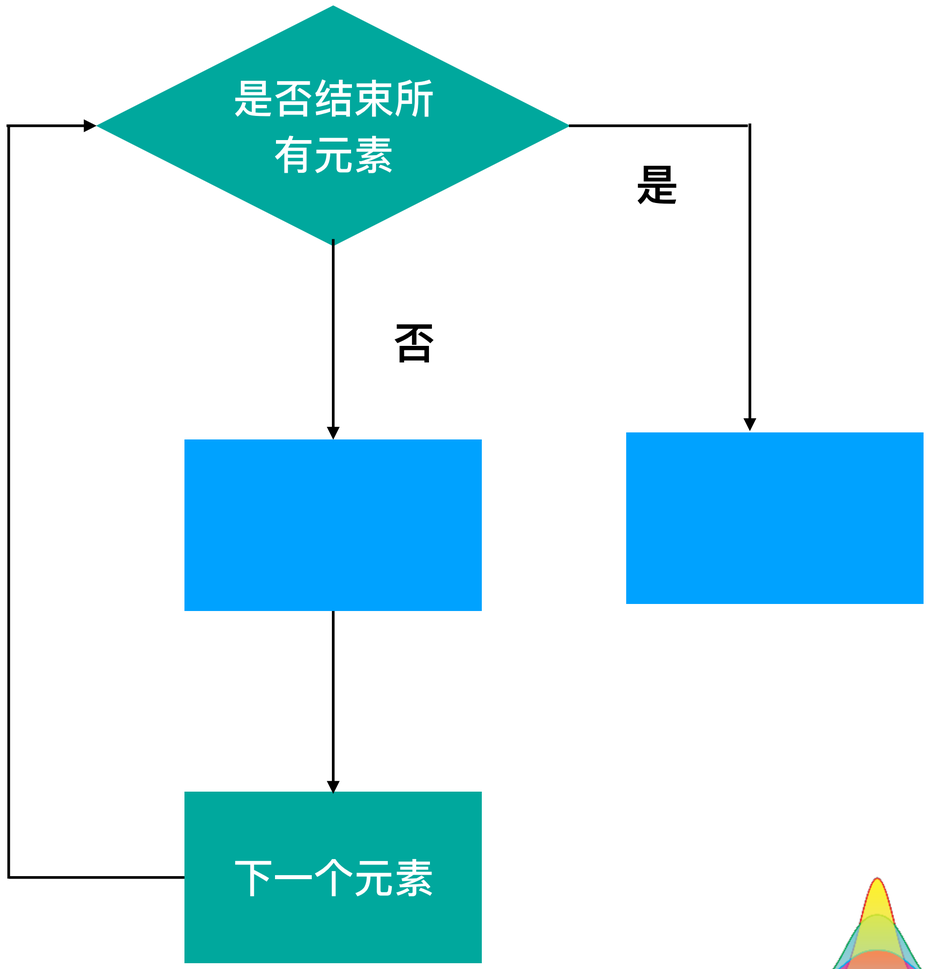
```
def find_target(target, nums):#找出排序数组的索引
    for i in range(len(nums)):
        if nums[i]==target:
            return i


print(find_target(5, [1,3,5,6]))
```

```
n=5

for i, j in zip(range(n-1, 0, -1), range(n//2)):

    print ('i = {0}, j = {1} '.format(i, j))
```



高思成像科技
Goss Imaging Technology

range() 函数用法

`range(start, stop[, step])`

- start: 计数从 start 开始。默认是从 0 开始。例如range（5）等价于range（0， 5）；
- stop: 计数到 stop 结束，但不包括 stop。例如：range（0， 5） 是[0, 1, 2, 3, 4]没有5
- step：步长，默认为1。例如：range（0， 5） 等价于 range(0, 5, 1)

```
>>>range(10)         # 从 0 开始到 10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 11)     # 从 1 开始到 11
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> range(0, 30, 5)  # 步长为 5
[0, 5, 10, 15, 20, 25]
>>> range(0, 10, 3)  # 步长为 3
[0, 3, 6, 9]
>>> range(0, -10, -1) # 负数
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> range(0)
[]
>>> range(1, 0)
[]
```

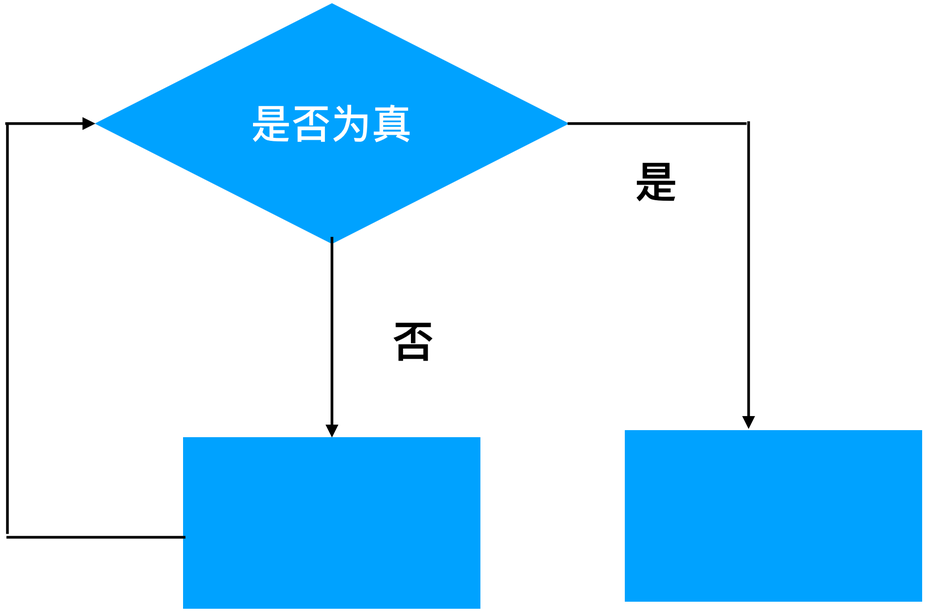
高思成像科技
Goss Imaging Technology

while 循环

```
count = 0
while (count < 9):
   print('The count is:', count)
   count = count + 1

print "Good bye!
```

```
count = 0
while count < 5:
   print(count, " is  less than 5”)
   count = count + 1
else:
   print(count, " is not less than 5”)
```



高思成像科技
Goss Imaging Technology

函数

定义一个函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号**()**。
- 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- **return [表达式]** 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 None。

```
def move(x, y, step, angle=0):
    nx = x + step * math.cos(angle)
    ny = y - step * math.sin(angle)
    return nx, ny
```

函数的返回值和调用

函数体内部的语句在执行时，一旦执行到return，函数就执行完毕，并将结果返回。

如果没有return语句，函数执行完毕后也会返回结果，只是结果为None。return None可以简写为return。

```
import math
def move(x, y, step, angle=0):
    nx = x + step * math.cos(angle)
    ny = y - step * math.sin(angle)
    return nx, ny

x, y = move(100, 100, 60, math.pi / 6)
r = move(100, 100, 60, math.pi / 6)
print(x, y)
print(r)
```

文件默认函数入口

1 通过默认的函数名字
if __name__ == '__main__':

2 脚本和函数混编
缺点容易有对齐问题

```
import math
def move(x, y, step, angle=0):
    nx = x + step * math.cos(angle)
    ny = y - step * math.sin(angle)
    return nx, ny

x, y = move(100, 100, 60, math.pi / 6)
r = move(100, 100, 60, math.pi / 6)
print(x, y)
print(r)
return nx, ny
```

环境设置

PYTHONPATH 变量
作为环境变量，PYTHONPATH 由装在一个列表里的许多目录组成。PYTHONPATH 的语法和 shell 变量 PATH 的一样。
在 Windows 系统，典型的 PYTHONPATH 如下：
set PYTHONPATH=c:\python27\lib;
在 UNIX 系统，典型的 PYTHONPATH 如下：
set PYTHONPATH=/usr/local/lib/python

Pycharm

1）打开File--》Setting—》打开 Console下的Python Console，把选项（Add source roots to PYTHONPAT）点击勾选上
2）右键点击自己的工作空间，找下面的Mark Directory as 选择Source Root，就可以解决上面的问题了

常用标准库

操作系统接口

```
import os
os.getcwd()      # 返回当前的工作目录
os.chdir('/server/accesslogs')   # 修改当前的工作目录
os.system('mkdir today')   # 执行系统命令 mkdir
```

文件通配符

```
import glob
glob.glob('*.py')
```

数学

```
import math
math.cos(math.pi / 4)
math.log(1024, 2)
```

日期和时间

Python的高级部分