关于本节中要掌握的内容

- 函数与引用
- 字典
- toml

函数与引用

在之前的学习中，其实已经接触到了很多函数，例如：print、enumrate、open等。
为什么要有函数呢？就是为了简化我们的编程，还记得之前的作业嘛？
将下面的list变成字符串，倒数第二个字符串前面加上--

```
l = ["a","b","c","d"]
temp = l.copy()
if len(temp) >= 2:
    temp[-2] = "--" + temp[-2]
    s = "".join(temp)
    print(s)
else:
    print("list len < 2")
l = ['apples', 'bananas', 'cats']
temp = l.copy()
if len(temp) >= 2:
    temp[-2] = "--" + temp[-2]
    s = "".join(temp)
    print(s)
else:
    print("list len < 2")
```

是不是发现，每次做这个处理都得写好几遍重复的代码对吧。再来看看下面的：

```
def fun(l):
    temp = l.copy()
    if len(temp) >= 2:
        temp[-2] = "--" + temp[-2]
        s = "".join(temp)
        print(s)
    else:
        print("list len < 2")
fun(["a","b","c","d"])
fun(['apples', 'bananas', 'cats'])
```

- 这就是函数的作用，为别人/自己将一些需要重复的代码，写到某个代码块中，并给它一个名称，后续只需要使用它就可以了。

函数通常由以下几部分组成：

- def关键字
- 函数名称
- 函数参数
- 代码块
- 返回值

考虑以下的功能：
List中包含了多个数字，输出其中能被2、3、5整除的数字。

```
def func(l):
    for it in l:
        if (it % 2 == 0) or (it % 3 == 0) or (it % 5 == 0):
            print(it)
l1 = [2,4,5,3,232,4324,12,1231,0,5,53]
func(l1)
```

尝试一下修改该函数，返回List中能被2、3、5整除的数字。

```
def func(l):
    return [it for it in l if (it % 2 == 0) or (it % 3 == 0) or (it % 5 == 0)]
print(func([2,4,5,3,232,4324,12,1231,0,5,53]))
```

这里使用了第三节课中List的for if语法，应该还记得吧。

在使用的函数的过程中，除了需要注意参数和返回值外，还有一个东西格外重要：**变量作用域**

看一个示例：

```
a = 1
l1 = [2,3,4]
def func():
    a = 2
    l1.append(5)
func()
print(a,l1)
```

- 猜一下最后会输出什么？

- 函数中对对象的赋值，相当于构造了一个新的对象
- 函数中可以访问到在执行前定义的对象
- 函数中定义的对象外部是访问不了的
- 如果函数内需要赋值全局对象，则用global声明一下

```
a = 1
b = [1,2,3]
def fun1():
    a = 2
    b = [4,5,6]
    c = [1,2,3]
    print(a,b,c)
def fun2():
    global a
    a = 3
    b.append(10)
    print(a,b,c)
fun1()
fun2()
print(a,b,c)
```

编写一个python程序，将x以内的所有质数输出到文件中，每个质数一行，文件名和x由用户输入参数给定，要求编写三个函数:

- print_all_Prime(x)
  该函数用于输出所有x内的质数输出到文件中
- is_prime(x)
  该函数用于判断x是否为质数，返回True则是，返回False不是
- log(x)
  日志的输出格式为："[time]: x is a prime number"

你可能需要的一些库：

- argparse
- datetime

为什么要使用命令行参数，考虑以下的场景：
1.你需要统计学生的考试信息，并以可视化/表格的形式输出。这时你编写了一个python脚本，里面实现了关于：本次考试x班级成绩表查询，本次考试x年级成绩表查询，不同班级对比数据查询……。这里包含了一堆查询，你想想通过参数让别人查询想要的内容，还是去看你的源码，修改你的源码来的好？
2.假设你完成了一个算法的自动化python脚本，其他人的python脚本中需要使用你的python脚本，如果你有参数的话，其他人可以直接在脚本中使用os.system()，一行代码就可以使用你的脚本了，但是如果没有参数，它每次都需要修改你的源码。

- 一句话总结就是：方便别人/自己更好的使用python脚本

从我们的需求中分析，它需要两个参数，x和log，那我们最理想简单的使用方式就是这样：

```
# test.py是我们python脚本的名称
# -x 100，意味着我们的x参数为100
# -l test.log，意味着我们的日志文件参数为test.log
# 我们希望这个脚本输出100以内的质数到test.log这个文件中
test.py -x 100 -l test.log
```

```
import argparse
parser = argparse.ArgumentParser(description='get all prime.')
parser.add_argument("-l", "--log", help="log file path",required=True)
parser.add_argument("-x","--value", help="x value",
                    type=int,required=True)
args = parser.parse_args()
print(args.log,args.value)
```

- 获取当前格式化时间的可能代码

```
from datetime import datetime
print(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
```

```
import argparse
from datetime import datetime
f = ""
def ini_log_file(path):
    global f
    f = open(path,"w")
    f.close()
    f = open(path,"a")
def get_now():
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
def log(x):
    global f
    f.write(f"[{get_now()}]: {x} is a prime number\n")
def is_prime(num):
    if num > 1:
        for i in range(2, int(num**0.5) + 1):
            if (num % i) == 0:
                return False
        return True
    else:
        return False
def print_all_Prime(x):
    for i in range(1,x+1):
        if is_prime(i):
            log(i)
parser = argparse.ArgumentParser(description='get all prime.')
parser.add_argument("-l", "--log", help="log file path",required=True)
parser.add_argument("-x","--value", help="x value",
                    type=int,required=True)
args = parser.parse_args()
ini_log_file(args.log)
print_all_Prime(args.value)
f.close()
```

字典

在Python中，字典（Dictionary）是一种无序、可变的数据结构，用于存储键-值对。字典以大括号 {} 表示，每个键值对之间使用冒号 : 分隔。以下是是一个使用示例：

```
book = {"name": "how to use git?","pages": 100,"type":"A"}
# 访问字典中的值
print("Name:", book["name"])
print("Pages:", book["pages"])
print("type:", book["type"])
# 修改字典中的值
book["pages"] = 125
print("Pages:", book["pages"])
# 添加新的键值对
book["author"] = "niko"
print("Author:", book["author"])
# 删除键值对
del book["type"]
print(book)
```

- 猜猜如何遍历字典呢?

聪明的你一定猜到了，使用迭代器遍历：

```
for key in book:
    print(key,book[key])
```

1.那么，可以这样去遍历book吗？为什么？

```
for i in range(len(book)):
    print(book[i])
```

2.下面的情况会输出什么？

```
book = {"name": "how to use git?","pages": 100,"type":"A"}
print(book['price'])
```

- 这里一定一定要注意的是，字典不是有序的结构，我们不能使用for range来遍历它
- 访问不存在的键，程序会直接崩溃的

1.下面的情况会输出什么?

```
book = {0:0,1:1,2:2}
for i in range(len(book)):
    print(book[i])
```

2.下面的情况会输出什么?

```
book1 = {"name": "how to use git?","pages": 100,"type":"A"}
book2 = {,"pages": 100,"type":"A","name": "how to use git?"}
print(book1 == book2)
```

- 任何可以被hash的对象，都可以作为键

> 只要该对象可以通过hash函数，变为int类型，都可以作为键。f(object) = int

- 字典是无序的结构，两个字典是否相同只看它们里面的键值是否相同

完成以下题目：
1.定义一个List和一个dict，输出在Dict中的List元素。
2.定义一个List和一个dict，输出在List中的Dist元素。
3.定义5名学生的信息（id，姓名，年龄），然后：
3.1找出年龄在范围内的学生信息
3.2找出id为x的学生信息
3.3找出姓名为x的学生信息

```
book = {0:0,1:1,2:2}
l = [0,2]
for it in l:
    if it in book:
        print(it)
for it in book:
    if it in l:
        print(it)
```

```
students = [{"id":123,"name":"a","age":13},{"id":124,"name":"b","age":14},
{"id":125,"name":"c","age":15},{"id":126,"name":"d","age":16},
{"id":127,"name":"e","age":17}]
for it in students:
    if it["age"] > 13 and it["age"] < 15:
        print(it)
for it in students:
    if it["id"] == 126:
        print(it)
for it in students:
    if it["name"] == "e":
        print(it)
```

有时为了仅获取字典的所有键/值/键值对的List，是不是需要这样：

```
keys = []
values = []
items = []
for it in book:
    keys.append(it)
    values.append(book[it])
    items.append((it,book[it]))
print(keys,values,items)
```

python已经帮我们实现好了，只需要这样既可：

```
print(book.keys(),book.values(),book.items())
for key,value in book.items():
    print(key,value)
```

toml

toml是一种配置文件的数据格式，因为有时我们可能会需要给python脚本中提供大量的参数，使用命令行参数不是太方便，使用toml可以很轻松的保存到文件中。

```
# style
[[server]]
ip = "127.0.0.1"
port = 8080
enabled = true

[[server]]
ip = "192.168.0.1"
port = 8080
enabled = flase

[database]
host = "localhost"
username = "admin"
password = "password"
```

我们将上述的toml保存到文件中，使用python读取它的内容

```
import toml
f = "test.toml"
data = toml.load(f)
print(data)
```

将配置写入到文件

```
data = {'name': 'test toml', 'number': 666, 'server': [{'ip': '127.0.0.1', 'port': 8080, 'enabled': True},
{'ip': '192.168.0.1', 'port': 8080, 'enabled': False}],
'database': {'host': 'localhost', 'username': 'admin', 'password': 'password'}}
toml.dump(data, open("config.toml", "w"))
```

- 自己编写一个toml，包含一组学生的信息配置，将它写入到toml中，再读取到python中，并输出。