Python快速入门

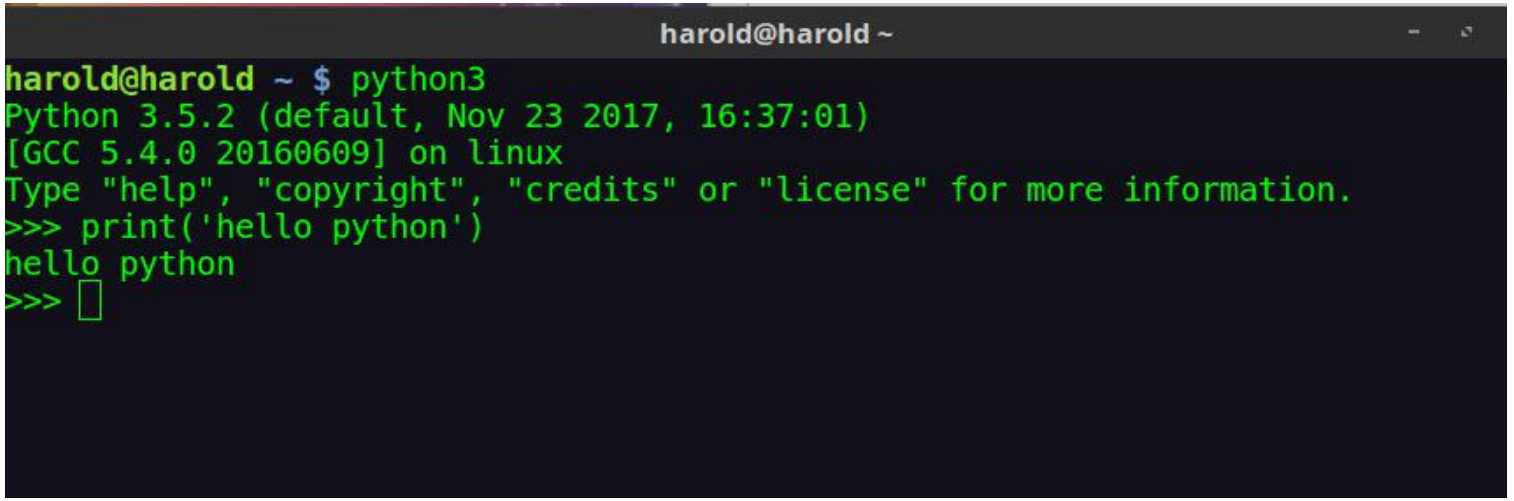
第一阶段：Python基础教程

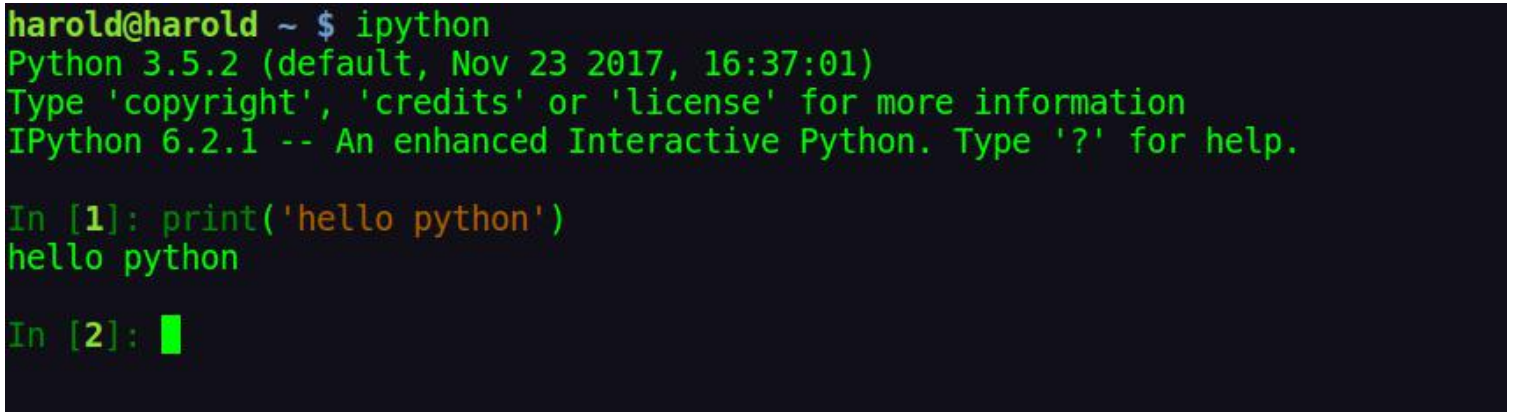
1 第一个Python程序

- 交互式编程
- 脚本式编程

1 第一个Python程序

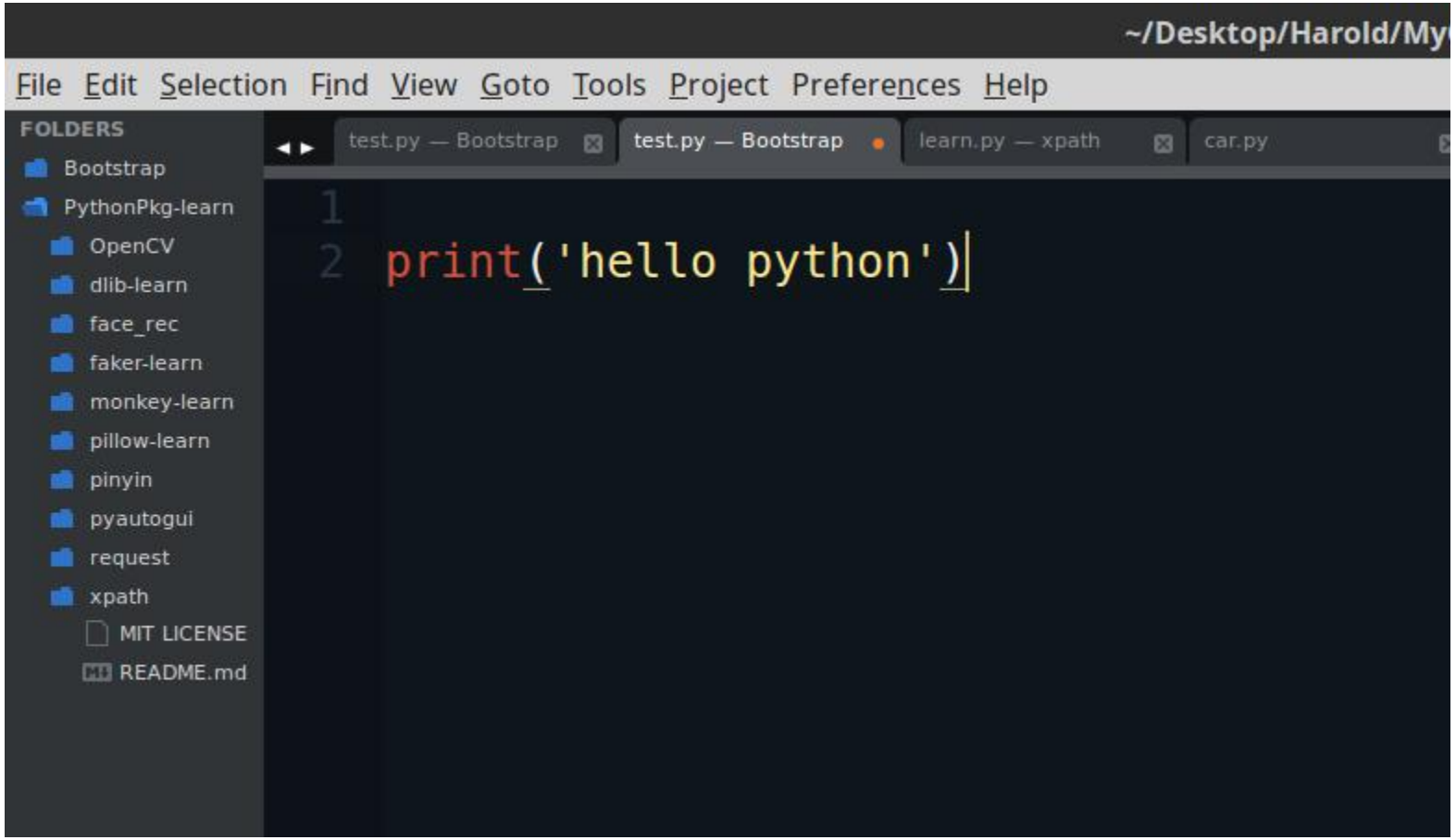
- 交互式编程
  - 命令行
  - ipython
  - jupyter notebook





1 第一个Python程序

- 脚本式编程
  - Sublime-text
  - pycharm
  - atom



2 变量命名及类型

- 变量命名
- 数据基本类型

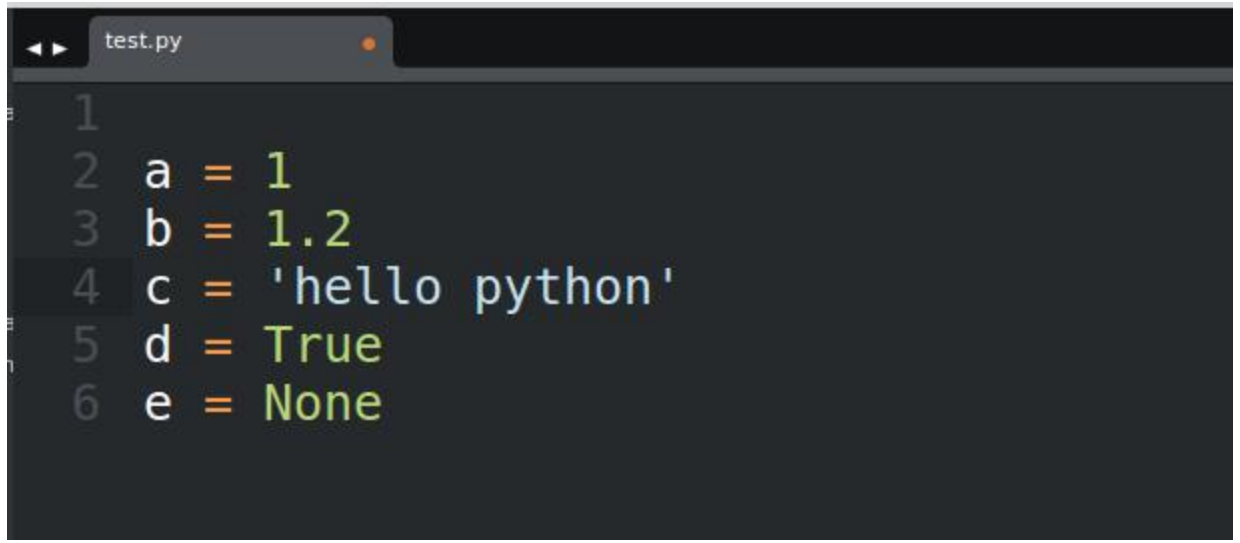
2 变量命名及类型

- 变量命名（驼峰代码写法）
  - 第一个字符必须是字母表中字母或下划线'_'。
  - 标识符的其他的部分有字母、数字和下划线组成。
  - 标识符对大小写敏感。
- 回避关键字

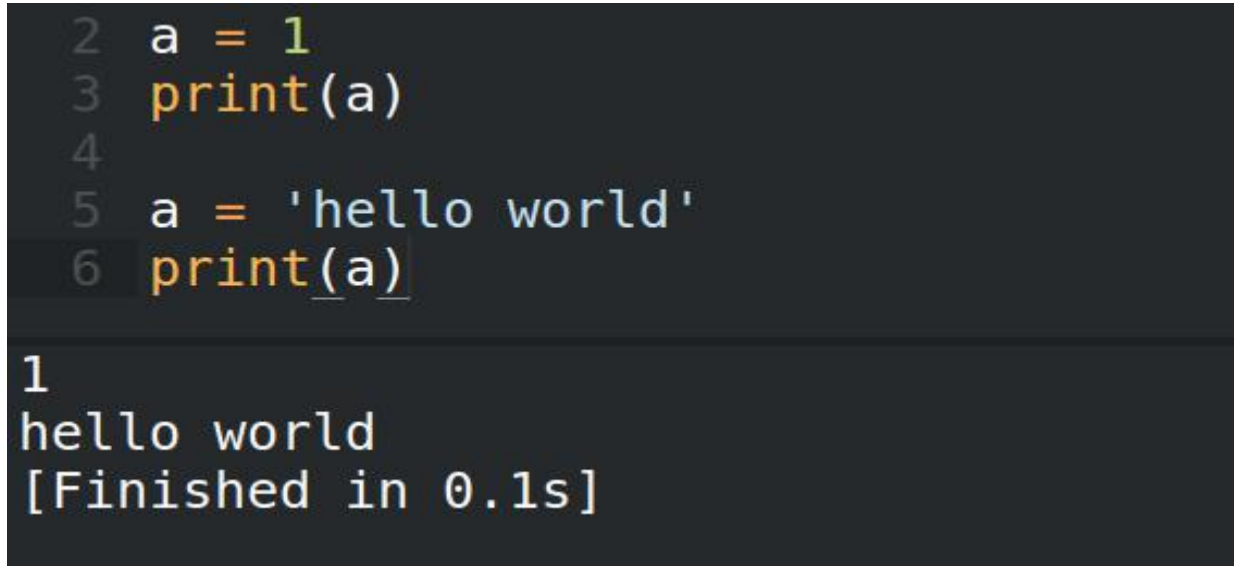
  'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'
- 代码注释
  - 单行注释用#
  - 多行注释''' '''

2 变量命名及类型

- 基本数据类型
  - 数字
  - 字符串
  - 真假值
  - 空



- 无需声明数据类型
- 字符串用引号
- 回收变量名



3 运算符

- 算术运算符
- 赋值运算符
- 比较运算符
- 逻辑运算符
- 成员运算符

3 运算符

- 算术运算符

以下假设变量a为10，变量b为21：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 31
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -11
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 210
/	除 - x 除以 y	b / a 输出结果 2.1
%	取模 - 返回除法的余数	b % a 输出结果 1
**	幂 - 返回x的y次幂	a**b 为10的21次方
//	取整除 - 返回商的整数部分	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

test.py

```
1  a = 10
2  b = 21
3
4  res1 = a + b
5  res2 = a - b
6  res3 = a * b
7  res4 = b / a
8  res5 = a**b
```

3 运算符

- 赋值运算符

以下假设变量a为10，变量b为20：

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

test.py

```
1  a = 10
2  b = 20
3  c = 'hello world'
4
5  d = e = f = 100
6  i,j,k = 10,20,'python'
7
8  a,b = b,a
```

3 运算符

- 比较运算符

以下假设变量a为10，变量b为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。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。	(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。

test.py

```
1  a = 10
2  b = 20
3
4  print(a==b)
5
6
7  if a != b:
8      print('no')
9  else:
10     print('yes')
```

3 运算符

- 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
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。
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。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。

test.py

```
1
2  hight = 180
3  face = 85
4
5  if hight>=175 and face>=80:
6      print('date')
7
8  elif hight>=180 or face>90:
9      print('date')
10
11 else:
12     print('no')
13
```

3 运算符

- 成员运算符

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

```
1  a = 'python'
2  b = 't'
3  c = 'm'
4
5  print(b in a)
6  print(c in a)
7  print(c not in a)
8
9  if b in a:
10     print('yes')
11 else:
12     print('no')
```

4 Python序列

- 列表
- 元组
- 列表
- 字符串

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

- 方口号
- 创建空列表
- 列表存入数据

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

test.py

```
1  c = [1,2,3,'python',1.2]
2
3  print(c[0])
4  print(c[1])
5  print(c[1:3])
6  print(c[2:])
7  print(c[:3])
```

```
1
2
[2, 3]
[3, 'python', 1.2]
[1, 2, 3]
```

- 方口号
- 索引
- 冒号

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

test.py

```
1  c = [1,2,3,'python',1.2]
2
3  print(c[2])
4
5  c[2] = 'hello'
6  print(c)
```

```
3
[1, 2, 'hello', 'python', 1.2]
```

- 方口号
- 索引
- =

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

```
1 c = [1,2,3,'python',1.2]
2
3 del c[3]
4
5 print(c)
6
7 del c
```

```
[1, 2, 3, 1.2]
```

- del使用

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

```
1 c = [1,2,3,'python',1.2]
2
3 del c[3]
4
5 print(c)
6
7 del c
```

```
[1, 2, 3, 1.2]
```

- del使用

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

- len（）获取长度
- 列表相加+
- 列表乘法*
- 成员判断

test.py

```
1  c = [1,2,3,'python',1.2]
2  print(len(c))
3
4  b = [1,'are','you','ok']
5  c = [2,'how','are','u']
6  d = b + c
7  print(d)
8
9  e = b*2
10 print(e)
11
12 res = 'are' in b
13 print(res)
```

```
5
[1, 'are', 'you', 'ok', 2, 'how', 'are', 'u']
[1, 'are', 'you', 'ok', 1, 'are', 'you', 'ok']
True
```

4 Python序列

- 列表
  - 创建列表
  - 访问元素
  - 更新元素值
  - 删除列表
  - 列表操作
  - 函数&方法

test.py

```
1 a = [5,1,4,3,6]
2 b = 'python'
3
4 print( len(a) )
5 print( max(a) )
6 print( min(a) )
7 print( sum(a) )
8 print( list(b) )
```

函数：len( )、max( )、min( )、sum( )、list( )

方法：.append( )、.insert( )、.remove( )、.pop( )、.index( )

test.py

```
1  a = [5,1,4,3,6]
2
3  a.append('python')
4  print(a)
5
6  a.insert(1,'hello')
7  print(a)
8
9  a.remove(5)
10 print(a)
11
12 a.pop()
13 print(a)
14
15 print( a.index(6) )
```

```
[5, 1, 4, 3, 6, 'python']
[5, 'hello', 1, 4, 3, 6, 'python']
['hello', 1, 4, 3, 6, 'python']
['hello', 1, 4, 3, 6]
4
```

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 删除元组
  - 元组操作
  - 函数&方法

- Python 的元组与列表类似，不同之处在于列表的元素不能修改。

- 列表使用小括号，列表使用方括号。

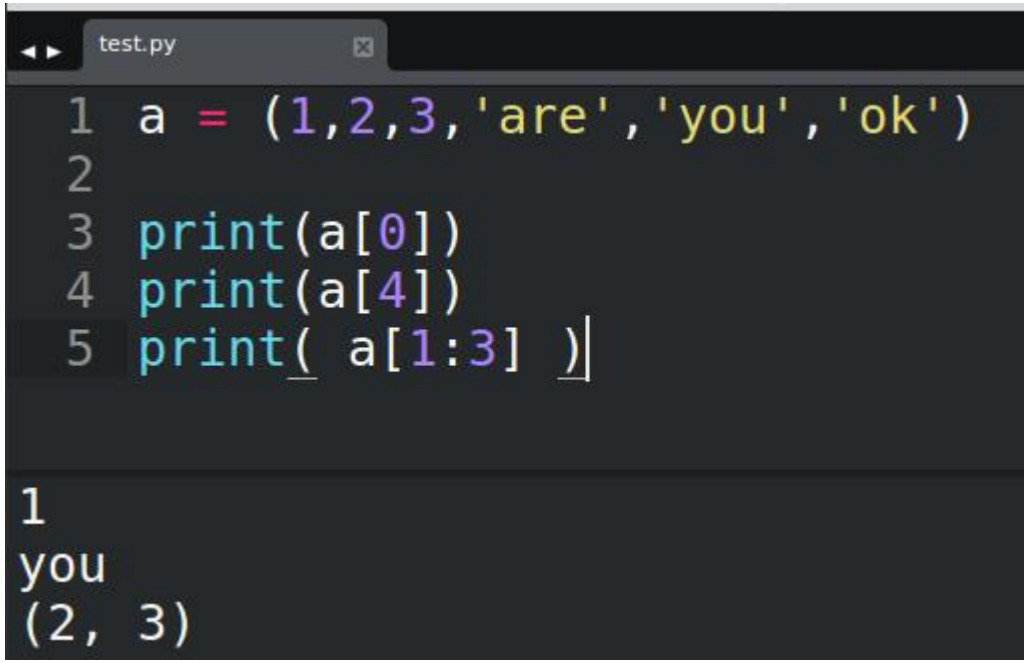
test.py

```
1 a = (1,2,3)
2 b = 4,5,6
3 c = (1,3.14,'python')
4
5 print(a)
6 print(b)
7 print(c)
```

```
(1, 2, 3)
(4, 5, 6)
(1, 3.14, 'python')
```

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 删除元组
  - 元组操作
  - 函数&方法



  - 下标访问
  - :号多元素访问，切片操作

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 删除元组
  - 元组操作
  - 函数&方法

test.py

```
1 a = (1,2,3,'are','you','ok')
2 b = (1,2,3)
3
4 del a
5 del b[1] #不能这样操作
```

- 元组只把整个元组删除
- 不能删除单个元素

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 删除元组
  - 元组操作
  - 函数&方法

Python 表达式	结果	描述
len((1, 2, 3))	3	计算元素个数
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	连接
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	复制
3 in (1, 2, 3)	True	元素是否存在
for x in (1, 2, 3): print x,	1 2 3	迭代

test.py

```
1  a = (1,2,3)
2  b = (4,5,6)
3
4  print(len(a))
5
6  c = a + b
7  print(c)
8
9  d = c*2
10 print(d)
11
12 res = 7 in b
13 print(res)
```

```
3
(1, 2, 3, 4, 5, 6)
(1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6)
False
```

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 删除元组
  - 元组操作
  - 函数&方法

Python 表达式	结果	描述
len((1, 2, 3))	3	计算元素个数
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	连接
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	复制
3 in (1, 2, 3)	True	元素是否存在
for x in (1, 2, 3): print x,	1 2 3	迭代

test.py

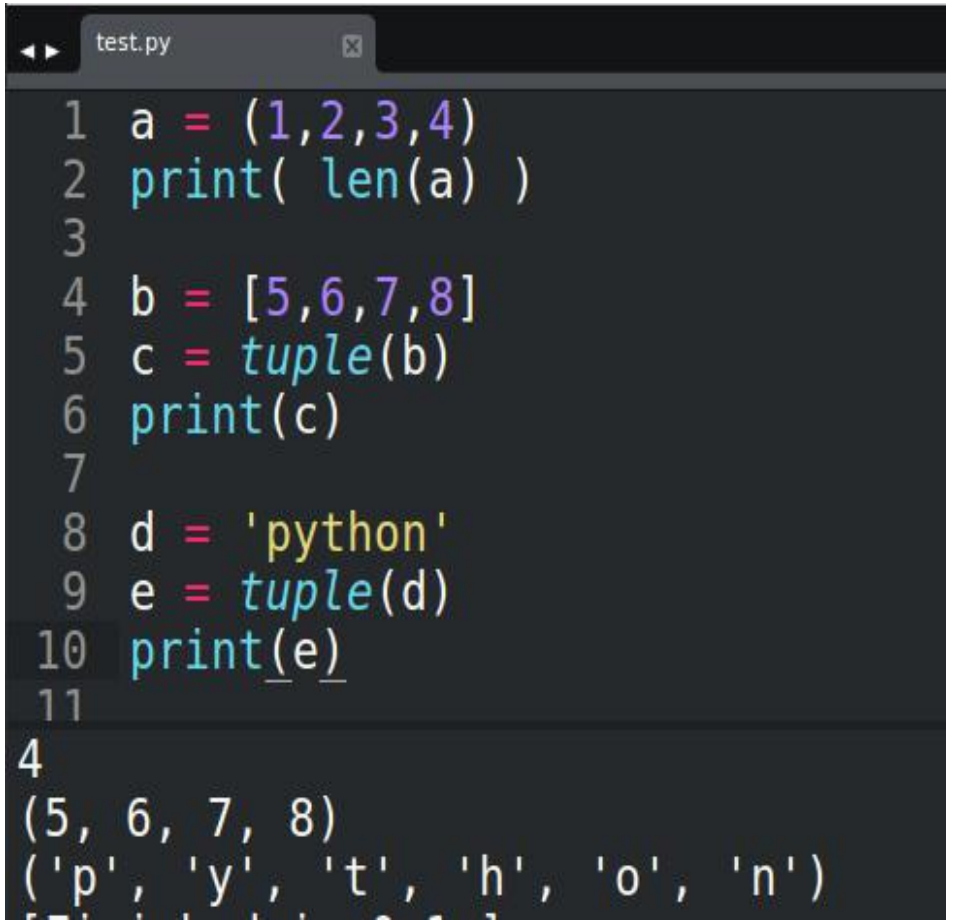
```
1  a = (1,2,3)
2  b = (4,5,6)
3
4  print(len(a))
5
6  c = a + b
7  print(c)
8
9  d = c*2
10 print(d)
11
12 res = 7 in b
13 print(res)
```

```
3
(1, 2, 3, 4, 5, 6)
(1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6)
False
```

4 Python序列

- 元组
  - 创建元组
  - 访问元素
  - 更新元素值
  - 删除元组
  - 元组操作
  - 函数&方法

函数：len( )、max( )、min( )、sum( )、tuple( )



4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2  a = {}
3
4  print(d)
5
```

```
{'school': 'it school', 'name': 'harold', 'id': 2017030121}
```

4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2  a = {}
3
4  print(d['id'])
5
```

```
2017030121
[Finished in 0.1s]
```

4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2  a = {}
3
4  d['name'] = 'adam'
5  print(d)
6
7  a['name'] = 'alice'
8  print(a)
9
```

```
{'id': 2017030121, 'school': 'it school', 'name': 'adam'}
{'name': 'alice'}
```

4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2
3  del d['name'] #删除一个元素
4  d.clear()    #清空字典
5  del d    #删除整个字典
```

4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2
3  del d['name'] #删除一个元素
4  d.clear()    #清空字典
5  del d    #删除整个字典
```

4 Python序列

- 字典
  - 创建字典
  - 访问元素
  - 更新元素值
  - 删除字典
  - 函数&方法

函数：len( )

方法：.keys( )、.values( )、.items( )

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2
3  l = len(d)
4  print(l)
```

```
3
```

test.py

```
1  d = {'name':'harold','id':2017030121,'school':'it school'}
2
3  k = d.keys()
4  print(k)
5
6  v = d.values()
7  print(v)
8
9  i = d.items()
10 print(i)
```

```
dict_keys(['id', 'name', 'school'])
dict_values([2017030121, 'harold', 'it school'])
dict_items([('id', 2017030121), ('name', 'harold'), ('school', 'it school')])
```

5 Python控制语句

- 条件语句
- 循环语句
- 缩进
- break、continue、pass

5 Python控制语句

- 条件语句
  - 单分支
  - 多分支
  - 缩进与嵌套

一定要注意缩进，与C区别{}

```
1  python_score = 70
2
3  if python_score >=75:
4      print('A')
5
6  elif python_score >=60:
7      print('B')
8
9  else:
10     print('恭喜你挂了')
11
12
```

密码错误

test.py

```
1  score = 65
2
3  # 单分支语句
4  if score >=60:
5      print('通过')
6
```

通过

test.py

```
1  score = 75
2
3  # 双分支语句
4  if score >= 60:
5      print('通过')
6
7  else:
8      print('挂科')
9
```

通过

```
1  score = 75
2
3  # 多分支语句
4  if score >=90:
5      print('A')
6
7  elif score>=80:
8      print('B')
9
10 elif score >=70:
11     print('C')
12
13 elif score >=60:
14     print('D')
15
16 else:
17     print('no')
18
```

C

5 Python控制语句

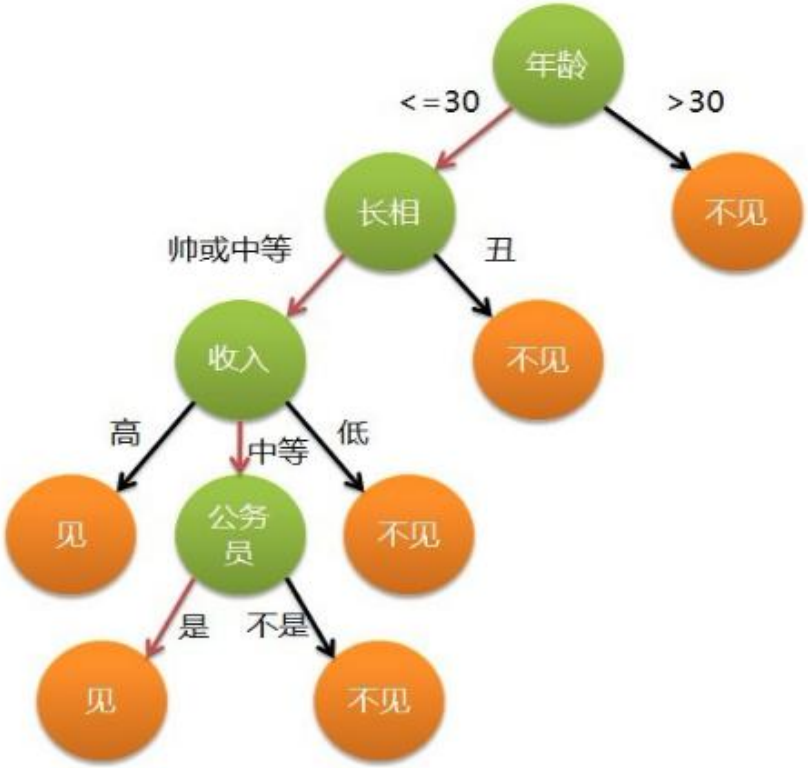
- 条件语句
  - 单分支
  - 多分支
  - 缩进与嵌套

一定要注意缩进，与C区别{}

test.py

```
1  userName = 'alice'
2  password = '123456'
3
4
5  if userName =='alice':
6
7      if password == '111111':
8          print('登录')
9      else:
10         print('密码错误')
11 else:
12     print('该用户没注册')
13
```

```
密码错误
```



5 Python控制语句

- 循环语句
  - for循环
  - while循环
  - 缩进与嵌套

一定要注意缩进，与C区别{}

```c
#include <stdio.h>

int main ()
{
   /* for 循环执行 */
   for( int a = 10; a < 20; a = a + 1 )
   {
      printf("a 的值:  %d\n", a);
   }

   return 0;
}
```

test.py

```python
# for循环
for i in range(10):
    print('hello python')


# in后面跟的是序列(列表，元组，字符串等)
seq = [4,5,6]
for i in seq:
    print(i)


t = (1,3,4)
for i in t:
    print(i)


for i in 'python':
    print(i)
```

test.py

```python
# range设置
# range(起始值，终止值，步长)

for i in range(10):
    print(i)

for i in range(0,10,2):
    print(i)
```

```python
# 循环嵌套

for i in range(10):
    for j in range(10):

        print(i,j)

```

5 Python控制语句

- 循环语句
  - for循环
  - while循环
  - 缩进与嵌套

一定要注意缩进，与C区别{}

```c
#include <stdio.h>

int main ()
{
   /* 局部变量定义 */
   int a = 10;

   /* while 循环执行 */
   while( a < 20 )
   {
      printf("a 的值:  %d\n", a);
      a++;
   }

   return 0;
}
```

test.py

```python

i = 1

while i <=10:
    print(i)
    i = i + 1

k = 1
while k <=10:
    print('python')
    k +=1

```

test.py

```python
# 不知到循环次数
i = 1
while i<=10:
    print('hello python')
    i = i + 0.3


# 循环嵌套
j = 1
k = 1
while j <=10:

    while k <=10:
        print(j,k)
        k = k + 1

    j = j + 1
```

5 Python控制语句

- break、continue语句
  - Python break语句，就像在C语言中，打破了最小封闭for或while循环
  - Python continue 语句跳出本次循环，而break跳出整个循环

test.py

```
1  # break使用
2  for i in range(5):
3
4      for j in range(5):
5
6          print(i,j)
7          if i ==3 and j==3:
8              break
9
```

test.py

```
1  # break使用
2  for i in range(10):
3      print(i)
4
5      if i ==5:
6          break
7
8  #break使用
9  for j in 'python':
10     print(j)
11
12     if j == 'h':
13         break
14
```

test.py

```
1  # continue
2  for i in range(10):
3
4      if i ==5:
5          continue
6
7      print(i)
8
9
10
11 for i in 'python':
12
13     if i == 'h':
14         continue
15     print(i)
16
```

6 Python函数

- 函数定义
- 函数调用
- 函数参数
- return

6 Python函数

- 函数
  - 函数定义
  - 函数调用
  - 函数参数
  - return

```
def 函数名（参数列表）：
    函数体
```

定义函数def

函数名

```
test.py
1  # 定义hello()函数
2
3
4  def hello():
5      print('hello python')
6
7
8  def hello1(n):
9      '''n是int,打印次数
10     '''
11     for i in range(n):
12         print('hello1 python')
13
14
15 def hello2(n,words):
16     '''n是int,打印次数
17        words是string, 打印的内容
18     '''
19     for i in range(n):
20         print(words)
21
22 hello()
23 hello1(5)
24 hello2(3,'hello2 python')
```

形参

函数体

实参

```
/* 函数返回两个数中较大的那个数 */
int max(int num1, int num2)
{
    /* 局部变量声明 */
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

带返回值的函数

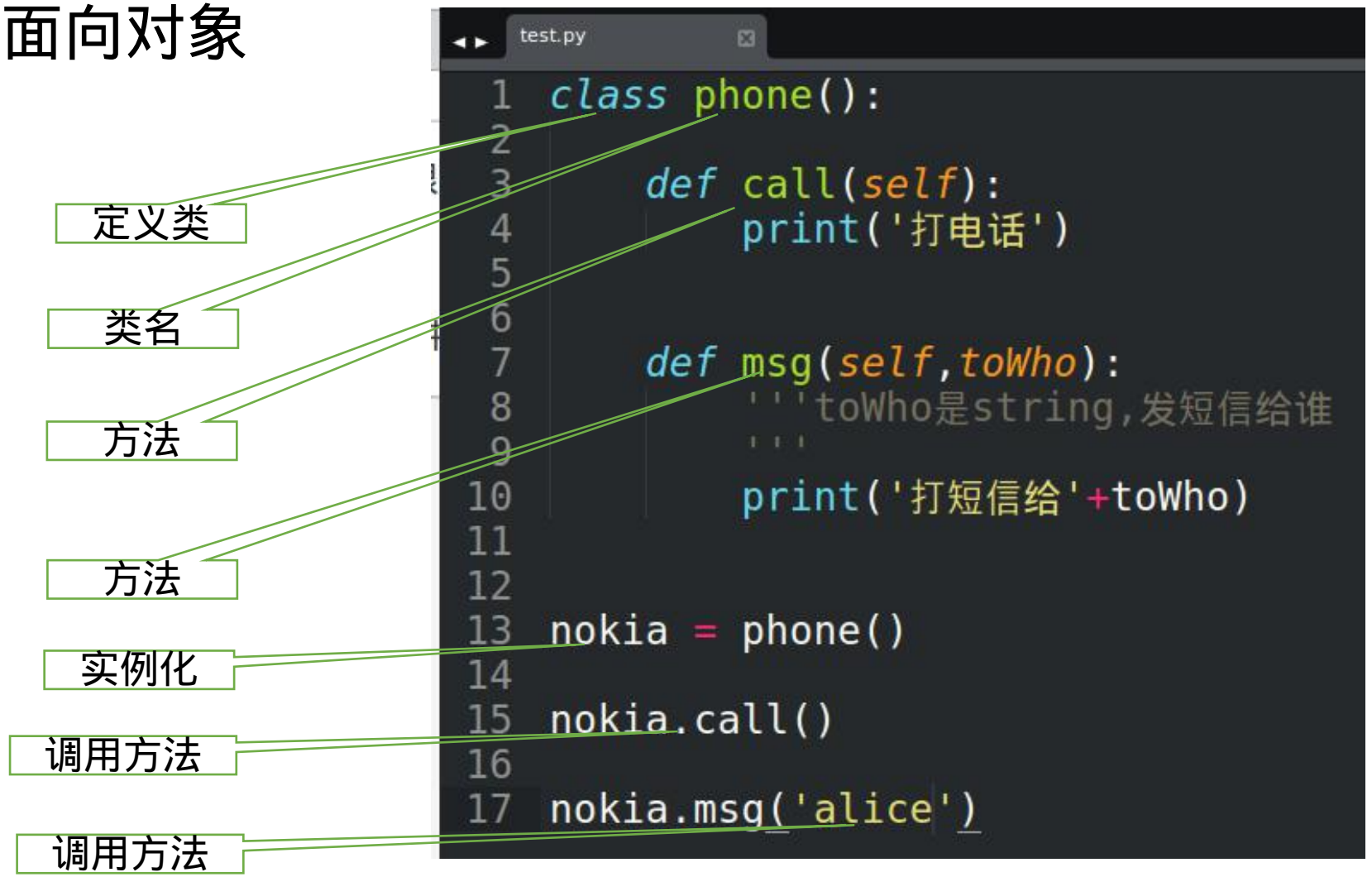
```
test.py
1  # 定义hello()函数
2
3
4  def add(n):
5      '''n是int
6      '''
7      res = 0
8      for i in range(1,n+1):
9
10         fz = 1
11         fm = i
12
13         res = res + fz/fm
14
15     return res
16
17
18
19 r = add(5)
20 print(r)
```

7 Python面向对象

- 类定义
- 方法与属性
- 实例化
- 方法调用
- 继承
- 构造方法、普通方法、类方法、静态方法

7 Python面向对象

- 面向对象



- 定义类
- 类中写方法（self）
- 实例化类
- 调用方法

7 Python面向对象

- 面向对象
  - 类的继承
  - 构造方法
  - 普通方法
  - 静态方法
  - 类方法

```python
class phone():

    def call(self):
        print('打电话')


    def msg(self,toWho):
        '''toWho是string,发短信给谁
        '''
        print('打短信给'+toWho)



class mi(phone):
    #继承phone,让其具备打电话call，发短信msg功能


    def __init__(self,series,price):#构造方法，实例化是传参
        '''series是string，机型
           price是float，是价钱
        '''
        self.series = series
        self.price = price


    def qq(self): #普通方法，调用时传参
        print('可以聊qq')


    @staticmethod #静态方法装饰器，须在方法头上,没有self，cls等默认参数
    def weChat():
        print('可以聊微信')


    @classmethod #类方法装饰器，须传cls参数
    def brower(cls):
        print('可以上网')



mi6 = mi('6','2999') #有构造方法时须传参
print(mi6.price)     #访问属性

mi6.call()  #可以拥有phone的方法
mi6.weChat() #调用静态方法
```

8 其他基础

- 包导入
- 程序排错
- 包安装与管理

8 其他基础

- 其他基础
  - 包导入与使用
  - 程序排错

```
# 两种方法导入包
# import ...
# from ... import ...
```

```
# 导入包
import time

for i in range(10):
    print(i)
    time.sleep(2)#调用包的方法


```

```
# 导入包
from time import sleep

for i in range(10):
    print(i)
    sleep(2) #这样就可以直接用sleep方法了

```

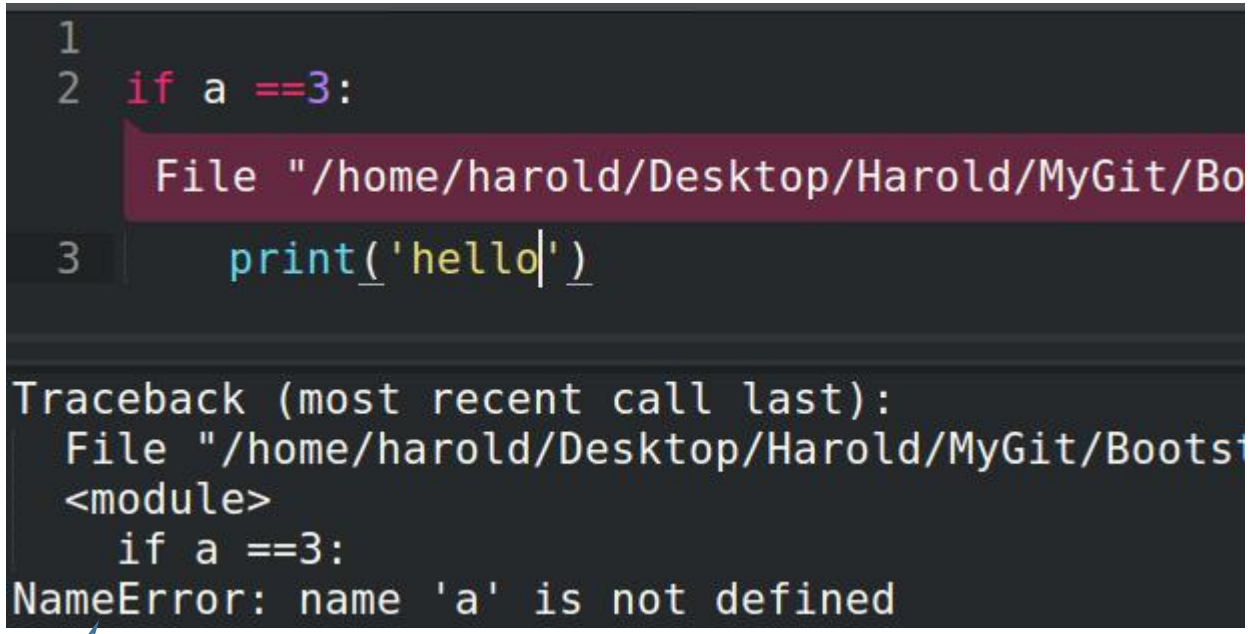
```
# 导入包
import time as t #把time重命名为t

for i in range(10):
    print(i)
    t.sleep(2) #调用时用t.sleep()
```

8 其他基础

- 其他基础
  - 包导入与使用
  - 程序排错

常犯的几种错误

- SyntaxError 语法错误
- IndentationError 缩进错误
- NameError 未定义或声明
- ImportError 导入模块错误
- IndexError 下标错误



未定义错误

8 其他基础

- pip包管理工具

```
#安装flask包
pip install flask


#卸载flask包
pip uninstall flask


#查看安装的包
pip list
```