Python编程新思维及实战

嵩 天

Python常用标准库解析(上)

嵩 天

Python常用标准库解析(上)

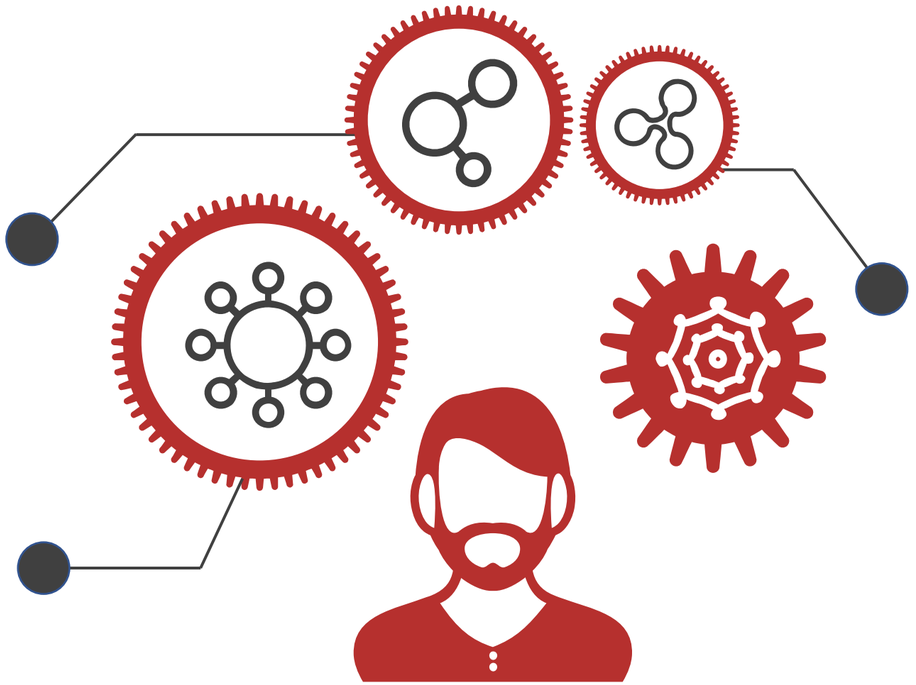
单元开篇

Python123

单元开篇

(1) time库的使用

(2) random库的使用



(3) re(正则表达式)库的使用

Python常用标准库解析(上)

Python常用标准库解析(上)

time库的使用

Python123

time库介绍

time库是Python中处理时间的标准库

- 库名：time，计算时间的表达
- 提供获取系统时间并格式化输出的功能
- 提供系统级精确计时功能，用于程序性能分析

https://docs.python.org/3.7/library/time.html

time库详解

time库的一些基本定义和概念

- 计时起点：1970年1月1日0时0分0秒，可以用time.gmtime(0)获得
- UTC时间：Coordinated Universal Time，世界标准时间，与GMT一致
- DST时间：Daylight Saving Time，夏令时时间，源于系统底层C函数
- struct_time：Python中用于保存时间对象、带有属性标签的数据类型

time库详解

time库的时间表示方法

- 浮点数时间：一般从计时起点开始计算，如：1535068435.3211677
- struct_time时间：便于程序员使用的内部结构，如：

  time.struct_time(tm_year=2018, tm_mon=6, tm_mday=25, tm_hour=3, tm_min=39, tm_sec=53,

  tm_wday=5, tm_yday=187, tm_isdst=0)

- 字符串时间：便于用户查看的时间，如：'Sat Jun 2 10:33:52 2018'

time库详解

理解struct_time

序号	属性名	取值范围
0	tm_year	四位整数，如 2018
1	tm_mon	[1, 12]
2	tm_mday	[1, 31]
3	tm_hour	[0, 23]
4	tm_min	[0, 59]
5	tm_sec	[0, 61] 60仅用于调时

序号	属性名	取值范围
6	tm_wday	[0, 6]，星期一为0
7	tm_yday	[1, 366]
8	tm_isdst	0: DST, 1: 非DST, -1: 随系统
N/A	tm_zone	时区的缩写表示
N/A	tm_gmtoff	与UTC时间相差的秒数

time库详解

理解struct_time

```
>>> t = time.gmtime()
time.struct_time(tm_year=2018, tm_mon=8, tm_mday=25, tm_hour=1, tm_min=53, tm_sec=43,
                 tm_wday=5, tm_yday=237, tm_isdst=0)
>>> t.tm_zone
'UTC'
>>> t.tm_gmtoff
0
>>> t[0]
2018
```

time库详解

time库函数的分类 (14个)

- 时间获取：time()、 gmtime()、 ctime()、 asctime() 、localtime()、mktime()
- 时间格式化：strftime()、strptime()
- 程序计时： clock()、monotonic()、perf_counter()、process_time()、sleep()
- 辅助函数：get_clock_info()

time库详解之时间获取

函数	描述
time()	返回一个从计时起点开始的表示时间的浮点数 >>>time.time() 1535104208.2367377
gmtime([s])	返回一个struct_time表示的时间，如果提供s，把s变成struct_time时间 >>>time.gmtime() time.struct_time(tm_year=2018, tm_mon=6, tm_mday=25, tm_hour=2, tm_min=46, tm_sec=42, tm_wday=5, tm_yday=187, tm_isdst=0)

time库详解之时间获取

函数	描述
localtime([s])	**返回一个struct_time表示的本地时间，如果提供s，把s变成本地时间** **>>>time.localtime()** **time.struct_time(tm_year=2018, tm_mon=6, tm_mday=25, tm_hour=2, tm_min=46, tm_sec=42, tm_wday=5, tm_yday=187, tm_isdst=0)**
mktime(t)	**将一个struct_time的本地时间t转变为一个浮点数时间** **>>>time.mktime(time.gmtime())** **1535139564.0**

time库详解之时间获取

函数	描述
asctime([t])	**返回一个字符串表示的时间，如果提供参数t，则把t变成字符串时间** **>>>time.asctime()** **'Sat Jun 2 10:33:52 2018'**
ctime([s])	**返回一个字符串表示的时间，如果提供参数s，则把s变成字符串时间** **>>>time.ctime()** **'Sat Jun 2 10:33:52 2018'**

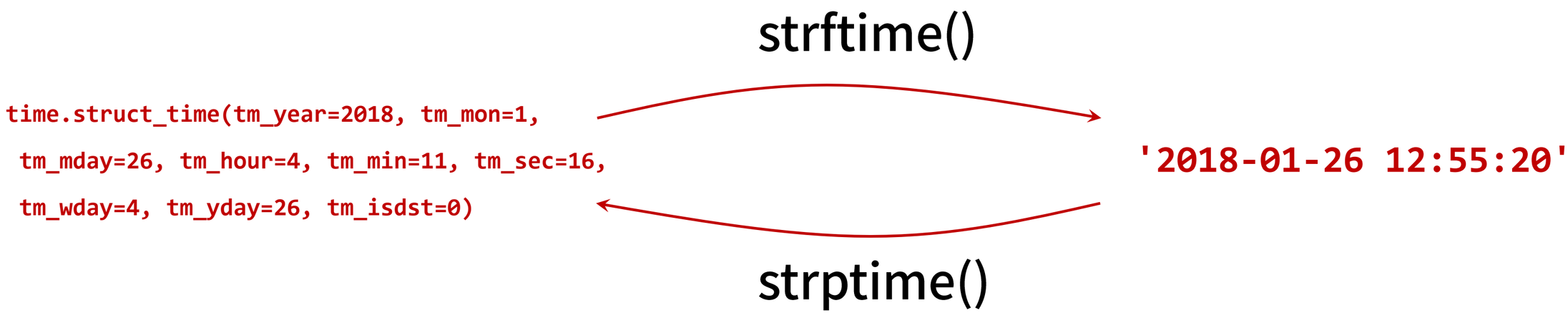
time库详解之时间获取

time库的时间获取

- 产生浮点数时间：time()、mktime()
- 产生struct_time时间：gmtime() 、localtime()
- 产生字符串时间： ctime()、 asctime()

time库详解之时间格式化

time库的时间格式化



格式化：类似字符串格式化，通过模板定制输出效果或指定输入模式

time库详解之时间格式化

函数	描述
strftime(tpl, t)	tpl是格式化模板字符串，定义输出效果，t是struct_time变量 >>>t = time.gmtime() >>>time.strftime("%Y-%m-%d %H:%M:%S",t) '2018-01-26 12:55:20'

time库详解之时间格式化

理解时间的格式化控制符

格式化	说明	范围和实例
%Y	年份	0000~9999，例: 2018
%m	月份	01~12，例: 10
%B	月份名全称	January~December
%b	月份名缩写	Jan~Dec，例: Apr
%d	日期	01~31，例: 25
%A	星期	Monday~Sunday
%a	星期缩写	Mon~Sun

格式化	说明	范围和实例
%H	小时 24h制	00~23
%I	小时 12h制	01~12
%p	上/下午	AM, PM
%M	分钟	00~59
%S	秒	00~59
%Z	时区	字符串，如: '中国标准时间'
%%	%字符	%

time库详解之时间格式化

函数	描述
strptime(str, tpl)	str是字符串形式时间，tpl是格式化模板字符串 >>>ts = '2018-01-26 12:55:20' >>>time.strptime(ts, "%Y-%m-%d %H:%M:%S") time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26, tm_hour=4, tm_min=11, tm_sec=16, tm_wday=4, tm_yday=26, tm_isdst=0)

time库详解之时间格式化

```
>>> s = time.strftime("%Y-%m-%b-%d-%a-%H-%p-%M-%S", time.localtime())
>>> s
'2018-08-Aug-25-Sat-12-PM-21-22'
>>> time.strptime(s, "%Y-%m-%b-%d-%a-%H-%p-%M-%S")
time.struct_time(tm_year=2018, tm_mon=8, tm_mday=25, tm_hour=12, tm_min=21,
                 tm_sec=22, tm_wday=5, tm_yday=237, tm_isdst=-1)
```

time库详解之程序计时

time库的五个计时时钟

时钟名称	对应函数	说明
'clock'	time.clock()	计时时钟
'monotonic'	time.monotonic()	单调计时时钟，不可更改，时间差有意义
'perf_counter'	time.perf_counter()	精确计时时钟，含起始和终止的所有时间
'process_time'	time.process_time()	进程计时时钟，不含进程sleep()时间
'time'	time.time()	系统计时时钟，精度不高，用于time()函数

time库详解之程序计时

函数	描述
clock()	返回一个计时时间，调用之差是间隔时间，不同平台精度不同 >>>time.clock() 6099.184732391898
monotonic()	返回一个计时时间，两次调用之差是间隔时间，单位为秒 >>>time.monotonic() 233020.406

time库详解之程序计时

函数	描述
perf_counter()	返回一个精确计时时间，包含全部时间，单位为秒 >>>time.perf_counter() 6336.718092311298
process_time()	返回一个进程计时时间，不包含sleep()时间，单位为秒 >>>time.monotonic() 0.3125

time库详解之程序计时

函数	描述
sleep(s)	将线程挂起s秒，s可以是浮点数 >>>time.sleep(5.5)

time库详解之辅助函数

函数	描述
get_clock_info(name)	返回以下五种计时时钟的属性值： 'clock'、'monotonic'、'perf_counter'、'process_time'、'time’ >>>time.get_clock_infor('perf_counter') namespace(adjustable=False, implementation= 'QueryPerformanceCounter()', monotonic=True, resolution=2.844443635358255e-07)

time库小结

time库函数的分类 (14个)

- 时间获取：time()、 gmtime()、 ctime()、 asctime() 、localtime()、mktime()
- 时间格式化：strftime()、strptime()
- 程序计时： clock()、monotonic()、perf_counter()、process_time()、sleep()
- 辅助函数：get_clock_info()

Python常用标准库解析(上)

random库的使用

Python123

random库介绍

random库是Python中生成随机数的标准库

- 库名：random，生成随机数
- 提供产生单个或系列随机数的功能
- 提供随机组合等相关操作的功能

https://docs.python.org/3.7/library/random.html

random库介绍

random库函数的分类 (21个)

- 基本随机函数：seed()、 random()、getstate()、setstate()
- 扩展随机函数：randint()、getrandbits()、randrange()、choice()、shuffle()、sample()
- 分布随机函数：uniform()、triangular()、betavariate()、expovariate()、gammavariate()、gauss()、lognormvariate()、normalvariate()、 vonmisesvariate()、paretovariate()、 weibullvariate()

random库详解之基本随机函数

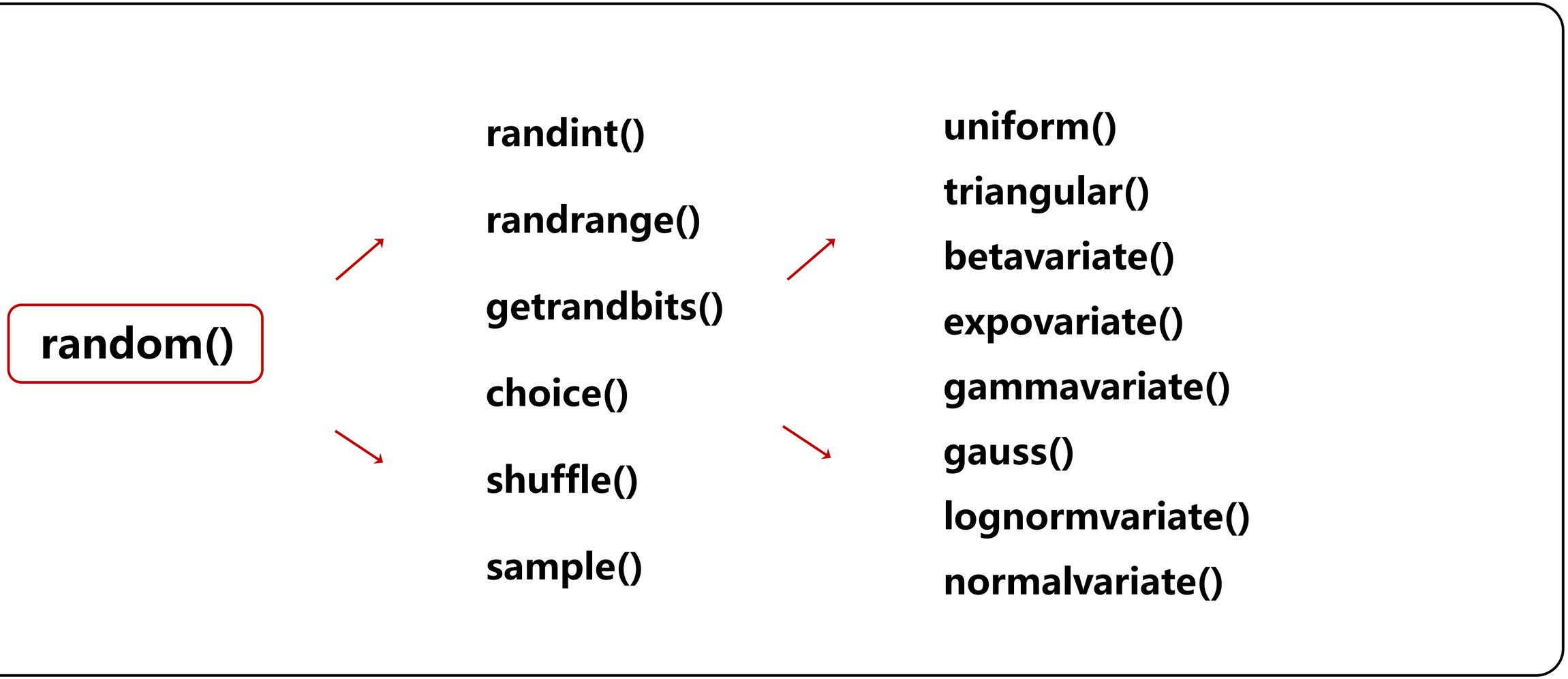
random库详解之基本随机函数

函数	描述
seed(a=None)	**初始化给定的随机数种子，默认为当前系统时间** **>>>random.seed(10)**
random()	**生成一个[0.0, 1.0)之间的随机小数** **>>>random.random()** **0.500492727732152**

random库详解之基本随机函数

函数	描述
getstate()	返回随机数生成器内部状态，元组类型 >>>t = random.getstate() >>>type(t) <class 'tuple'>
setstate(state)	设置随机数生成器内部状态，该状态从getstate()函数获得 >>>random.setstate(t)

random库详解之扩展随机函数



random库详解之扩展随机函数

函数	描述
randint(a, b)	生成一个[a, b]之间的整数 >>>random.randint(10, 100) 64
randrange(m, n[, k])	生成一个[m, n)之间以k为步长的随机整数 >>>random.randrange(10, 100, 10) 80

random库详解之扩展随机函数

函数	描述
getrandbits(k)	生成一个k比特长的随机整数 >>>random.getrandbits(16) 37885
choice(seq)	从序列seq中随机选择一个元素 >>>random.choice([1,2,3,4,5,6,7,8,9]) 8

random库详解之扩展随机函数

函数	描述
shuffle(seq)	将序列seq中元素随机排列，原序列被修改 >>>ls = [1,2,3,4,5,6,7,8,9] >>>random.shuffle(ls) #ls被修改 [3, 5, 8, 9, 6, 1, 2, 7, 4]
sample(pop, k)	从序列或集合pop中随机选择k个元素，原序列或集合不变 >>>ls = [1,2,3,4,5,6,7,8,9] >>>random.sample(ls, 5) #sample不变 [5, 1, 2, 7, 3]

random库详解之分布随机函数

函数	描述
uniform(a, b)	生成一个[a, b]之间的随机小数，采用等概率分布 >>>random.randint(10, 100) 64
betavariate(alpha, beta)	生成一个[0, 1]之间的随机小数 采用beta分布
triangular(low, high, mode)	生成一个[low, high]之间的随机小数 采用三角分布（也叫辛普森分布）

random库详解之分布随机函数

函数	描述
expovariate(lambda)	生成一个(0, ∞)之间的随机整数，采用指数分布
gammavariate(alpha, beta)	生成一个随机小数，采用gamma分布
gauss(mu, sigma)	生成一个随机小数，采用高斯分布（也叫正态分布）
lognormvariate(mu, sigma)	生成一个随机小数，采用对数正态分布
normalvariate(mu, sigma)	生成一个随机小数，采用正态分布

random库详解之分布随机函数

函数	描述
vonmisesvariate(mu,kappa)	**生成一个随机小数，采用冯米塞斯分布**
paretovariate(alpha)	**生成一个随机小数，采用帕累托分布**
weibullvariate(alpha,beta)	**生成一个随机小数，采用韦伯分布**

random库小结

random库函数的分类 (21个)

- 基本随机函数：seed()、 random()、getstate()、setstate()
- 扩展随机函数：randint()、getrandbits()、randrange()、choice()、shuffle()、sample()
- 分布随机函数：uniform()、triangular()、betavariate()、expovariate()、gammavariate()、gauss()、lognormvariate()、normalvariate()、 vonmisesvariate()、paretovariate()、 weibullvariate()

Python常用标准库解析(上)

re库的使用

Python123

re库介绍

re库是Python中处理正则表达式的标准库

- 库名：re
- 正则表达式是用来简洁表达一组字符串的表达式

'PN'
'PYN'
'PYTN '

'PYTHN'
'PYTHON'

⟺

正则表达式：

P(Y|YT|YTH|YTHO)?N

https://docs.python.org/3.7/library/re.html

正则表达式语法

正则表达式语法由字符和操作符构成

正则表达式语法

操作符	说明	实例
.	表示任何单个字符	
[ ]	字符集，对单个字符给出取值范围	[abc]表示a、b、c，[a-z]表示a到z单个字符
[^]	非字符集，对单个字符给出排除范围	[^abc]表示非a或b或c的单个字符
*	前一个字符0次或无限次扩展	abc* 表示 ab、abc、abcc、abccc等
+	前一个字符1次或无限次扩展	abc+ 表示 abc、abcc、abccc等
?	前一个字符0次或1次扩展	abc? 表示 ab、abc
|	左右表达式任意一个	abc|def 表示 abc、def

正则表达式语法

操作符	说明	实例
{m}	扩展前一个字符m次	ab{2}c表示abbc
{m,n}	扩展前一个字符m至n次（含n）	ab{1,2}c表示abc、abbc
^	匹配字符串开头	^abc表示abc且在一个字符串的开头
$	匹配字符串结尾	abc$表示abc且在一个字符串的结尾
( )	分组标记，内部只能使用 | 操作符	(abc)表示abc，(abc|def)表示abc、def
\d	数字，等价于[0-9]	
\w	单词字符，等价于[A-Za-z0-9_]	

正则表达式语法

经典正则表达式实例

^[A-Za-z]+$	由26个字母组成的字符串
^[A-Za-z0-9]+$	由26个字母和数字组成的字符串
^-?\d+$	整数形式的字符串
^[0-9]*[1-9][0-9]*$	正整数形式的字符串
[1-9]\d{5}	中国境内邮政编码，6位
[\u4e00-\u9fa5]	匹配中文字符
\d{3}-\d{8}|\d{4}-\d{7}	国内电话号码，010-68913536

re库介绍

原生字符串类型：raw string，不包含对转义符解释的字符串

- 原生字符串： `r'text'`
- 实例：

```
r'[1-9]\d{5}'                    对比    '[1-9]\\d{5}'
r'\d{3}-\d{8}|\d{4}-\d{7}'               '\\d{3}-\\d{8}|\\d{4}-\\d{7}'
```

re库介绍

re库的主要函数 (7个)

- 基础函数：compile()
- 功能函数：search()、match()、findall()、split()、finditer()、sub()

re库详解

re.search(pattern, string, flags=0)

在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象

- pattern：正则表达式的字符串或原生字符串表示
- string ：待匹配字符串
- flags ：正则表达式使用时的控制标记

re库详解

re.search(pattern, string, flags=0)

- flags ：正则表达式使用时的控制标记

常用标记	说明
re.I re.IGNORECASE	忽略正则表达式的大小写，[A-Z]能够匹配小写字符
re.M re.MULTILINE	正则表达式中的^操作符能够将给定字符串的每行当作匹配的开始
re.S re.DOTALL	正则表达式中的.操作符能够匹配所有字符，默认匹配除换行外的所有字符

re库详解

re.search(pattern, string, flags=0)

```
>>> import re
>>> match = re.search(r'[1-9]\d{5}', 'BIT 100081')
>>> if match:
        print(match.group(0))


100081
>>>
```

re库详解

re.match(pattern, string, flags=0)

在一个字符串的开始位置起匹配正则表达式，返回match对象

- pattern：正则表达式的字符串或原生字符串表示
- string ：待匹配字符串
- flags ：正则表达式使用时的控制标记

re库详解


```
>>> import re
>>> match = re.match(r'[1-9]\d{5}', 'BIT 100081')
>>> if match:
        match.group(0)


>>> match.group(0)
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    match.group(0)
AttributeError: 'NoneType' object has no attribute 'group'
>>> match = re.match(r'[1-9]\d{5}', '100081 BIT')
>>> if match:
        match.group(0)


'100081'
>>>
```

re库详解

re.findall(pattern, string, flags=0)

搜索字符串，以列表类型返回全部能匹配的子串

- pattern：正则表达式的字符串或原生字符串表示
- string ：待匹配字符串
- flags ：正则表达式使用时的控制标记

re库详解

re.findall(pattern, string, flags=0)

```
>>> import re
>>> ls = re.findall(r'[1-9]\d{5}', 'BIT100081 TSU100084')
>>> ls
['100081', '100084']
>>>
```

re库详解

re.split(pattern, string, maxsplit=0 , flags=0)

将一个字符串按照正则表达式匹配结果进行分割，返回列表类型

- pattern : 正则表达式的字符串或原生字符串表示
- string ：待匹配字符串
- maxsplit: 最大分割数，剩余部分作为最后一个元素输出
- flags ：正则表达式使用时的控制标记

re库详解

re.split(pattern, string, maxsplit=0 , flags=0)

```
>>> import re
>>> re.split(r'[1-9]\d{5}', 'BIT100081 TSU100084')
['BIT', ' TSU', '']
>>> re.split(r'[1-9]\d{5}', 'BIT100081 TSU100084', maxsplit=1)
['BIT', ' TSU100084']
>>>
```

re库详解

re.finditer(pattern, string, flags=0)

搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象

- pattern：正则表达式的字符串或原生字符串表示
- string：待匹配字符串
- flags：正则表达式使用时的控制标记

re库详解

re.finditer(pattern, string, flags=0)

```
>>> import re
>>> for m in re.finditer(r'[1-9]\d{5}', 'BIT100081 TSU100084'):
        if m:
            print(m.group(0))


100081
100084
>>>
```

re库详解

re.sub(pattern, repl, string, count=0, flags=0)

在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

- pattern ：正则表达式的字符串或原生字符串表示
- repl ：替换匹配字符串的字符串
- string ：待匹配字符串
- count ：匹配的最大替换次数
- flags ：正则表达式使用时的控制标记

re库详解

re.sub(pattern, repl, string, count=0, flags=0)

```
>>> import re
>>> re.sub(r'[1-9]\d{5}', ':zipcode', 'BIT100081 TSU100084')
'BIT:zipcode TSU:zipcode'
>>>
```

re库介绍

函数	说明
re.search()	在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象
re.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
re.findall()	搜索字符串，以列表类型返回全部能匹配的子串
re.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
re.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
re.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

re库详解

re库的另一种等价用法

```
>>> rst = re.search(r'[1-9]\d{5}', 'BIT 100081')
```

函数式用法：一次性操作

```
>>> pat = re.compile(r'[1-9]\d{5}')
>>> rst = pat.search('BIT 100081')
```

面向对象用法：编译后的多次操作

re库详解

regex = re.compile(pattern, flags=0)

将正则表达式的字符串形式编译成正则表达式对象

- pattern：正则表达式的字符串或原生字符串表示
- flags ：正则表达式使用时的控制标记

```
>>> regex = re.compile(r'[1-9]\d{5}')
```

re库介绍：另一种等价用法

函数	说明
regex.search()	在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象
regex.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
regex.findall()	搜索字符串，以列表类型返回全部能匹配的子串
regex.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
regex.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
regex.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

re库的match对象

match对象是一次匹配的结果，包括匹配的各种信息

```
>>> match = re.search(r'[1-9]\d{5}', 'BIT 100081')
>>> if match:
        print(match.group(0))
>>> type(match)
<class '_sre.SRE_Match'>
```

re库的match对象

属性	说明
.string	待匹配的文本
.re	匹配时使用的patter对象（正则表达式）
.pos	正则表达式搜索文本的开始位置
.endpos	正则表达式搜索文本的结束位置

re库的match对象

方法	说明
.group(0)	获得匹配后的字符串
.start()	匹配字符串在原始字符串的开始位置
.end()	匹配字符串在原始字符串的结束位置
.span()	返回(.start(), .end())

re库的match对象

```
>>> import re
>>> m = re.search(r'[1-9]\d{5}', "BIT100081 TSU100084")
>>> m.string
'BIT100081 TSU100084'
>>> m.re
re.compile('[1-9]\\d{5}')
>>> m.pos
0
>>> m.endpos
19
>>> m.group(0)
'100081'
>>> m.start()
3
>>> m.end()
9
>>> m.span()
(3, 9)
```

re库小结

re库的主要函数 (7个)

- 基础函数：compile()
- 功能函数：search()、match()、findall()、split()、finditer()、sub()
- compile()的用法、match对象

Python常用标准库解析(上)

单元小结

单元开篇

(1) time库的使用

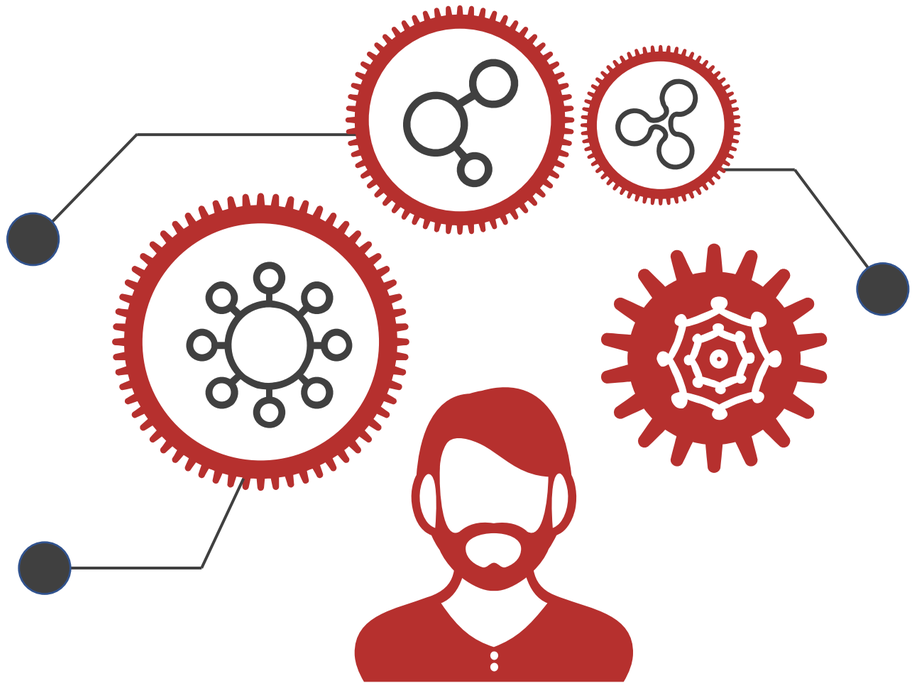
时间获取、格式化、程序计时等

14个函数

(2) random库的使用

基本、扩展和分布随机函数等

21个函数



(3) re(正则表达式)库的使用

功能函数、compile()等

7个函数

Python常用标准库解析(上)

time库小结

time库函数的分类 (14个)

- 时间获取：time()、 gmtime()、 ctime()、 asctime() 、localtime()、mktime()
- 时间格式化：strftime()、strptime()
- 程序计时： clock()、monotonic()、perf_counter()、process_time()、sleep()
- 辅助函数：get_clock_info()

random库小结

random库函数的分类 (21个)

- 基本随机函数：seed()、 random()、getstate()、setstate()
- 扩展随机函数：randint()、getrandbits()、randrange()、choice()、shuffle()、sample()
- 分布随机函数：uniform()、triangular()、betavariate()、expovariate()、gammavariate()、gauss()、lognormvariate()、normalvariate()、 vonmisesvariate()、paretovariate()、 weibullvariate()

re库小结

re库的主要函数 (7个)

- 基础函数：compile()
- 功能函数：search()、match()、findall()、split()、finditer()、sub()
- compile()的用法、match对象


Python▸123
Thank you