

5.3. 使用list和tuple

5.4. 条件判断

5.5. 模式匹配

5.6. 循环

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

11.1. 使用_slots_

11.2. 使用@property

11.3. 多重继承

11.4. 定制类

11.5. 使用枚举类

11.6. 使用元类

12. 错误、调试和测试

12.1. 错误处理

12.2. 调试

12.3. 单元测试

12.4. 文档测试

13. IO编程

高级特性



廖雪峰



资深软件开发工程师，业余马拉松选手。

掌握了Python的数据类型、语句和函数，基本上就可以编写出很多有用的程序了。

比如构造一个 `1, 3, 5, 7, ..., 99` 的列表，可以通过循环实现：

```
L = []
n = 1
while n <= 99:
    L.append(n)
    n = n + 2
```

取list的前一半的元素，也可以通过循环实现。

但是在Python中，代码不是越多越好，而是越少越好。代码不是越复杂越好，而是越简单越好。

基于这一思想，我们来介绍Python中非常有用高级特性，1行代码能实现的功能，决不写5行代码。请始终牢记，代码越少，开发效率越高。

« 递归函数

切片 »



Comments

Comments loaded. To post a comment, please Sign In



Meow @ 2025/11/13 03:26:06

构造一个`1, 3, 5, 7, ..., 99`的列表：

```
lst = [i for i in range(1, 100, 2)]
print(lst)
```



失眠的树 @ 2025/9/19 01:30:46

```
nums = list(range(1, 100, 2))
print(nums)
```



禅心梵音见悟佛国国主 @ 2025/7/17 04:30:21

```
list(range(1, 100, 2))
```

禅心梵音见悟佛国国主 @ 2025/7/17 04:28:30

```
np.arange(1, 100, 2).tolist()
```

@ 2025/7/15 08:20:36

```
L = []
for i in range(1,100,2):
    L.append(i)
print(L)
```

秋水揽星河 @ 2025/7/6 01:25:20

```
1
```

王瑞东 @ 2025/3/19 01:56:14

```
print([i for i in range(1, 100, 2)])
```

惜简 @ 2025/7/1 04:55:14

```
厉害了我的哥
```

😊 @ 2025/5/29 22:55:05

```
day6
```

路 @ 2025/5/6 21:05:31

```
7.1. 切片 done
```

青笺画卿颜 @ 2025/4/20 21:41:01

```
打卡
```

寥寥星辰 @ 2025/4/13 09:31:56

```
day5 L=[] n=1 while n<=99: L.append(n) n=n+2 l=[] m=0 while m<=25: L1=L[m] l.append(L1) m=m+1 print(l)
```

Qiang @ 2025/4/8 02:55:13

第一种

```
L = []
n = 1
while n <= 99:
    L.append(n)
    n = n + 2
print(L)
```

第二种

```
L = []
for n in range(1,100,2):
    L.append(n)
print(L)
```

李清照 @ 2025/3/12 06:02:11

5.3. 使用list和tuple

5.4. 条件判断

5.5. 模式匹配

5.6. 循环

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

11.1. 使用__slots__

11.2. 使用@property

11.3. 多重继承

11.4. 定制类

11.5. 使用枚举类

11.6. 使用元类

12. 错误、调试和测试

12.1. 错误处理

12.2. 调试

12.3. 单元测试

12.4. 文档测试

13. IO编程

- 5.3. 使用list和tuple
- 5.4. 条件判断
- 5.5. 模式匹配
- 5.6. 循环
- 5.7. 使用dict和set

6. 函数

- 6.1. 调用函数
- 6.2. 定义函数
- 6.3. 函数的参数
- 6.4. 递归函数

7. 高级特性

- 7.1. 切片
- 7.2. 迭代
- 7.3. 列表生成式
- 7.4. 生成器
- 7.5. 迭代器

8. 函数式编程

- 8.1. 高阶函数
 - 8.1.1. map/reduce
 - 8.1.2. filter
 - 8.1.3. sorted
- 8.2. 返回函数
- 8.3. 匿名函数
- 8.4. 装饰器
- 8.5. 偏函数

9. 模块

- 9.1. 使用模块
- 9.2. 安装第三方模块

10. 面向对象编程

- 10.1. 类和实例
- 10.2. 访问限制
- 10.3. 继承和多态
- 10.4. 获取对象信息
- 10.5. 实例属性和类属性

11. 面向对象高级编程

- 11.1. 使用_slots_
- 11.2. 使用@property
- 11.3. 多重继承
- 11.4. 定制类
- 11.5. 使用枚举类
- 11.6. 使用元类

12. 错误、调试和测试

- 12.1. 错误处理
- 12.2. 调试
- 12.3. 单元测试
- 12.4. 文档测试

13. IO编程

```
L, n = [], 1
while n <= 99:
    L.append(n)
    n += 2
print(L)
```

李清照 @ 2025/3/12 06:04:39

```
li = [i for i in range(1, 100, 2)]
print(li)
```

樺 @ 2024/11/29 00:19:45

```
L = [] # method 1
for x in range(0,100):
    if x % 2 == 0:
        continue
    L.append(x)
print(L)
```

2425 @ 2024/11/5 03:33:29

```
L=[] i=1 while i <=99: L.append(i) i+=2 print(L)
```

楠梓 @ 2024/11/4 01:38:46

```
l=[]
n=1
while n<=99:
    l.append(n)
    n=n+2
print(sum(l))
print(len(l))
print(l)
```

楠梓 @ 2024/11/4 01:39:47

```
2500
50
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43]
```

陈先森 @ 2024/8/25 12:06:54

```
L = [] L = list(range(1,100,2))
```

木木 @ 2024/8/19 03:14:45

```
li = [i for i in range(1, 100, 2)]
```

Breeze @ 2024/8/2 21:57:37

```
L = []
n = 1
while n <= 99:
    L.append(n)
    n = n + 2
print(L)
```



袖子稚气 @ 2024/7/30 20:29:38

L = [] n=1 while n<=99: L.append(n) n = n+2 R =[]

for i in range(len(L) // 2):

```
R.append(L[i])
```

```
print(len(R))
```

5.3. 使用list和tuple

5.4. 条件判断

5.5. 模式匹配

5.6. 循环

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

11.1. 使用__slots__

11.2. 使用@property

11.3. 多重继承

11.4. 定制类

11.5. 使用枚举类

11.6. 使用元类

12. 错误、调试和测试

12.1. 错误处理

12.2. 调试

12.3. 单元测试

12.4. 文档测试

13. IO编程

©liaoxfeng.com - 微博 - GitHub - License