

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | Python数据分析 | | | | |
| 实验编号： | 实验三 | | | | |
| 实验名称： | Python函数的定义与函数式编程 | | | | |
| 实验人员： | 学号 | 18111207248 | | | |
| 姓名 | 吴钰 | | | |
| 班级 | 18级计算机科学与技术（创新班） | | | |
| 实验日期： | 2020.10.12 | | | | |
| 实验室： | 2060402 | | | | |
|  |  | | | | |
| 实验评价： |  | | | | |
| 实验成绩： | |  | 评价日期： |  |
|  | 指导教师： | |  | | |

# 实验目的

## （1） 掌握Python函数的定义与使用。 （2） 掌握Python函数式编程。

## （3） 掌握类的定义与使用

# 二、实验要求

## 完成实验后请填写实验报告并上交。要求在实验报告中必须记录实验中遇到的问题及其问题解决方案。

# 三、实验内容

题目1：编写程序。先生成一个密码（允许含有数字与字母），再要求用户从键盘输入密码，用户只有三次输入错误密码的机会。

代码：

import random

import string

def function():

print('生成密码：（8位）')

ran\_str = ''.join(random.sample(string.ascii\_letters + string.digits, 8))

print(ran\_str)

flag=0

for i in range(3):

s=input('请输入密码(8位)')

if ran\_str==s:

flag=1

break

else:

print('密码错误！')

continue

if flag==0:

print('错误超过三次！禁止输入！正确密码是：',ran\_str)

else:

print('密码正确！')

function()

运行结果：



题目2：编写程序。接收一个正整数作为参数，返回对其进行因数分解后的结果列表。例如，接收数50，则输出为[2,5,5]

代码：

list1=[]

def function(n, list1):

for i in range(2, n):

while i!= n:

if n%i==0:

list1.append(i)

n=n/i

else:

break

list1.append(int(n))

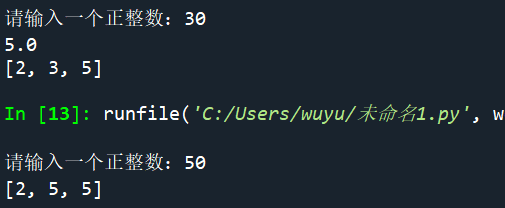
return list1

n=int(input('请输入一个正整数：'))

function(n,list1)

print(list1)

运行结果：



遇到的问题：一开始我没有写list1.append(int(n))这条代码，导致最后总是少一个结果

解决方案：经过检查，我发现要最后把n加进列表，n最后是被修改了的，而不是最开始输入的那个结果，相当于成为了最后剩下的那个因子

题目3：编写程序。将所有能被17整除的三位数输出。

代码：

def function():

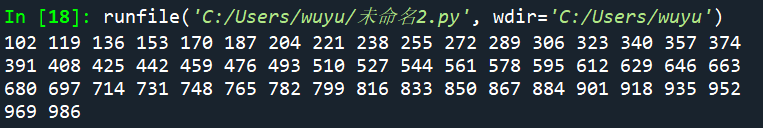
for i in range(100,1000):

if i%17==0:

print(i,end=' ')

function()

运行截图：



题目4：编写程序。了解标准库itertools中的cycle()函数的功能与用法，以及生成器函数的内容，然后编写生成器函数模拟该函数。

答：

生成器对象是通过使用yield关键字定义的函数对象，因此，生成器也是一个函数。生成器用于生成一个值的序列，以便在迭代器中使用。

yield 语句是生成器中的关键语句，生成器在实例化时并不会被执行，而是等待调用其\_\_next\_\_()方法才开始运行。并且当程序运行完yield语句后就会保持当前状态且停止运行，等待下一次遍历时才恢复运行

cycle()函数是一个无限迭代函数

代码：

def function(it):

while True:

for i in it:

yield i

function=function(['a','b','c','d','e'])

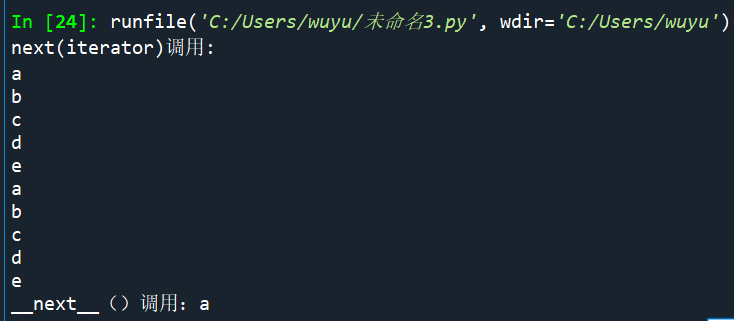
print("next(iterator)调用:")

for i in range(10):

print(next(function))

print("\_\_next\_\_（）调用",function.\_\_next\_\_(),sep="：")

运行结果：



题目5：编写程序。创建PayCalculator类，拥有pay\_rate属性，以每小时人民币数量为单位。该类拥有compute\_pay(hours)方法，计算给定工作时间的报酬，并返回。创建类的对象，调用成员方法。

代码：

class PayCalculator:

def \_\_init\_\_(self,pay\_rate):

self.pay\_rate=pay\_rate

def compute\_pay(self,hours):

self.hours=hours

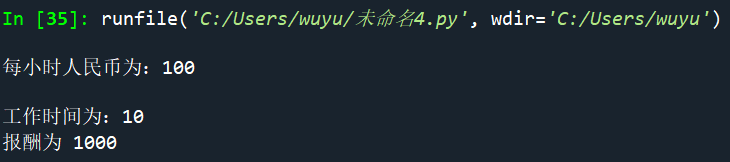
return self.pay\_rate\*hours

mm=int(input('每小时人民币为：'))

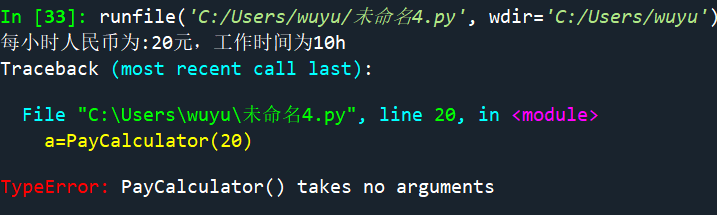
nn=int(input('工作时间为：'))

a=PayCalculator(mm)

print('报酬为',a.compute\_pay(nn))

运行截图：  


遇到的问题：一开始在写代码时总是会出现这个报错：



解决方案：

通过查找资料我知道\_\_init\_\_左右是各有两个下划线而不是一个下划线！

题目6：编写程序。创建SchoolKid类，初始化对象的姓名、年龄。也有访问每个属性的方法和修改属性的方法。然后创建ExaggeratingKid类，继承SchoolKid类，在子类中覆盖访问年龄的方法，并将实际年龄加2。创建这两个类的对象，调用成员方法。

代码：

class SchoolKid:

def \_\_init\_\_(self,Name,Age):

self.name=Name

self.age=Age

def visitName(self):

print(self.name)

def visitAge(self):

print(self.age)

def alterName(self):

self.name=input('请输入修改后的姓名：')

def alterAge(self):

self.age=int(input('请输入修改后的年龄：'))

class ExaggeratingKid(SchoolKid):

def visitAge(self):

print(self.age+2)

a=SchoolKid('wy',3)

a.visitName()

a.visitAge()

b=ExaggeratingKid('wxy',5)

b.visitName()

b.visitAge()

a.alterAge()

a.alterName()

a.visitName()

a.visitAge()

b.alterAge()

b.alterName()

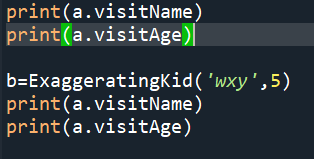
b.visitName()

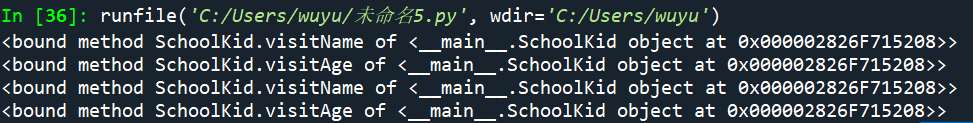
b.visitAge()

运行截图：



遇到的问题：





解决方法：直接通过对象调用方法就可以了，不需要套print来引用

a.visitName()

a.visitAge()