

# Python3 标准库

Stephen CUI

2023-09-20

# Contents

<b>1</b>	<b>文本</b>	<b>3</b>
1.1	re: 正则表达式 . . . . .	3
1.1.1	模式语法 . . . . .	3
<b>2</b>	<b>算法</b>	<b>5</b>
2.1	functools: 管理函数的工具 . . . . .	5
2.1.1	缓存 . . . . .	5
2.1.2	归约数据集 . . . . .	5
2.1.3	泛型函数 . . . . .	5
2.2	itertools: 迭代器函数 . . . . .	6
2.2.1	合并和分解迭代器 . . . . .	6
2.2.2	转换输入 . . . . .	6
2.2.3	生成新值 . . . . .	6
2.2.4	过滤 . . . . .	7
2.2.5	数据分组 . . . . .	7
2.2.6	合并输入 . . . . .	7
2.3	operator: 内置操作符的函数接口 . . . . .	7
2.3.1	逻辑操作 . . . . .	8
2.3.2	比较操作符 . . . . .	8
2.3.3	算术操作符 . . . . .	8
2.3.4	序列操作符 . . . . .	8
2.3.5	就地操作符 . . . . .	8
2.3.6	属性和元素“获取方法” . . . . .	8
2.3.7	结合操作符和定制类 . . . . .	8
<b>3</b>	<b>日期和时间</b>	<b>9</b>
3.1	time: 时钟日期 . . . . .	9
3.2	datetime: 日期和时间值管理 . . . . .	9
3.3	calendar: 处理日期 . . . . .	9

*CONTENTS* 3

**4** **10**

4.1 inspect: Inspect Live Objects . . . . . 10

4.1.1 检查模块 . . . . . 10

# Chapter 1

## 文本

### 1.1 re: 正则表达式

#### 1.1.1 模式语法

##### 重复

模式中有 5 中表示重复的方法

1. 元字符 `*`，则表示重复 0 次或多次（运行一个模式重复 0 次是指这个模式即使不出现也可以出现匹配）
2. `+`，那么模式必须至少出现 1 次才能匹配
3. `?` 表示出现 0 次或 1 次
4. 如果需要指定出现次数，需要在模式后面使用 `{m}`，这里 *m* 是模式应重复的次数
5. 如果要允许一个可变但有限的重复次数，那么可以使用 `{m,n}`，这里 *m* 是最小重复次数，*n* 是最大重复次数。如果省略 *n* (*m*,) 则表示值必须知道出现 *m* 次，但没有最大限制

##### 字符集

A character set is a group of characters, any one of which can match at that point in the pattern. For example, `[ab]` would match either a or b.

A character set can also be used to exclude specific characters. The `caret` `^` means to look for characters that are not in the set following the `caret`.

As character sets grow larger, typing every character that should (or should not) match becomes tedious. A more compact format using **character ranges** can be used to define a character set to include all of the contiguous characters between the specified start and stop points.

As a special case of a character set, the meta-character dot, or period (`.`), indicates that the pattern should match any single character in that position.

##### 转义码

##### 锚定

使用锚定指令指定模式在输入文本中的相对位置。

Table 1.1: Regular Expression Escape Codes

Code	Meaning
d	A digit
D	A non-digit
s	Whitespace (tab, space, newline, etc.)
S	Non-whitespace
w	Alphanumeric
W	Non-alphanumeric

Table 1.2: 正则表达式锚定码

锚定码	含义
^	字符串或行的开头
\$	字符串或行末尾
A	字符串开头
Z	字符串末尾
b	单词开头或末尾的空串
B	不在单词开头或末尾的空串

# Chapter 2

## 算法

Python 包含很多模块，可以采用最适合任务的方式来精巧而简洁地实现方式。它支持不同的编程方式，包括纯过程式、面向对象式和函数式。这 3 中方式经常在同一个程序中不同部分混合使用。

`functools` 包含的函数用于与创建函数修饰符、启用面向方面（`aspect-oriented`）编程以及传统面向对象方法不能支持的代码重用。它还提供一个类修饰符以使用一个快捷方式来实现所有富比较 API，另外提供了 `partial` 对象来创建函数（包含其参数）的引用。

`itertools` 模块包含的函数用于创建和处理函数式编程中使用的迭代器和生成器。通过提供基于函数的内置操作接口（如算术操作和元素查找）。

`operator` 模块在使用函数式编程时不在需要很多麻烦的 `lambda` 函数。

不论使用哪一种编程方式，`contextlib` 都会让资源管理更容易、更可靠且更简洁。结合上下文管理器和 `with` 语句，可以减少 `try:finally` 块的个数和所需的缩进层次，同时还能确保文件、套接字、数据库事务和其他资源在适当的时候关闭和释放。

### 2.1 `functools`: 管理函数的工具

#### 2.1.1 缓存

The `lru_cache()` decorator wraps a function in a “least recently used” cache. Arguments to the function are used to build a hash key, which is then mapped to the result. Subsequent calls with the same arguments will fetch the value from the cache instead of calling the function. The decorator also adds methods to the function to examine the state of the cache (`cache_info()`) and empty the cache (`cache_clear()`).

#### 2.1.2 归约数据集

The `reduce()` function takes a callable and a sequence of data as input. It produces a single value as output based on invoking the callable with the values from the sequence and accumulating the resulting output.

#### 2.1.3 泛型函数

在类似 Python 的动态类型语言中，通常需要基于参数的类型完成稍有不同的操作，特别是在处理元素列表与单个元素的差别时。直接检查参数的类型固然很简单，但是有些情况下，行为差异可

能需要被隔离到单个的函数中，对于这些情况，`functools` 提供了 `singledispatch()` 修饰符来注册一组泛型函数（generic function），可以根据函数第一个参数的类型自动切换。

## 2.2 `itertools`: 迭代器函数

`itertools` 包括一组用于处理序列数据集的函数。

### 2.2.1 合并和分解迭代器

`chain()` 函数取多个迭代器作为参数，最后返回一个迭代器，它会生成所有输入迭代器的内容，就好像这些内容来自一个迭代器一样。利用 `chain()`，可以轻松地处理多个序列而不必构造一个很大的列表。

If the iterables to be combined are not all known in advance, or if they need to be evaluated lazily, `chain.from_iterable()` can be used to construct the chain instead.

The built-in function `zip()` returns an iterator that combines the elements of several iterators into tuples. As with the other functions in this module, the return value is an iterable object that produces values one at a time. `zip()` stops when the first input iterator is exhausted. To process all of the inputs, even if the iterators produce different numbers of values, use `zip_longest()`. 也就是说最短的那个迭代器迭代完就结束，只要遇到 `StopIteration` 就结束。By default, `zip_longest()` substitutes `None` for any missing values. Use the `fillvalue` argument to use a different substitute value.

The `islice()` function returns an iterator that returns selected items from the input iterator, by index. `islice()` takes the same arguments as the slice operator for lists: start, stop, and step. The start and step arguments are optional.

The `tee()` function returns several independent iterators (defaults to 2) based on a single original input. `tee()` 返回的迭代器可以用来为并行处理的多个算法提供相同的数据集。`tee()` 创建的新迭代器会共享器输入迭代器，所有创建了新迭代器后，不应再使用原迭代器。

### 2.2.2 转换输入

The built-in `map()` function returns an iterator that calls a function on the values in the input iterators, and returns the results. It stops when any input iterator is exhausted.

The `starmap()` function is similar to `map()`, but instead of constructing a tuple from multiple iterators, it splits up the items in a single iterator as arguments to the mapping function using the `*` syntax. Where the mapping function to `map()` is called `f(i1,i2)`, the mapping function passed to `starmap()` is called `f(*i)`.

### 2.2.3 生成新值

The `count()` function returns an iterator that produces consecutive integers, indefinitely. The first number can be passed as an argument (the default is zero). There is no upper bound argument.

The `cycle()` function returns an iterator that repeats the contents of the arguments it is given indefinitely. Because it has to remember the entire contents of the input iterator, it may consume quite a bit of memory if the iterator is long.

The `repeat()` function returns an iterator that produces the same value each time it is accessed.

### 2.2.4 过滤

The `dropwhile()` function returns an iterator that produces elements of the input iterator after a condition becomes false for the first time.

The opposite of `dropwhile()` is `takewhile()`. It returns an iterator that itself returns items from the input iterator as long as the test function returns true.

The built-in function `filter()` returns an iterator that includes only items for which the test function returns true.

`filterfalse()` returns an iterator that includes only items where the test function returns false.

`compress()` offers another way to filter the contents of an iterable. Instead of calling a function, it uses the values in another iterable to indicate when to accept a value and when to ignore it.

### 2.2.5 数据分组

The `groupby()` function returns an iterator that produces sets of values organized by a common key. 输入的序列要根据键值排序，以保证得到预期的分组。

### 2.2.6 合并输入

The `accumulate()` function processes the input iterable, passing the  $n$ th and  $n + 1$ st item to a function and producing the return value instead of either input. The default function used to combine the two values adds them, so `accumulate()` can be used to produce the cumulative sum of a series of numerical inputs. `accumulate()` 可以与任何取两个输入值得函数结合来得到不同得结果。

迭代处理多个序列的嵌套 `for` 循环通常可以被替换为 `product()`，它会生成一个迭代器，值为输入值集合的笛卡尔积。The values produced by `product()` are tuples, with the members taken from each of the iterables passed in as arguments in the order they are passed. The first tuple returned includes the first value from each iterable. The last iterable passed to `product()` is processed first, followed by the next-to-last, and so on.

The `permutations()` function produces items from the input iterable combined in the possible permutations of the given length. It defaults to producing the full set of all permutations.

为了将值限制为唯一的组合而不是排列，可以使用 `combinations()`。只要输入的成员是唯一的，输出就不会包含任何重复的值。

## 2.3 operator: 内置操作符的函数接口

Programming using iterators occasionally requires creating small functions for simple expressions. Sometimes, these can be implemented as lambda functions, but for some operations new functions are not needed at all. The `operator` module defines functions that correspond to the built-in arithmetic, comparison, and other operations for the standard object APIs.



**2.3.1 逻辑操作**

**2.3.2 比较操作符**

**2.3.3 算术操作符**

**2.3.4 序列操作符**

The operators for working with sequences can be organized into four groups: building up sequences, searching for items, accessing contents, and removing items from sequences.

**2.3.5 就地操作符**

**2.3.6 属性和元素“获取方法”**

**2.3.7 结合操作符和定制类**

## Chapter 3

# 日期和时间

不同于 `int`、`float` 和 `str`，Python 没有包含对应日期和时间的原生类型，不过提供了 3 个相应的模块，可以采用多种表示来管理日期和时间值。

`time` 模块由底层 C 库提供与事件相关的函数。它包含一些函数，可以用于获取时钟时间和处理器运行时间，还提供了多种表示来管理日期和时间值。

`datetime` 模块为日期、时间以及日期时间值提供了一个更高层接口。`datetime` 中的类支持算术、比较和时区设置。

`calendar` 模块可以创建周、月和年的格式化表示。它还可以用来计算重复事件，给定日期是星期几，以及其他基于日历的值。

### 3.1 `time`: 时钟日期

The `time` module provides access to several types of clocks, each useful for different purposes. The standard system calls such as `time()` report the system “wall clock” time. The `monotonic()` clock can be used to measure elapsed time in a long-running process because it is guaranteed never to move backward, even if the system time is changed. For performance testing, `perf_counter()` provides access to the clock with the highest available resolution, which makes short time measurements more accurate. The CPU time is available through `clock()`, and `process_time()` returns the combined processor time and system time.

### 3.2 `datetime`: 日期和时间值管理

### 3.3 `calendar`: 处理日期

# Chapter 4

## 4.1 inspect: Inspect Live Objects

The inspect module provides functions for learning about live objects, including modules, classes, instances, functions, and methods. The functions in this module can be used to retrieve the original source code for a function, look at the arguments to a method on the stack, and extract the sort of information useful for producing library documentation for source code.

### 4.1.1 检查模块

第一种对现场对象的检查是为了了解这些对象。Use `getmembers()` to discover the member attributes of objects. The types of members that might be returned depend on the type of object scanned. Modules can contain classes and functions, classes can contain methods and attributes, and so on.

The arguments to `getmembers()` are an object to scan (a module, class, or instance) and an optional predicate function that is used to filter the objects returned. The return value is **a list of tuples** with two values: the name of the member and the type of the member. The inspect module includes several such predicate functions with names like `ismodule()`, `isclass()`, and so on.