Python&Git

Python

- 一种弱类型语言 不需要单独指定类型
- 使用 # 作为注释符号
- 三种序列 列表（list）、元组（tuple）、数列（range）

列表list

列表：可变序列 使用方括号表示[] 可存放不同类型的数据

- 添加元素：append
- 清空 clear
- 浅复制 copy
- 统计某一元素出现的次数 count
- 插入 insert
- 排序 sort
- 翻转：reverse
- 切片 list[start:stop:step] start的默认值是0 step的默认值是1

```
In [1]: l1 = [0,1,2,3,4,5,6,7,8]

In [3]: l1.count(1)
Out[3]: 1

In [4]: l1[0:3:1]  # start:stop:step  开始索引 结束索引 步长
        # 取不到结束索引对应的值
Out[4]: [0, 1, 2]

In [5]: l1[0:3:2]
Out[5]: [0, 2]

In [6]: l1[:3]
Out[6]: [0, 1, 2]

In [7]: l1[0:]
Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8]

In [8]: l1[:]
Out[8]: [0, 1, 2, 3, 4, 5, 6, 7, 8]

In [9]: l1[::-1]
Out[9]: [8, 7, 6, 5, 4, 3, 2, 1, 0]

In [10]: l1[::2]
Out[10]: [0, 2, 4, 6, 8]
```

元组 tuple

- 元组使用圆括号()表示
- 元组和列表可以相互转换 list(tupleName) tuple(listName)

```
In [1]: list1=[1,2]

In [2]: list1.reverse()

In [3]: list1
Out[3]: [2, 1]

In [4]: type(list1)
Out[4]: list

In [5]: list1=tuple(list1)

In [6]: type(list1)
Out[6]: tuple

In [11]: list1
Out[11]: (2, 1)
```

数列 range

- 常用于for循环中的控制迭代
- range(start,stop,step) step默认值为1 前闭后开区间

```
In [13]: for i in range(10):
             print(i)

0
1
2
3
4
5
6
7
8
9
```

```
In [14]: for i in range(0,5,1):
             print(i)

0
1
2
3
4
```

```
In [15]: for i in range(5,0,-1):
             print(i)

5
4
3
2
1
```

字符串 str

- python中字符串可以用单引号、双引号、三引号构成
- join方法 str.join(str)

```
In [16]: str1=""

In [17]: str2="abc"

In [18]: str1.join(str2)
Out[18]: 'abc'
```

集合 set

- 常用 用于算法题中进行元素去重
- 集合中的元素具有唯一性 不可重复出现

```
In [19]: l1=[1,2,2,3,3,4]

In [20]: l1
Out[20]: [1, 2, 2, 3, 3, 4]

In [21]: set(l1)
Out[21]: {1, 2, 3, 4}
```

- frozenset 不可更改的set

函数

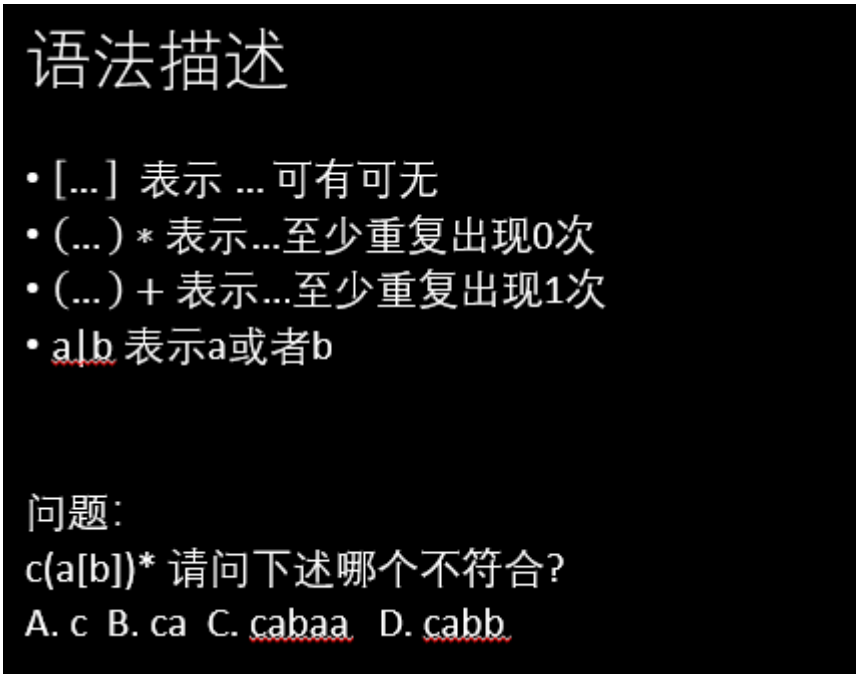
- def 注意使用冒号

```
In [22]: def test(a):
             print("a={}".format(a))
             return a

In [24]: a=test(2)
         a=2

In [25]: a
Out[25]: 2
```

控制语句



- if while for需要掌握 这些句子的后面都需要冒号:
- try except语句用于异常捕获与处理

- ```
  try:
      1/0
  except Exception as e:
      print(e)
  finally:
      print("run over")
  ```

- with语句块常用于打开文件
- with open(file)

类

- class表示类 注意冒号
- 初始化写在 _ _ init _ _函数中
- 时刻留意self的使用
- 类中所有的函数的第一个形参永远且必须是self
- 类的成员使用 self.data

```
In [1]: class ex:
            def __init__(self):
                self.a="48"
                self.b="26"
            def add(self):
                self.c = self.a + self.b
            def cout(self):
                print(self.c)

In [4]: a=ex()

In [5]: a.cout()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\ADMINI~1\AppData\Local\Temp/ipykernel_11992/493893658.py in <module>
----> 1 a.cout()

C:\Users\ADMINI~1\AppData\Local\Temp/ipykernel_11992/921695624.py in cout(self)
      6         self.c = self.a + self.b
      7     def cout(self):
----> 8         print(self.c)

AttributeError: 'ex' object has no attribute 'c'
```

```
In [6]: a.add()

In [7]: a.cout()
```

```
4826
```

Git

- 配置全局信息：git config --global user.name "name"
- git config --global user.email "email"
- 初始化仓库 git init
- 添加所有文件 git add .
```

- 添加单个文件 git add filename
- 提交到缓冲区 git commit -m "message"
- 提交到仓库 git push origin branchName
- 克隆仓库 git clone URL
- 分支操作
    - 新建分支 git branch branchName
    - 切换分支 git checkout branchName
    - 查看当前仓库的所有分支 git branch
- 查看git 日志 git log
- 查看当前状态 git status