基于python的web开发

笔记地址

---

1. 知识点和实际开发项目结合
2. 课后留作业,下节课看完成情况
3. 每位同学所掌握的知识层次不齐,需考虑到
4. 课后接收同学发骢,下节课进行调整,
   包括课程内容,课程难易程度,课程时长等

准备篇(2021年12月27日)

一台能上网的笔记本(工具兼查资料)

linux的安装(ubuntu,centos,kali)

python的安装(python3)

编辑器(vi,vscode,sublime等)

笔记(md,txt,word)

linux的安装

直接装在计算机上,虚拟机,子系统(WSL)

直接ubuntu安装

虚拟机安装

子系统安装

注意的点:

- 更新软件源
- 各linux系统命令会有所差异

  apt,yum

python的安装

centos安装python3

更新pip源

也可以直接用命令

```
pip config set global.index--url https://pypi.tuna.tsinghua.edu.cn/simple
```

编辑器

Vim

Vscode

Sublime

Pycharm

notepad++

...

笔记

typora

markdown

```python
def test():
    print('这是一段python代码')
```

Linux
MacOS
Windows

2021年12月27日作业

作业

在自己的电脑上,用虚拟机装一个centos7.9或者ubuntu18.04

更新软件源为阿里源或者清华源

启动之后,安装python3(ubuntu18.04默认装了3.6)

1. 编写 一个 xxx.py 文件，写入Python 的语句

```
# file: hello.py
print("hello world!")
```

2. 用 python3 的解释执行器执行 xxx.py
    - 在Windows 用终端命令

```
C:> python hello.py
```

    - 在Linux 下用终端命令

```
$ python3 hello.py
```

预习

linux有哪些命令

2021年12月27日作业情况

1.root用户操作

2.没有虚拟环境

linux(2022年1月4日-1)

常用命令

command [-options] [parameter]

开始

ls,cd,pwd,whoami

建用户

如：zbxu，请使用root用户操作（root用户请询问银行管理员）

```
useradd -m zbxu
```

设置密码

```
passwd zbxu
```

sudo权限

```
usermod -a -G wheel zbxu
```

增删改

mkdir,rm,mv,cp,touch,rm,vi

查

find,grep,cat,more,less,tail,head,ps ux,↑↓

管道

ps ux | grep sh | grep -v grep

权限管理

sudo 管理员权限执行

chmod [o]+x xxx.txt (给other增加执行权限)

chmod 777 xxx.txt(所有人权限全开)

例如：
`777` ===> `u=rwx,g=rwx,o=rwx`
`755` ===> `u=rwx,g=rx,o=rx`
`644` ===> `u=rw,g=r,o=r`

解压缩

tar -zcvf xxx.tar.gz 要打包的文件

tar -zxvf xxx.tar.gz

zip xxx.zip 要打包的文件

unzip xxx.zip

重定向

ls -la >> ~/xxx.log

符号链接

ln -s hello.py hello

关机重启

shutdown -r now 立即重启

shutdown now 立即关机

shutdown +10 10分钟后关机

shutdown -c 取消关机计划

帮助

xxx --help

man xxx

shell编程

启动停止脚本

```
#!/bin/sh
filetime=`date +%Y%m%d`
case "$@" in
    start)
        nohup python hello.py >> ~/logs/hello.log &
        ;;
    stop)
        ps ux|grep hello|grep -v grep|awk '{print $2}'|xargs kill -9
        ;;
    *)
        echo "unknown argument args(start|stop)"
        exit 1
        ;;
esac
```

vim 粘贴

```
set paste
```

命令分解

重复执行脚本

```
import time
print('hello', time.time())
```

```
#!/bin/sh
while true
    do
        python time_print.py
        echo "wait 5s"
        sleep 5
    done
```

虚拟环境(2022年1月4日-2)

目的

相互隔离的 Python 环境

指的说明的是包管理器除了pip还有其他,遇到了不要感觉陌生

相关概念链接

python 包管理器

软件包源中的软件包数量巨大，版本多样，所以需要借助于软件源管理工具，例如 pip、conda、Pipenv、Poetry 等

- pip 是最常用的包管理工具，通过 `pip install <packagename>` 命令格式来安装软件包，使用的是 pypi 软件包源
- conda 多用作科学计算领域的包管理工具，功能丰富且强大，使用的软件包源是 Anaconda repository 和 Anaconda Cloud，conda 不仅支持 Python 软件包，还可以安装 C、C++ 、R 以及其他语言的二定制软件包。除了软件包管理外，还能提供相互隔离的软件环境。
- Pipenv 是 Kenneth Reitz 在2017年1月发布的Python依赖管理工具，现在由PyPA维护。Pipenv 会自动帮你管理虚拟环境和依赖文件，并且提供了一系列命令和选项来帮助你实现各种依赖和环境管理相关的操作
- Poetry 和 Pipenv 类似，是一个 Python 虚拟环境和依赖管理工具，另外它还提供了包管理功能，比如打包和发布。你可以把它看做是 Pipenv 和 Flit 这些工具的超集。它可以让你用 Poetry 来同时管理 Python 库和 Python 程序

很多包管理工具不仅提供了基本的包管理功能，还提供了虚拟环境构建，程序管理的等功能

创建

virtualenv

在 python3.3 之前，只能通过 virtualenv 创建虚拟环境，首先需要安装 virtualenv

```
pip install virtualenv
```

指定python解析器

```
virtualenv --python=python3 myenv
```

virtualenv: error: unrecognized arguments: --no-site-packages

venv

Python3.3 之后，可以用模块 venv 代替 virtualenv 工具，好处是不用单独安装，3.3 及之后的版本，都可以通过安装好的 Python 来创建虚拟环境:

```
python -m venv myvenv
```

可以在当前目录创建一个名为 myvenv 的虚拟环境

因为 venv 是依附于一个 Python 解析器创建的，所以不需要指定 Python 解释器版本

激活

```
$ source myvenv/bin/activate
```

激活后，可以在命令行中看到虚拟环境标记

判断当前虚拟环境的python版本:

```
which python
```

退出

```
deactivate
```

部署

之所以在开发时选择虚拟环境，除了避免库之间的冲突，还有重要的原因是方便部署，因为虚拟环境是独立的，仅包含了项目相关的依赖库，所以部署的效率更高，风险更小

一般部署流程是:

1. 开发完成后，使用 `pip freeze > requirements.txt` 命令将项目的库依赖导出，作为代码的一部分
2. 将代码上传到服务器
3. 在服务器上创建一个虚拟环境
4. 激活虚拟环境，执行 `pip install -r requirements.txt`，安装项目依赖

ssh(2022年1月4日-3)

口令

```
ssh user@host
```

需要输入密码

公钥

创建

```
ssh-keygen
```

一路回车即可

在~/.ssh中有生成的密钥:

id_rsa

以及公钥:

id_rsa.pub

将公钥的内容拷到对方服务器中的authorized_keys中,若原先已有内容,往后追加

如自己新建的authorized_keys,则需要配置文件权限

文件权限

加完记得重启ssh服务

查看服务状态

```
systemctl status sshd
```

重启服务

```
systemctl restart sshd
```

启动服务

```
systemctl start sshd
```

停止服务

```
systemctl stop sshd
```

使用

```
ssh user@host
```

不用输入密码

2022年1月4日作业

作业1

1. 在自己创建的用户的家目录创建三个目录,分别是scripts,logs,projects
2. 在scripts中写一个sh脚本(print_time.sh),
3. 启动print_time.sh脚本,每隔30s输出当前时间,格式如下:

   2022年1月4日 10:43:22

   并且重定向到~/logs/homework.log中(不需要输出到前台)

作业2

1. 在~/projects/中创建一个python3的虚拟环境py3
2. 进入虚拟环境
3. 在环境中使用pip安装flask
4. 启动一个web服务,浏览器打开

   服务器ip:8888/自己的名字

   能够看到hello 自己的名字
5. 退出虚拟环境

作业3

1. 使用公钥ssh连上自己的虚拟机,启动flaskweb服务
2. 主机(非虚拟机)的浏览器打开网页显示

   hello 自己的名字

预习

python 的基础语法

python中包的使用

python基础(2022年1月10日-1)

定义

是一个免费、开源、跨平台、动态、面向对象的编程语言。

Python程序的执行方式

交互式

在命令行输入指令，回车即可得到结果。

文件式

将指令编写到.py文件，可以重复运行程序。

数据基本运算

注释

给人看的，通常是对代码的描述信息。

1. 单行注释：以#号开头。
2. 多行注释：三引号开头，三引号结尾。

函数

表示一个功能，函数定义者是提供功能的人，函数调用者是使用功能的人。

例如：

1. print(数据) 作用：将括号中的内容显示在控制台中
2. 变量 = input(“需要显示的内容”) 作用：将用户输入的内容赋值给变量

变量

1. 定义：关联一个对象的标识符。
2. 命名：必须是字母或下划线开头，后跟字母、数字、下划线。

   不能使用关键字(蓝色)，否则发生语法错误：SyntaxError: invalid syntax。

3. 建议命名：字母小写，多个单词以下划线隔开。

   class_name = “xxx”

4. 赋值：创建一个变量或改变一个变量关联的数据。
5. 语法：

变量名 = 数据

变量名1 = 变量名2 = 数据

变量名1, 变量名2, = 数据1, 数据2

del 语句

1. 语法:

del 变量名1, 变量名2

2. 作用:

用于删除变量,同时解除与对象的关联.如果可能则释放对象。

3. 自动化内存管理的引用计数:

每个对象记录被变量绑定(引用)的数量,当为0时被销毁。

核心数据类型

在python中变量没有类型，但关联的对象有类型。

整形int

1. 表示整数，包含正数、负数、0。

如： -5, 100, 0

2. 字面值:

十进制：每位用十种状态计数，逢十进一，写法是0~9。

二进制：每位用二种状态计数，逢二进一，写法是0b开头，后跟0或者1。

八进制：每位用八种状态计数，逢八进一，写法是0o开头，后跟0~7。

十六进制：每位用十六种状态计数，逢十六进一，

写法是0x开头，后跟0~9,A~F,a~f

浮点型float

1. 表示小数，包含正数、负数，0.0。
2. 字面值:

小数：1.0 2.5

科学计数法：e/E (正负号) 指数

1.23e-2 (等同于0.0123)

1.23456e5(等同于123456.0)

字符串str

是用来记录文本信息(文字信息)。

字面值：双引号

布尔bool

用来表示真和假的类型

True 表示真(条件满足或成立)，本质是1

False 表示假(条件不满足或不成立)，本质是0

数据类型转换

1. 转换为整形: int(数据)
2. 转换为浮点型:float(数据)
3. 转换为字符串:str(数据)
4. 转换为布尔:bool(数据)

   结果为False：bool(0) bool(0.0) bool(None)

5. 混合类型自动升级：

1 + 2.14 返回的结果是 3.14

1 + 3.0 返回结果是: 4.0

运算符

算术运算符

- 加法
- 减法
- 乘法

/ 除法：结果为浮点数

// 整除：除的结果去掉小数部分

% 求余

** 幂运算

优先级从高到低： ()

**

* / % //

+ -

增强运算符

y += x 相当于 y = y + x

y -= x 相当于 y = y - x

y *= x 相当于 y = y * x

y /= x 相当于 y = y / x

y //= x 相当于 y = y // x

y %= x 相当于 y = y % x

y **= x 相当于 y = y** x

比较运算符

< 小于

<= 小于等于

> 大于

>= 大于等于

== 等于

!= 不等于

返回布尔类型的值

比较运算的数学表示方式:0 <= x <= 100

逻辑运算符

与and

表示并且的关系，一假俱假。

示例:

```
True and True   # True
True and False  # False
False and True  # False
False and False  # False
```

或or

表示或者的关系，一真俱真

示例:

```
True or True    # True
True or False    # True
False or True    # True
False or False   # False
```

非 not

表示取反

例如:

```
not True  # 返回False
not False # 返回True
```

练习:根据命题写出代码

年龄大于25 并且 身高小于170

职位是高管 或者 年薪大于500000

短路运算

一但结果确定，后面的语句将不再执行。

```
In [3]: a = "python"
In [4]: print("hello,", a or "world")
hello, python
```

身份运算符

语法:

x is y

x is not y

作用:

is 用于判断两个对象是否是同一个对象,是时返回True,否则返回False。

is not 的作用与is相反

优先级

高到低:

算数运算符

比较运算符

增强运算符

身份运算符

逻辑运算符

python语句(2022年1月10日-2)

行

1. 物理行：程序员编写代码的行。
2. 逻辑行：python解释器需要执行的指令。
3. 建议一个逻辑行在一个物理行上。
4. 如果一个物理行中使用多个逻辑行，需要使用分号；隔开。
5. 如果逻辑行过长，可以使用隐式换行或显式换行。

隐式换行：所有括号的内容换行,称为隐式换行

括号包括: () [] {} 三种

显式换行：通过折行符（反斜杠)换行，必须放在一行的末尾，目的是告诉解释器,下一行也是本行的语句。

选择语句

If elif else 语句

1. 作用:

让程序根据条件选择性的执行语句。

2. 语法:

```
if 条件1:
  语句块1
elif 条件2:
  语句块2
else:
  语句块3
```

3. 说明:

elif 子句可以有0个或多个。

else 子句可以有0个或1个，且只能放在if语句的最后。

if 语句的真值表达式

```
if 100:
   print("真值")
```

等同于

```
if bool(100):
  print("真值")
```

条件表达式

语法：变量 = 结果1 if 条件 else 结果2

作用：根据条件(True/False) 来决定返回结果1还是结果2。

循环语句

while语句

1. 作用:

可以让一段代码满足条件，重复执行。

2. 语法:

while 条件:

满足条件执行的语句

else:

不满足条件执行的语句

3. 说明:

else子句可以省略。

在循环体内用break终止循环时,else子句不执行。

for 语句

1. 作用:

用来遍历可迭代对象的数据元素。

可迭代对象是指能依次获取数据元素的对象，例如：容器类型。

2. 语法:

for 变量列表 in 可迭代对象:

语句块1

else:

语句块2

3. 说明:

else子句可以省略。

在循环体内用break终止循环时,else子句不执行。

range 函数

1. 作用:

   用来创建一个生成一系列整数的可迭代对象(也叫整数序列生成器)。

2. 语法:

   range(开始点，结束点，间隔)

3. 说明:

函数返回的可迭代对象可以用for取出其中的元素

返回的数字不包含结束点

开始点默认为0

间隔默认值为1

跳转语句

break 语句

1. 跳出循环体，后面的代码不再执行。
2. 可以让while语句的else部分不执行。

continue 语句

跳过本次，继续下次循环。

python容器(2022年1月10日-3)

通用操作

数学运算符

1. +：用于拼接两个容器
2. +=：用原容器与右侧容器拼接,并重新绑定变量
3. *：重复生成容器元素
4. *=：用原容器生成重复元素, 并重新绑定变量
5. < <= > >= == !=：依次比较两个容器中元素,一但不同则返回比较结果。

成员运算符

1. 语法：

   数据 in 序列

数据 not in 序列

2. 作用：

如果在指定的序列中找到值，返回bool类型。

索引index

1. 作用：定位单个容器元素。
2. 语法：容器[整数]
3. 说明：

正向索引从0开始，第二个索引为1，最后一个为len(s)-1。

反向索引从-1开始,-1代表最后一个,-2代表倒数第二个,以此类推,第一个是-len(s)。

切片slice

1. 作用：

定位多个容器元素。

2. 语法：

容器[开始索引:结束索引:步长]

3. 说明：

结束索引不包含该位置元素

步长是切片每次获取完当前元素后移动的偏移量

开始、结束和步长都可以省略

内建函数

1. len(x) 返回序列的长度
2. max(x) 返回序列的最大值元素
3. min(x) 返回序列的最小值元素
4. sum(x) 返回序列中所有元素的和(元素必须是数值类型)

字符串 str

定义

由一系列字符组成的不可变序列容器，存储的是字符的编码值。

编码

1. 字节byte：计算机最小存储单位，等于8 位bit.
2. 字符：单个的数字，文字与符号。
3. 字符集(码表)：存储字符与二进制序列的对应关系。
4. 编码：将字符转换为对应的二进制序列的过程。
5. 解码：将二进制序列转换为对应的字符的过程。
6. 编码方式：

--ASCII编码：包含英文、数字等字符，每个字符1个字节。

--GBK编码：兼容ASCII编码，包含21003个中文；英文1个字节，汉字2个字节。

--Unicode字符集：国际统一编码，旧字符集每个字符2字节，新字符集4字节。

--UTF-8编码：Unicode的存储与传输方式，英文1字节，中文3字节。

相关函数

1. ord(字符串):返回该字符串的Unicode码。
2. chr(整数):返回该整数对应的字符串。

字面值

单引和双引号的区别

1. 单引号内的双引号不算结束符
2. 双引号内的单引号不算结束符

三引号作用

1. 换行会自动转换为换行符n
2. 三引号内可以包含单引号和双引号
3. 作为文档字符串

转义字符

1. 改变字符的原始含义。

   \n \t \r

2. 原始字符串：取消转义。

   a = r”C:newfiletest.py”

字符串格式化

1. 定义：

生成一定格式的字符串。

2. 语法：

字符串%(变量)

"我的名字是%s,年龄是%s" % (name, age)

3. 类型码：

%s 字符串 %d整数 %f 浮点数

列表 list

定义

由一系列变量组成的可变序列容器。

基础操作

1. 创建列表：

列表名 = []

列表名 = list(可迭代对象)

2. 添加元素：

列表名.append(元素)

列表.insert(索引，元素)

3. 定位元素：

列表名[索引] = 元素

变量 = 列表名[索引]

变量 = 列表名[切片] # 赋值给变量的是切片所创建的新列表

列表名[切片] = 容器 # 右侧必须是可迭代对象，左侧切片没有创建新列表。遍历列表：

正向：

for 变量名 in 列表名:

变量名就是元素

反向：

for 索引名 in range(len(列表名)-1,-1,-1):

列表名[索引名]就是元素

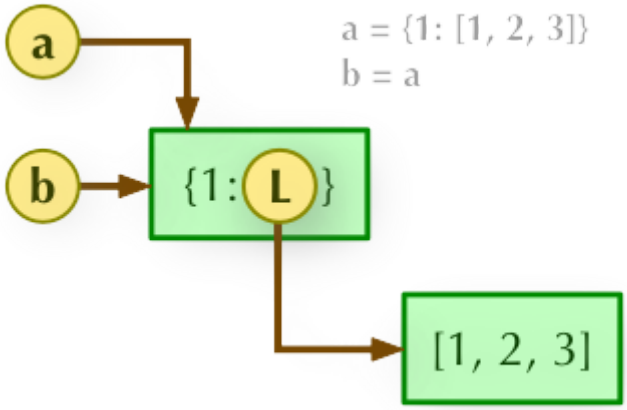
4. 删除元素：

列表名.remove(元素)

del 列表名[索引或切片]

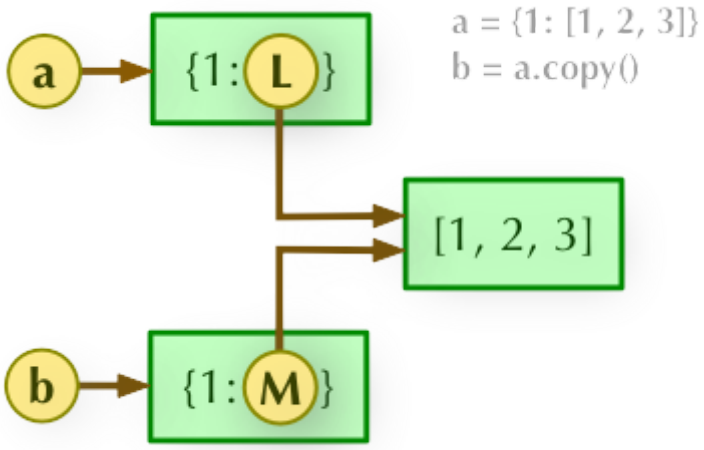
赋值,浅拷贝和深拷贝

1、**b = a:** 赋值引用，a 和 b 都指向同一个对象。

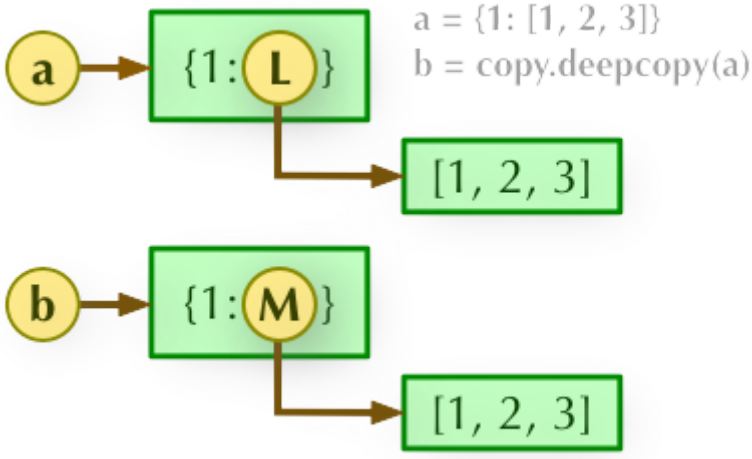


2、**b = a.copy():** 浅拷贝, a 和 b 是一个独立的对象，但他们的子对象还是指向统一对象（是引用）。

切片[:]操作也会有这个效果



3、**b = copy.deepcopy(a):** 深度拷贝, a 和 b 完全拷贝了父对象及其子对象，两者是完全独立的。



列表与字符串转换

1. 列表转换为字符串：

result = "连接符".join(列表)

2. 字符串转换为列表：

列表 = “a-b-c-d”.split(“分隔符”)

列表推导式

1. 定义：

使用简易方法，将可迭代对象转换为列表。

2. 语法：

变量 = [表达式 for 变量 in 可迭代对象]

变量 = [表达式 for 变量 in 可迭代对象 if 条件]

3. 说明:

如果if真值表达式的布尔值为False,则可迭代对象生成的数据将被丢弃。

列表推导式嵌套

1. 语法：

变量 = [表达式 for 变量1 in 可迭代对象1 for 变量2 in可迭代对象2]

2. 传统写法：

```
result = []
for r in ["a", "b", "c"]:
  for c in ["A", "B", "C"]:
    result.append(r + c)
```

3. 推导式写法：

```
result = [r + c for r in list01 for c in list02]
```

元组 tuple

定义

1. 由一系列变量组成的不可变序列容器。
2. 不可变是指一但创建，不可以再添加/删除/修改元素。

基础操作

1. 创建空元组：

元组名 = ()

元组名 = tuple()

2. 创建非空元组：

元组名 = (20,)

元组名 = (1, 2, 3)

元组名 = 100,200,300

元组名 = tuple(可迭代对象)

3. 获取元素：

变量 = 元组名[索引]

变量 = 元组名[切片] # 赋值给变量的是切片所创建的新列表

4. 遍历元组：

正向：

for 变量名 in 列表名:

变量名就是元素

反向：

for 索引名 in range(len(列表名)-1,-1,-1):

元组名[索引名]就是元素

作用

1. 元组与列表都可以存储一系列变量，由于列表会预留内存空间，所以可以增加元素。
2. 元组会按需分配内存，所以如果变量数量固定，建议使用元组，因为占用空间更小。
3. 应用：

变量交换的本质就是创建元组：x, y = （y, x ）

格式化字符串的本质就是创建元祖："姓名:%s, 年龄:%d" % ("海绵宝宝", 15)

字典 dict

定义

1. 由一系列键值对组成的可变散列容器。
2. 散列：对键进行哈希运算，确定在内存中的存储位置，每条数据存储无先后顺序。
3. 键必须惟一且不可变(字符串/数字/元组)，值没有限制。

基础操作

1. 创建字典：

字典名 = {键1：值1，键2：值2}

字典名 = dict (可迭代对象)

2. 添加/修改元素：

语法:

字典名[键] = 数据

说明:

键不存在，创建记录。

键存在，修改值。

3. 获取元素：

变量 = 字典名[键] # 没有键则错误

4. 遍历字典：

for 键名 in 字典名:

字典名[键名]

for 键名,值名 in 字典名.items():

语句

5. 删除元素：

del 字典名[键]

字典推导式

1. 定义：

使用简易方法，将可迭代对象转换为字典。

2. 语法:

{键:值 for 变量 in 可迭代对象}

{键:值 for 变量 in 可迭代对象 if 条件}

集合 set

定义

1. 由一系列**不重复**的不可变类型变量(元组/数/字符串)组成的可变散列容器。
2. 相当于只有键没有值的字典(键则是集合的数据)。

基础操作

1. 创建空集合：

集合名 = set()

集合名 = set(可迭代对象)

2. 创建具有默认值集合：

集合名 = {1, 2, 3}

集合名 = set(可迭代对象)

3. 添加元素：

集合名.add(元素)

4. 删除元素：

集合名.discard(元素)

运算

1. 交集&：返回共同元素。

   s1 = {1, 2, 3}

   s2 = {2, 3, 4}

   s3 = s1 & s2 # {2, 3}

2. 并集：返回不重复元素

   s1 = {1, 2, 3}

   s2 = {2, 3, 4}

   s3 = s1 | s2 # {1, 2, 3, 4}

3. 补集-：返回只属于其中之一的元素

   s1 = {1, 2, 3}

   s2 = {2, 3, 4}

   s1 - s2 # {1} 属于s1但不属于s2

补集^：返回不同的的元素

s1 = {1, 2, 3}

s2 = {2, 3, 4}

s3 = s1 ^ s2 # {1, 4} 等同于(s1-s2 | s2-s1)

4. 子集<：判断一个集合的所有元素是否完全在另一个集合中
5. 超集>：判断一个集合是否具有另一个集合的所有元素

   s1 = {1, 2, 3}

   s2 = {2, 3}

s2 < s1 # True

s1 > s2 # True

6. 相同或不同== !=：判断集合中的所有元素是否和另一个集合相同。

   s1 = {1, 2, 3}

   s2 = {3, 2, 1}

   s1 == s2 # True

   s1 != s2 # False

子集或相同,超集或相同 <= >=

集合推导式

1. 定义：

使用简易方法，将可迭代对象转换为集合。

2. 语法:

{表达式 for 变量 in 可迭代对象}

{表达式 for 变量 in 可迭代对象 if 条件}

python函数(2022年1月10日-4)

定义

1. 用于封装一个特定的功能，表示一个功能或者行为。
2. 函数是可以重复执行的语句块, 可以重复调用。

作用

提高代码的可重用性和可维护性（代码层次结构更清晰）。

定义函数

1. 语法：

def 函数名(形式参数):

函数体

2. 说明：

def 关键字：全称是define，意为”定义”。

函数名：对函数体中语句的描述，规则与变量名相同。

形式参数：方法定义者要求调用者提供的信息。

函数体：完成该功能的语句。

3. 函数的第一行语句建议使用文档字符串描述函数的功能与参数。

调用函数

1. 语法：函数名(实际参数)
2. 说明：根据形参传递内容。

返回值

1. 定义：

方法定义者告诉调用者的结果。

2. 语法：

return 数据

3. 说明：

return后没有语句，相当于返回 None。

函数体没有return，相当于返回None。

可变 / 不可变类型在传参时的区别

1. 不可变类型参数有:

数值型(整数，浮点数)

布尔值bool

None 空值

字符串str

元组tuple

2. 可变类型参数有:

列表 list

字典 dict

集合 set

3. 传参说明:

不可变类型的数据传参时，函数内部不会改变原数据的值。

可变类型的数据传参时，函数内部可以改变原数据。

```
In [6]: def func01(list_target):
...:     list_target = list_target[3:]
...:     print(list_target)
...:
...:
...:
         In [7]: func01(list01)
[3, 0]
         In [8]: list01
Out[8]: [5, 3, 2, 3, 0]
```

?

函数参数

实参传递方式argument

位置传参

定义：实参与形参的位置依次对应。

序列传参

定义：实参用*将序列拆解后与形参的位置依次对应。

关键字传参

定义：实参根据形参的名字进行对应。

字典关键字传参

1. 定义：实参用**将字典拆解后与形参的名字进行对应。
2. 作用：配合形参的缺省参数，可以使调用者随意传参。

形参定义方式parameter

缺省形参

1. 语法：

def 函数名(形参名1=默认实参1, 形参名2=默认实参2, ...):

  函数体

2. 说明：

缺省参数必须自右至左依次存在，如果一个参数有缺省参数，则其右侧的所有参数都必须有缺省参数。

缺省参数可以有0个或多个，甚至全部都有缺省参数。

位置形参

语法：

def 函数名(形参名1, 形参名2, ...):

函数体

命名关键字形参

1. 语法：

   def 函数名(*args, 命名关键字形参1, 命名关键字形参2, ...):
     函数体

def 函数名(*, 命名关键字形参1, 命名关键字形参2, ...):

函数体

2. 作用：

强制实参使用关键字传参

不定长形参

星号元组形参

1. 语法：

def 函数名(*元组形参名):

 函数体

2. 作用：

可以将多个位置实参合并为一个元组

3. 说明：

一般命名为'args'

形参列表中最多只能有一个

双星号字典形参

1. 语法：

def 函数名(**字典形参名):

函数体

2. 作用：

可以将多个关键字实参合并为一个字典

3. 说明:

一般命名为'kwargs'

形参列表中最多只能有一个

参数自左至右的顺序

位置形参 --> 星号元组形参 --> 命名关键字形参 --> 双星号字典形参

python作用域LEGB(2022年1月10日-5)

1. 作用域：变量起作用的范围。
2. Local局部作用域：函数内部。
3. Enclosing 外部嵌套作用域 ： 函数嵌套。
4. Global全局作用域：模块(.py文件)内部。
5. Builtin内置模块作用域：builtins.py文件。

变量名的查找规则

1. 由内到外：L -> E -> G -> B
2. 在访问变量时，先查找本地变量，然后是包裹此函数外部的函数内部的变量，之后是全局变量，最后是内置变量。

局部变量

1. 定义在函数内部的变量(形参也是局部变量)
2. 只能在函数内部使用
3. 调用函数时才被创建，函数结束后自动销毁

全局变量

1. 定义在函数外部,模块内部的变量。
2. 在整个模块(py文件)范围内访问（但函数内不能将其直接赋值）。

global 语句

1. 作用：

在函数内部修改全局变量。

在函数内部定义全局变量(全局声明)。

2. 语法：

global 变量1, 变量2, ...

3. 说明

在函数内直接为全局变量赋值，视为创建新的局部变量。

不能先声明局部的变量，再用global声明为全局变量。

nonlocal 语句

1. 作用：

在内层函数修改外层嵌套函数内的变量

2. 语法

nonlocal 变量名1,变量名2, ...

3. 说明

在被嵌套的内函数中进行使用

2022年1月10日作业

作业一(基础练习)

1. 在终端中输入一个疫情确诊人数再录入一个治愈人数，打印治愈比例

格式：治愈比例为xx%

效果：

请输入确诊人数：500

请输入治愈人数：495

治愈比例为99.0%

2. 程序产生1个,1到100之间的随机数。

让玩家重复猜测,直到猜对为止。

每次提示：大了、小了、恭喜猜对了,总共猜了多少次。

效果(假设随机数为32)：

请输入要猜的数字:50

大了

请输入要猜的数字:25

小了

请输入要猜的数字:35

大了

请输入要猜的数字:30

小了

请输入要猜的数字:32

恭喜猜对啦,总共猜了5次

3. 在终端中,循环录入字符串,如果录入空则停止.

停止录入后打印所有内容(一个字符串)

效果：

```
请输入内容：香港

请输入内容：上海

请输入内容：新疆
请输入内容：
香港_上海_新疆
```

4. 将下列英文语句按照单词进行翻转.

转换前：To have a government that is of people by people for people

转换后：people for people by people of is that government a have To

作业二(容器训练)

1. 运用循环在终端中打印如下图形

```
   *
  ***
 *****
*******
```

2. 二维列表

list01 = [

[1, 2, 3, 4, 5],

[6, 7, 8, 9, 10],

[11, 12, 13, 14, 15],

]

1. 将第一行从左到右逐行打印

效果:

1

2

3

4

5

2. 将第二行从右到左逐行打印

效果:

10

9

8

7

6

3. 将第三列行从上到下逐个打印

效果:

3

8

13

4. 将第四列行从下到上逐个打印

效果:

14

9

4

5. 将二维列表以表格状打印

效果:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

3. dict_hobbies = {

"于谦": ["抽烟", "喝酒", "烫头"],

"郭德纲": ["说", "学", "逗", "唱"],

}

1. 打印于谦的所有爱好(一行一个)

效果:

抽烟

喝酒

烫头

2. 计算郭德纲所有爱好数量

效果: 4

3. 打印所有人(一行一个)

效果:

于谦

郭德纲

4. 打印所有爱好(一行一个)

抽烟

喝酒

烫头

说

学

逗

唱

作业三(函数训练)

根据下列代码，创建函数。

```
# 商品字典
dict_commodity_infos = {
  1001: {"name": "屠龙刀", "price": 10000},
  1002: {"name": "倚天剑", "price": 10000},
  1003: {"name": "金箍棒", "price": 52100},
  1004: {"name": "口罩", "price": 20},
```

```
  1005: {"name": "酒精", "price": 30},
}

# 订单列表
list_orders = [
  {"cid": 1001, "count": 1},
  {"cid": 1002, "count": 3},
  {"cid": 1005, "count": 2},
]

# 1.定义函数，打印所有商品信息,
for cid, info in dict_commodity_infos.items():
  # print(f"商品编号{cid},商品名称{info['name']},商品单价{info['price']}.")
  print("商品编号%d,商品名称%s,商品单价%d." % (cid, info["name"], info["price"]))

# 2. 定义函数，打印单价大于10000的商品信息,
for order in list_orders:
    if info["price"]>10000:
  # print(f"商品编号{cid},商品名称{info['name']},商品单价{info['price']}.")
  print("商品编号%d,商品名称%s,商品单价%d." % (cid, info["name"], info["price"]))

# 3. 定义函数，查找数量最多的订单(使用自定义算法,不使用内置函数)
max_value = list_orders[0]
for i in range(1, len(list_orders)):
  if max_value["count"] < list_orders[i]["count"]:
    max_value = list_orders[i]
print(max_value)
# 4. 定义函数，根据购买数量对订单列表降序(大->小)排列
```

作业四 长期练习

e.g:

```
Question:

Write a program to solve a classic ancient Chinese puzzle:
We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many
rabbits and how many chickens do we have?

Hint:
Use for loop to iterate all possible solutions.
```

答案

python进阶

路漫漫其修远兮...

文件读写,面向对象编程(类),正则表达式,数据处理...

文件读写

文本文件

面向对象

class

正则表达式

re

数据处理

pandas

其他有趣的包

yagmail

数据库

oracle mysql postgresql

web框架

django flask

爬虫

requests post get

nginx,apache

反向代理

git.....

代码管理