python 知识点总结

W.J.Z

1 基础知识

1.1 python 必备知识

1. 12 个数学操作符：+、-、*、/、%、//、**、+=、-=、*=、/=、%=。
2. 2 个基本函数：print()、input()。
3. 12 个逻辑符号:True、False、==、! =、<、>、<=、>=、and、or、not。

1.2 控制语句

if 控制语句：

```
age = 10
if age < 10:
        print('low')
elif age > 10:
        print('high')
else:
        print('right')
```

while 控制语句：

```
age = 1
while age < 5:
        age = age +1
print(age)
```

for 控制语句

```
for i in range(1,10,2):
        print(i)
```

1.3 函数定义

```
def function(num)
while num < 5:
        print(num)
        num = num + 1
return num
new = function(1)
print(new)
```

1.4 数据结构

1.4.1 列表

列表常用方法：

```
parm = [1,2,3,4]
parm[3]  #查看指定下标的数据
parm[-1]   #倒着数列表
parm[1:3]  #对数据进行切片
len(parm)#len()函数求列表的长度
parm * 2       #重复2遍列表
parm + [5,6]  #列表连接
del parm[3]#删除指定坐标的数据
parm.index(2) #获取数据2所在的下标
```

```
parm.append(4)#末尾添加数据4
parm.insert(2,2)#在下标2插入数据2
parm.remove(2)#删除第一个2
parm.sort(reverse=True)#逆序排列列表
```

1.5 元组

元组数据内容不能被修改、添加、删除，其他函数方法与列表相同。将数据转换为列表和元组使用 list() 和 tuple() 函数。

```
list((1,2,3,4))
tuple([1,2,3,4])
```

1.6 字典

```
parm = {'name':'xiaoli','age':'12'}
parm.keys()  #键
parm.values() #值
parm.items()  #所有数据
#返回name对应的值.没有返回0
parm.get('name',0)
#设置默认sex
parm.setdefault('sex','female')
```

1.7 字符串

方法	说明
upper()	所有字母变为大写
lower()	所有字母变为小写
isupper()	字符串是否为大写
islower()	字符串是否为小写

upper() 和 lower() 函数会创建新的副本，而不是直接修改的原有数据, 如果需要更改使用 pparm = parm.lower() 方式。

方法	说明
isalpha()	字符串只包含字母，非空
isalnum()	只包含字母和数字，非空
isdecimal()	只包含数字字符，非空
isspace()	只包含空格、制表符、换行，非空
istitle()	大写字母开头，后面都是小写字母
startswith()	字符串开头的单词判断
endswith()	字符串结尾单词判断
join()	连接字符串列表
split()	切割字符串
rjust()	参数一指定从右对齐长度，参数二指定填充字符
ljust()	参数一指定从左对齐长度，参数二指定填充字符
cjust()	参数一指定从左右对齐长度，参数二指定填充字符
strip()	删除字符左右的空白字符
rstrip()	删除字符右的空白字符
lstrip()	删除字符左的空白字符

2 正则表达式

2.1 正则表达式速记技巧

. [ ] ∧ $ 是所有语言都支持的正则表达式，正则难理解因为其有一个等价概念，将等价恢复于原始写法就简单的多。

1、等价

? ,*,+,\d,\w 都是等价字符。

? 等价于匹配长度 {0,1}

* 等价于匹配长度 {0,}
+ 等价于匹配长度 {1,}
\d 等价于 $[0-9]$
\D 等价于 $[^0-9]$
\w 等价于 $[A-Za-z0-9]$
\W 等价于 $[^A-Za-z0-9]$

2、常用运算符与表达式
^ 开始
() 域段
[ ] 包含
[^] 不包含
{m,n} 匹配长度
. 任何单个字符
| 或
\ 转义
$ 结尾
[A-Z] 26 个大写字母
[a-z] 26 个小写字母
[0-9] 0 到 9 数字
[A-Za-z0-9]
, 分割
[0,3] 包含 0 或 3 数字

3、语法与释义
基础语法 "^([]{})([]{})$"
技巧：?,*,+,\d,\w 这些都是简写的, 完全可以用 [] 和 {} 代替

2.2 python 正则表达式

1. 用 import re 导入正则表达式模块

2. 用 re.complie() 函数创建一个 Regex 对象

3. 向 Regex 对象的 search() 方法传入向查找的字符串

4. 调用 Match 对象的 group() 方法返回实际匹配的字符串

括号分组
```
regex=re.compile(r'(\d{3})-(\d{3}-\d{4})')
mo=regex.search('my num 415-555-4344')
mo.group() #415-555-4344
mo.group(1)#415
mo.group(2)#555-4344
```

匹配多个分组 , 和c语言的或作用一样
```
regex = re.compile(r'b |a')
mo = regex.search('b')
mo.group() #b
mo = regex.search('a')
mo.group() #a
```

贪心与非贪心
```
regex = re.compile(r'(ha){3,5}')
mo=regex.search('hahahahaha')
mo.group() #hahahahaha
regex = re.compile(r'(ha){3,5}?')
mo = regex.search('hahahahaha')
mo.group() #hahaha
```

findall()方法
search()方法返回一个Match对象，包含被查找字符串中的第一次匹配的文本，而findall()方法将返回一组字符串。

sub()替换字符串

sub(arg1,arg2);参数一为替换字符串
参数二为匹配内容

3 读取文件

3.1 路径

```
import os
#根据系统获取正确的路径
os.path.join('usr','bin')
#获取当前工作路径
os.path.getcwd()
#改变当前工作路径
os.path.chdir('c:\\')
#创建新文件夹
os.makedirs('c:\\newdir')
#获取绝对路径
os.path.abspath('.')
#是否为绝对路径
os.path.isabs(path)
#返回从start到path的相对路径
os.path.relpath(path,start)
#获取路径名
os.path.dirname(path)
#获取文件名
os.path.basename(path)
#获取文件大小
os.path.getsize(path)
#获取文件夹下文件名列表
os.listdir(path)
#检查路径是否存在
os.path.exists(path)
#检查文件是否存在
os.path.isfile(path)
#检查是否为文件夹
os.path.isdir(path)
```

3.2 读取文件

```
#打开文件,r读模式 w写模式 a追加模式
file = open(path, 'r')
#读取文件
content = file.read();
content1 = file.readlines();
#写文件
file  = open(path,'w')
file.write(text+'\n')
file.close()
```

4 组织文件

4.1 文件和文件夹操作

```
import shutile,os
#将spam.txt文件复制到dev文件夹下
#并重命名pam.txt
shutil.copy('c:\\spam.txt','c:\\dev\\pam.txt'
#复制整个文件夹
shutil.copytree('c:\\bacon','c:\\backup')
#将pam文件移动到eggs文件夹重命名spam.txt
shutil.move('c:\\pam.txt','c:\\eggs\\spam.txt
#删除文件夹和文件
shutil.rmtree(path)
#使用send2trash安全删除
import send2trash
send2trash.send2trash(file)
```

```
#遍历目录树
os.walk();
#读取zip文件
improt zipfile,os
zip=zipfole.ZipFile(file)
#获取压缩包李文件名列表
zip.namelist()
#加压zip文件
zip.extractall()
#指定解压文件
zip.extract(file)
#创建zip文件
zip = zipfile.ZipFile(file,'w')
zip.write('filename',compress='')
zip.close()
```

5 调试

5.1 抛出异常

```
def demo(num):
if num != 1:
        raise Exception("num")

for i in range(0,3,1):
try:
        demo(i)
except Exception as err:
        print(str(err))
```

5.2 日志模块

```
import logging
logging.basicConfig(filename='log.txt',
level=logging,DEBUG,format=
'%(asctime)s-%(levelname)s-%(message)s')
logging.disable(logging.DEBUG)
```

python 的日志级别

级别	日志函数	描述
DEBUG	loggging.debug()	最低级别
INFO	logging.info()	记录程序中一般时间
WARNING	logging.warning()	表示可能的问题
ERROR	logging.error()	记录错误
CRITICAL	logging.critical()	表示致命错误

6 web 爬虫

6.1 webbrowser

```
import requests
res = requests.get('http://www.baidu.com/1/tx
#检查下载是否成功
res.raise_for_status()
#将下载内容写入文件
file = open('2.txt','wb')
for duff in res.iter_content(10000)
        playfile.write(chunk)
file.close()
import requests,bs4
res = requests.get('www://d.com')
res.raise_for_status()
elem = bs4.BautifulSoup(res.text)
#使用select()方法寻找元素
elems = elem.select('span')
#获取元素的属性值
elems.get('id')
```

6.2 selemium

selenium 的 WebDriver 方法，用于寻找元素。

```
#使用CSS类的name元素
browser.find_element_by_class_name(name)
browser.find_elements_by_class_name(name)
#匹配CSS selector元素
browser.find_element_by_css_selector(selector)
browser.find_elements_by_css_selector(selector)
#匹配id属性值得元素
browser.find_element_by_id(id)
browser.find_elements_by_id(id)
#完全匹配提供的text的<a>元素
browser.find_element_by_link_text(text)
browser.find_element_by_link_text(text)
#包含提供的text的<a>元素
browser.find_element_by_partial_link_text(text)
browser.find_elements_by_partial_link_text(text)
#匹配name属性值得元素
browser.find_element_by_name(name)
browser.find_elements_by_name(name)
#匹配标签name的元素
browser.find_element_by_tag_name(name)
browser.find_elements_by_tag_name(name)
```

```
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('www.baidu.com')
elem = browser.find_element_by_link_text('a')
elem.click()
```

7 处理 execl 文件

```
import openpyxl
wb=openpyl.load_workbook('ee.xlxs')
wb.get_sheet_names()
sheet=wb.get_sheet_by_name('sheet1')
sheet=wb.get_active_sheet()
sheet['A1']
sheet.cell(row=1,column=2)
sheet['A1':'C3']
sheet.columns[1]
sheet.rows[1]
wb.creat_sheet(index=2,title='a')
#sheet为worksheet对象，不是字符串
wb.remove_sheet(sheet)
```

8 处理 CSV 和 JSON 文件

处理 csv 文件：

```
import csv
file = open('file.csv')
reader = csv.reader(file)
data = list(reader)
newfile = open('out.csv','w',
newline='')
output = csv.writer(newfile)
output.writerow(data[1])
output.close()
```

处理 JSON 文件：

```
improt json
data= json.loads(file.text)
```

9 时间函数

1. datetime 对象包含一些整型值，保存在 year、month、day、hour、minute、second 等属性中。
2. timedelta 对象表示一段时间
3. time.time() 返回当前时刻 Unix 纪元时间戳
4. time.sleep(seconds) 让程序暂停 seconds 参数指定的秒数
5. datetime.datetime(year,month,day,hour,minute,second) 返回指定时刻的 datetime 对象
6. datetime.datetime.now() 函数返回当前时刻的 datetime 对象

```
import threading
#APP为自定义函数,args为传参
threading.Tread(target=APP, args=[])
```

10 发送短信

```
from twilio.rest
improt TwilioRestClient
accountSID='xxx'
authToken='xxxx'
twilioCli=TwilioRestClient(accountSID,
authToken)
myTwilioNumber='xxx'
myCellPhone='xxx'
message=twilioCli.messages.create(body=''
,from_=myTwilioNumber,to=myCellPhone)
```

11 用 gui 自动化控制键盘和鼠标

11.1 控制鼠标

```
import pyautogui
#执行动作之前都会等待一秒
pyautogui.PAUSE=1
#获取屏幕的大小
pyautogui.size()
#移动鼠标
pyautogui.moveTo(x,y,duration=0.25)
#获取当前坐标信息
pyautogui.position()
#点击鼠标
pyautogui.click(x,y)
#拖动鼠标
pyautogui.dragTo(x,y,duration=1)
#从当前坐标拖动鼠标
pyautogui.dragRel(x,y,duration=1)
#滚动鼠标，200个单位
pyautogui.scroll(200)
#获取屏幕快照
im=pyautogui.screenshot()
#匹配坐标
pyautogui.pixelMatchesColor(50,200,(r,n,g))
im.getpixel((200,200))
#图像识别，obj为目标图像
pyautogui.locateOnScreen('obj.png')
#查找所有匹配的目标
list(pyautogui.locateAllOnScreen('obj.png'))
#获取目标的坐标
pyautogui.center((1,2,2,3))
#点击
pyautogui.click((1,2))
```

11.2 控制键盘

```
#发送字符串
pyautogui.typewrite('hello')
#按键
pyautogui.press('1')
#热键组合
pyautogui.hotkey('ctrl','c')
```