第 3 章

CHAPTER 3

标准库简介

Python 的标准库是一个包含许多模块和函数的集合，这些模块和函数都是 Python 语言本身的一部分。Python 标准库中的一些常用模块和函数有以下几个。

(1) os 模块：提供了与操作系统交互的函数，例如文件和文件夹操作。

(2) time 模块：提供了时间相关的函数，例如获取当前时间、延迟等。

(3) 数字和数学模块：提供了与数字和数学相关的函数和数据类型。Math 模块包含各种数学函数。decimal 模块支持使用任意精度算术的十进制数的精确表示。random 模块用于生成随机数。

(4) pymysql：一个用于连接和操作 MySQL 数据库的模块。

(5) threading 和 multiprocessing 模块：用于多线程和多进程编程。

(6) queue 模块：一个线程安全的队列类，用于多线程之间的通信。

(7) datetime 模块：处理日期和时间的类。

下面我们就常用的数学与日期和时间模块做一下说明，其他内容可参看官方说明文档。

3.1 数字与数学模块

本章介绍的模块提供了与数字和数学相关的函数和数据类型。numbers 模块定义了数字类型的抽象层次结构。Math 模块提供了对 C 标准定义的数学函数的访问。decimal 模块支持使用任意精度算术的十进制数的精确表示。

3.1.1 数学 Math 模块的使用

Math 模块包括数学函数、三角函数、角度转换函数和常量等。在使用 Math 模块之前，必须首先使用 import 语句导入 Math 模块，代码如下：

```
import math
```

Math 模块常用的函数如下。

(1) math.ceil(x)对 x 的值向上取整,即大于或等于 x 的最小的整数,代码如下:

```
import math                                    #导入 Math 模块,下同,不再重复
n1 = math.ceil(5.6)                            #向上取整,输出:6
n2 = math.ceil(5.1)                            #向上取整,输出:6
```

(2) math.fabs(x)返回 x 的绝对值,代码如下:

```
n1 = math.fabs( - 66)
print("n1 = ",n1)                              #输出:n1 = 66.0
```

(3) math.floor(x)返回 x 的向下取整,小于或等于 x 的最大整数,代码如下:

```
n1 = math.floor( - 6.3)
n2 = math.floor(9.6)
print("n1 = ",n1)                              #输出:n1 =  - 7
print("n2 = ",n2)                              #输出:n2 = 9
```

(4) math.trunc(x)返回去除小数部分的 x,只留下整数部分,代码如下:

```
n1 = math.trunc(12.45678)
print("n1 = ",n1)                              #输出:n1 = 12
```

(5) math.sqrt(x)返回 x 的平方根,代码如下:

```
x = math.sqrt(100)
print("100 的平方根为"x)

#输出的结果如下
100 的平方根为 10.0
```

(6) math.radians(x)将角度 x 从度数转换为弧度。

(7) math.cos(x)返回 x 弧度的余弦值,代码如下:

```
#计算 60°的余弦值
#注意:math.cos 接收的是弧度,所以需要将 60°转换为弧度
radians = math.radians(60)
print(radians)                                 #输出:1.0471975511965976
cos_value = math.cos(radians)
print(cos_value)                               #输出:0.5000000000000001
```

(8) math.sin(x)返回 x 弧度的正弦值,代码如下:

```
#计算 45°的正弦值
radians = math.radians(45)
sin_value = math.sin(radians)
print(sin_value)                               #输出:0.7071067811865476
```

(9) math.tan(x)返回 x 弧度的正切值,代码如下:

```
#计算 45°的正切值
radians = math.radians(45)
tan_value = math.tan(radians)
print(tan_value)                               #输出:0.9999999999999999
```

(10) math.atan(x)返回以弧度为单位的 x 的反正切值。用法与正切函数类似。

(11) math.pi 是一个预定义的浮点数，表示数学常数 π(圆周率)，代码如下：

```
radius = 5
area = math.pi * radius ** 2
print(f"圆的面积是 {area:8.2f}")

#输出的结果如下
圆的面积是     78.54
```


4min

3.1.2 精度 decimal 模块

计算机在对浮点类型的数字进行数学运算时，结果会有误差，代码如下：

```
x = 2.356
y = 3.60
z = x + y
print(z)
#输出的结果如下
5.9559999999999995
```

Decimal 数字的表示是完全精确的。在 Python 中，使用 Decimal 类可以对两个浮点数进行精确求和，以避免浮点数运算中的精度问题。Decimal 类位于 decimal 模块中，因此在使用之前需要先导入该模块。

下面演示如何使用 Decimal 类对两个浮点数进行求和，代码如下：

```
#第 3 章 3.1 使用 Decimal 类对两个浮点数进行求和
from decimal import Decimal

num1 = Decimal('2.356')
num2 = Decimal('3.60')
result = num1 + num2
print(result)

#输出的结果如下
5.956
```

在上面的代码中，from decimal import Decimal 表示从 decimal 模块中导入 Decimal 类。首先使用 Decimal 类创建了两个浮点数 num1 和 num2，然后使用加法运算符“+”对它们进行求和，并将结果存储在变量 result 中。最后，使用 print()函数打印结果。

使用 Decimal 类可以确保浮点数运算的精确性，特别是在涉及金钱计算等需要高精度的场景中非常有用。


6min

3.1.3 随机数 random 模块

在 Python 语言中，random 模块实现了各种分布的伪随机数生成器。这个模块提供了

多种生成随机数的功能。

(1) 生成一个[0,1)的随机浮点数(包含 0,但不包含 1),代码如下:

```
import random                                    #导入 random 模块

random_number = random.random()
print(random_number)                             #随机生成一个[0, 1)的随机浮点数

#输出的结果如下
0.24206096892194762
```

(2) 生成一个指定范围内的随机整数,代码如下:

```
import random

random_number = random.randint(1, 10)            #生成 1 到 10(包括 10)的随机整数
print(random_number) #输出:2
```

(3) 从一个序列中随机选择一个元素,代码如下:

```
import random

my_list = [1, 2, 3, 4, 5]                        #定义一组数列
random_choice = random.choice(my_list)           #从一组数列中随机选择一个数
print(random_choice)                             #输出:3
```

3.2　日期和时间模块


1min

Python 语言中的 datetime 模块用于日期和时间的获取、表达和转换,此模块提供了以下类。

(1) datetime:表示一个具体的日期和时间。

(2) date:表示一个具体的日期(年、月、日)。

(3) time:表示一天中的时间。

(4) timedelta:表示两个日期或时间之间的差异。

根据日期和时间,可以得到与日期和时间相关的信息,例如一个月内的留言、一天内的留言或设定八小时后的工作等。

3.2.1　日期时间 datetime 类


8min

通常 datetime 类包含对日期和时间的处理,以下是 datetime 模块中常用的方法。

(1) datetime.now():返回当前日期和时间。

(2) datetime.strptime(date_string, format):根据指定的格式字符串将字符串解析为日期。

（3）datetime.combine(date，time)：将给定的日期和时间组合成一个新的 datetime 对象。

（4）datetime.date()：返回 datetime 对象的日期部分。

（5）datetime.time()：返回 datetime 对象的时间部分。

（6）datetime.timestamp()：返回表示日期和时间的浮点数时间戳。

（7）datetime.strftime(format)：将 datetime 对象格式化为字符串。

（8）datetime.isoformat()：将 datetime 对象格式化为 ISO 8601 格式的字符串。

（9）datetime.replace(year，month，day，hour，minute，second，microsecond)：返回一个新的 datetime 对象，使用指定的年、月、日、时、分、秒和微秒部分替换当前对象的相应部分。

（10）datetime.isoweekday()：返回 ISO 周几的整数，其中 1 表示星期一，7 表示星期日。

以下是一些使用 datetime 模块处理时间和日期的例子。

【示例 3-1】 获取当前日期和时间，代码如下：

```
from datetime import datetime      #导入 datetime 模块的 datetime 类

now = datetime.now()               #获取当前日期和时间
print("当前时间:", now)

#输出的结果如下
当前时间: 2023-12-27 20:53:10.254654
```

由于要使用 datetime 模块，所以必须在程序的第 1 行导入相应的模块和类。在语句 from datetime import datetime 中，第 1 个 datetime 表示模块名，第 2 个 datetime 表示 datetime 类。

【示例 3-2】 将指定数字组成日期类型，代码如下：

```
from datetime import date        #导入 datetime 模块的 date 类

#创建一个特定日期的实例
d = date(2023, 7, 4) #2023 年 7 月 4 日
print("特定日期:", d)

#输出的结果如下
特定日期: 2023-07-04
```

【示例 3-3】 日期格式化，代码如下：

```
from datetime import datetime

#获取当前时间
now = datetime.now()

#格式化输出
formatted_time = now.strftime("%Y-%m-%d %H:%M:%S")
print("当前时间为", formatted_time)
```

使用 strftime()方法将当前时间格式化为字符串。strftime()方法的参数是一个格式字符串，它指定了日期的输出格式。在这个示例中，使用%Y-%m-%d %H:%M:%S 作为格式字符串，它将日期和时间格式化为"年-月-日 时:分:秒"的格式。

也可以使用不同的格式字符串来控制日期的输出格式。例如，如果想要将日期和时间格式化为"月/日/年 时:分:秒"的格式，则可以将格式字符串改为"%m/%d/%Y %H:%M:%S"。

【示例 3-4】 获取 5 天内的所有日期，代码如下：


```
#第 3 章 3.2 获取 5 天内的所有日期
from datetime import datetime,timedelta

#获取 5 天内的所有日期
current_date = datetime.now()
dates_in_month = []
future_date = current_date + timedelta(days = 5)
while current_date < future_date:
    dates_in_month.append(current_date)
    current_date += timedelta(days = 1)

print("5 天内的所有日期:")
for date in dates_in_month:
    print(date)

#输出的结果如下
5 天内的所有日期:
2024 - 01 - 30 21:11:52.499425
2024 - 01 - 31 21:11:52.499425
2024 - 02 - 01 21:11:52.499425
2024 - 02 - 02 21:11:52.499425
2024 - 02 - 03 21:11:52.499425
```

3.2.2　时间间隔 timedelta 类

其中 timedelta 是 datetime 模块中的一个类，用于表示时间间隔，其参数包括以下几个。

(1) days：表示天数，可以为正数(表示天数)或负数(表示往前推的天数，下同)。

(2) seconds：表示秒数。

(3) microseconds：表示微秒数。

(4) milliseconds：表示毫秒数。

(5) minutes：表示分钟数。

(6) hours：表示小时数。

(7) weeks：表示周数。

这些参数都是可选的，可以根据需要选择使用。例如，可以使用 timedelta(days=10)

表示 10 天的时间间隔。

【示例 3-5】 timedelta 用于表示时间间隔，即两个日期或时间之间的差异。可以得到当前时间，并推断当前时间之后 8h45min 后的日期，代码如下：

```
#第 3 章 3.3 计算两个日期之间的间隔
from datetime import datetime, timedelta
#获取当前日期
now = datetime.now()
print("当前日期:", now)

#创建 8h45min 后的日期
future_date = now + timedelta(hours = 8, minutes = 45)
print("8h45min 后的日期:", future_date)

#计算两个日期之间的差异
difference = future_date - now
print("日期差:", difference)

#输出的结果如下
当前日期: 2024 - 01 - 12 14:36:05.641164
8h45min 后的日期: 2024 - 01 - 12 23:21:05.641164
日期差: 8:45:00
```

3.2.3 日期 date 类

在 Python 的 date 模块中，主要提供了 date 类，用于表示日期(年、月、日)。以下是一些常用的方法。

(1) date(year, month, day)：创建一个表示指定日期的 date 对象。

(2) year、month、day：获取日期的年、月、日部分。

(3) replace(year, month, day)：返回一个新的日期对象，使用指定的年、月、日部分替换当前日期对象中的相应部分。

(4) weekday()：返回日期对应的星期几，其中 0 表示星期一，6 表示星期日。

(5) isoweekday()：返回日期对应的 ISO 周几，其中 1 表示星期一，7 表示星期日。

(6) isocalendar()：返回包含年份、ISO 周数和一周中的日期的元组。

(7) strftime(format)：根据指定的格式字符串将日期格式化为字符串。

【示例 3-6】 演示 date 模块的用法，代码如下：

```
#第 3 章 3.4 date 模块的用法
from datetime import date,datetime

#创建一个 date 对象
d = date(2023, 7, 5)
#获取当前日期和时间
#d = datetime.now()
```

```
#获取 date 对象中的年、月、日部分
year = d.year
month = d.month
day = d.day
print(year, month, day) #输出:2023 7 5

#使用 replace 方法替换年、月、日部分
new_d = d.replace(year = 2024, month = 8, day = 15)
print(new_d) #输出:2024 - 08 - 15

#获取星期几和 ISO 星期几
weekday = d.weekday()
isoweekday = d.isoweekday()
print(weekday, isoweekday) #输出:4 5(根据实际日期而定)

#将日期格式化为字符串
formatted_date = d.strftime("%Y - %m - %d")
print(formatted_date) #输出:2023 - 07 - 05
```

3.2.4　时间 time 类

在 Python 的 time 类中，提供了一系列与时间相关的功能，以下是一些常用的方法。

(1) time()：返回当前的时间戳，表示从 1970 年 1 月 1 日 00:00:00 起经过的秒数。

(2) sleep(seconds)：暂停程序执行指定的秒数，参数 seconds 代表秒数。

(3) localtime()：将时间戳转换为本地时间的 time.struct_time 元组。

(4) gmtime()：将时间戳转换为 UTC 时间的 time.struct_time 元组。

(5) asctime(time_tuple)：将 time.struct_time 元组格式化为字符串表示。

(6) ctime(seconds)：将时间戳转换为字符串表示。

(7) strptime(string, format)：根据指定的格式字符串解析字符串中的时间。

这些方法可以帮助我们在 Python 中进行时间戳计算、格式化时间、处理本地时间和 UTC 时间等操作。有时需要使用和时间相关的组件，例如提醒用户在某个时间节点做什么事情。

【示例 3-7】 时间的输出，代码如下：

```
import time

#将时间值格式化输出
current_time = time.strftime("%H:%M", time.gmtime())
print(f"现在的时间为{current_time}。")

#输出的结果如下
现在的时间为 06:41。
```

【示例 3-8】 时间的比较和定时执行某项工作，代码如下：

```
#第 3 章 3.5 定时执行某项工作
import time

last_time = time.time()
current_time = time.strftime("%H:%M", time.localtime())
print(f"当前的时间为 {current_time},1 分钟后有提示信息")

time.sleep(60)                              #其中 sleep()函数指延迟执行程序给定的秒数

new_cur_time = time.time()
time_passed = int((new_cur_time - last_time) / 60)    #计算时间差
print(f" {time_passed} 分钟过去了,请按时完成任务")

#输出的结果如下
当前的时间为 14:51,1 分钟后有提示信息
1 分钟过去了,请按时完成任务
```

3.3 实训作业

（1）编写一个程序，输入一个正整数，使用数学模块中的函数计算其阶乘。

（2）已知列表 numbers = [5, 2, 8, 1, 9]，使用数学模块中的函数求出最大值和最小值。

（3）编写一个程序，输入一个日期，计算该日期的前一天和后一天，使用时间和日期模块中的函数完成计算。

（4）编写一个程序，输入两个日期，计算两个日期之间的天数差，使用时间和日期模块中的函数完成计算。

（5）编写一个程序，输入一个日期，判断该日期是星期几，使用时间和日期模块中的函数完成判断。