**Python基础数据类型详解**

**主要内容:**

1. 格式化输出

2. 简单运算符

3. 编码初识(ascii,unicode,utf-8,gbk等历史)以及bytes

4. 基础数据类型bool

5. 基础数据类型str

6. 基础数据类型list

7. 基础数据类型tuple

8. 基础数据类型dict

9. 基本数据类型Set

     10.深浅copy

11. 知识点补充

12. 文件操作

**一. 格式化输出**

 现在有以下需求,让用户输入name, age, job,hobby 然后输出如下所示:

------------ info of Sylar -----------

Name  : Sylar

Age   : 22

job   : Teacher

Hobbie: girl

------------- end -----------------

你怎么实现呢？你会发现，用字符拼接的方式还难实现这种格式的输出，所以一起来学一下新姿势

只需要把要打印的格式先准备好, 由于里面的 一些信息是需要用户输入的，你没办法预设知道，因此可以先放置个占位符，再把字符串里的占位符与外部的变量做个映射关系就好啦

name = input("Name:")

age = input("Age:")

job = input("Job:")

hobby = input("Hobbie:")

info = '''

------------ info of %s ----------- #这里的每个%s就是一个占位符，本行的代表 后面拓号里的 name

Name  : %s  #代表 name

Age   : %s  #代表 age

job   : %s  #代表 job

Hobbie: %s  #代表 hobbie

------------- end -----------------

''' % (name,name,age,job,hobbie)  # 这行的 % 号就是 把前面的字符串 与拓号 后面的 变量 关联起来

print(info)

%s就是代表字符串占位符，除此之外，还有%d, 是数字占位符， 如果把上面的age后面的换成%d，就代表你必须只能输入数字啦, 这时对应的数据必须是int类型. 否则程序会报错

使用时,需要进行类型转换.

int(str)    # 字符串转换成int

str(int)    # int转换成字符串

类似这样的操作在后面还有很多

如果, 你头铁. 就不想转换. 觉着转换很麻烦. 也可以全部都用%s. 因为任何东西都可以直接转换成字符串--> 仅限%s

现在又来新问题了. 如果想输出:

我叫xxx, 今年xx岁了,我们已经学习了2%的python基础了

这里的问题出在哪里呢? 没错2%, 在字符串中如果使用了%s这样的占位符. 那么所有的%都将变成占位符. 我们的2%也变成了占位符. 而"%的"是不存在的, 这里我们需要使用%%来表示字符串中的%.

注意: 如果你的字符串中没有使用过%s,%d站位. 那么不需要考虑这么多. 该%就%.没毛病老铁.

print("我叫%s, 今年22岁了, 学习python2%%了" % '王尼玛')     # 有%占位符

print("我叫王尼玛, 今年22岁, 已经凉凉了100%了")       # 没有占位符

python3.5以后可以使用f来格式化字符串.

语法 : f"{变量}"

name = "sylar"

print(f"{name}真是一个蠢蠢的小蠢蠢")

**二. 基本运算符**

    计算机可以进行的运算有很多种，可不只加减乘除这么简单，运算按种类可分为:

        算数运算、

        比较运算、

        逻辑运算、

        赋值运算、

        成员运算、

        身份运算、

        位运算.

    今天我们暂时学习前面的几个. 后面的身份运算和位运算咱们把它放在后面再讲.

**2.1 算数运算**

以下假设变量：a=10，b=20



**2.2 比较运算**

以下假设变量：a=10，b=20



**2.3 赋值运算**

以下假设变量：a=10，b=20



**2.4 算逻辑运->重点**



逻辑运算的运算顺序: () > not > and > or

例题：

判断下列逻辑语句的True，False。

3>4 or 4<3 and 1==1

1 < 2 and 3 < 4 or 1>2

2 > 1 and 3 < 4 or 4 > 5 and 2 < 1

1 > 2 and 3 < 4 or 4 > 5 and 2 > 1 or 9 < 8

1 > 1 and 3 < 4 or 4 > 5 and 2 > 1 and 9 > 8 or 7 < 6

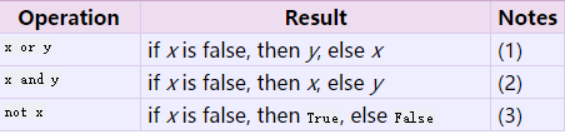
not 2 > 1 and 3 < 4 or 4 > 5 and 2 > 1 and 9 > 8 or 7 < 6

综上, 记住, 使用复杂的逻辑运算的时候切记加括号.

逻辑运算符前后有可能是数字. 此时如何进行运算呢? (了解)

x or y , x为真，值就是x, x为假，值是y；

             x and y, x为真，值是y, x为假，值是x。



例题：求出下列逻辑语句的值。

8 or 4

0 and 3

0 or 4 and 3 or 7 or 9 and 6

**2.5 成员运算**

从名字上也能看出来, 成员运算就是判断xxxx是否在xxxxx中出现了. 比如, 判断一句话中是否包含了"苍井空"

content = input("请输入你的评论内容:")

if "苍井空" in content:

print("存在苍井空")

else:

print("不存在苍井空")

成员运算还有一个叫 not...in 逻辑和in正好相反. 表示xxxx是否不在xxxx中出现.

例如,

content = input("请输入你的评论内容:")

if "苍井空" not in content:

print("不存在苍井空, 该评论合法")

else:

print("存在苍井空, 该评论不合法")

**三. 编码的问题(记结论)**

计算机是一个电子产品, 它内部其实是用通电和不通电来记录数据的. 通电和不通电一共就两种状态. 也就是只能描述0和1. 那么问题来了. 数字还好, 转化成2进制就可以了, 文字怎么办? 聪明的科学家就把文字按照一个固定的顺序编排成数字. 以后看见这个数字就表示相对应的文字. 反过来. 我们看到的文字, 对应存储在计算机中就是一串数字. 这个过程被称为编码.

早期. 计算机是美国发明的. 普及率不高, 一般只是在美国使用. 所以. 最早的编码结构就是按照美国人的习惯来编码的. 对应数字+字母+特殊字符一共也没多少. 所以就形成了最早的编码ASCII码. 直到今天ASCII依然深深的影响着我们. ASCII（American Standard Code for Information Interchange，美国标准信息交换代码）是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言，其最多只能用 8 位来表示（一个字节），即：2\*\*8 = 256，所以，ASCII码最多只能表示 256 个符号。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bin(二进制) | Oct(八进制) | Dec(十进制) | Hex(十六进制) | 缩写/字符 | 解释 |
| 0000 0000 | 0 | 0 | 00 | NUL(null) | 空字符 |
| 0000 0001 | 1 | 1 | 01 | SOH(start of headline) | 标题开始 |
| 0000 0010 | 2 | 2 | 02 | STX (start of text) | 正文开始 |
| 0000 0011 | 3 | 3 | 03 | ETX (end of text) | 正文结束 |
| 0000 0100 | 4 | 4 | 04 | EOT (end of transmission) | 传输结束 |
| 0000 0101 | 5 | 5 | 05 | ENQ (enquiry) | 请求 |
| 0000 0110 | 6 | 6 | 06 | ACK (acknowledge) | 收到通知 |
| 0000 0111 | 7 | 7 | 07 | BEL (bell) | 响铃 |
| 0000 1000 | 10 | 8 | 08 | BS (backspace) | 退格 |
| 0000 1001 | 11 | 9 | 09 | HT (horizontal tab) | 水平制表符 |
| 0000 1010 | 12 | 10 | 0A | LF (NL line feed, new line) | 换行键 |
| 0000 1011 | 13 | 11 | 0B | VT (vertical tab) | 垂直制表符 |
| 0000 1100 | 14 | 12 | 0C | FF (NP form feed, new page) | 换页键 |
| 0000 1101 | 15 | 13 | 0D | CR (carriage return) | 回车键 |
| 0000 1110 | 16 | 14 | 0E | SO (shift out) | 不用切换 |
| 0000 1111 | 17 | 15 | 0F | SI (shift in) | 启用切换 |
| 0001 0000 | 20 | 16 | 10 | DLE (data link escape) | 数据链路转义 |
| 0001 0001 | 21 | 17 | 11 | DC1 (device control 1) | 设备控制1 |
| 0001 0010 | 22 | 18 | 12 | DC2 (device control 2) | 设备控制2 |
| 0001 0011 | 23 | 19 | 13 | DC3 (device control 3) | 设备控制3 |
| 0001 0100 | 24 | 20 | 14 | DC4 (device control 4) | 设备控制4 |
| 0001 0101 | 25 | 21 | 15 | NAK (negative acknowledge) | 拒绝接收 |
| 0001 0110 | 26 | 22 | 16 | SYN (synchronous idle) | 同步空闲 |
| 0001 0111 | 27 | 23 | 17 | ETB (end of trans. block) | 结束传输块 |
| 0001 1000 | 30 | 24 | 18 | CAN (cancel) | 取消 |
| 0001 1001 | 31 | 25 | 19 | EM (end of medium) | 媒介结束 |
| 0001 1010 | 32 | 26 | 1A | SUB (substitute) | 代替 |
| 0001 1011 | 33 | 27 | 1B | ESC (escape) | 换码(溢出) |
| 0001 1100 | 34 | 28 | 1C | FS (file separator) | 文件分隔符 |
| 0001 1101 | 35 | 29 | 1D | GS (group separator) | 分组符 |
| 0001 1110 | 36 | 30 | 1E | RS (record separator) | 记录分隔符 |
| 0001 1111 | 37 | 31 | 1F | US (unit separator) | 单元分隔符 |
| 0010 0000 | 40 | 32 | 20 | (space) | 空格 |
| 0010 0001 | 41 | 33 | 21 | ! | 叹号 |
| 0010 0010 | 42 | 34 | 22 | " | 双引号 |
| 0010 0011 | 43 | 35 | 23 | # | 井号 |
| 0010 0100 | 44 | 36 | 24 | $ | 美元符 |
| 0010 0101 | 45 | 37 | 25 | % | 百分号 |
| 0010 0110 | 46 | 38 | 26 | & | 和号 |
| 0010 0111 | 47 | 39 | 27 | ' | 闭单引号 |
| 0010 1000 | 50 | 40 | 28 | ( | 开括号 |
| 0010 1001 | 51 | 41 | 29 | ) | 闭括号 |
| 0010 1010 | 52 | 42 | 2A | \* | 星号 |
| 0010 1011 | 53 | 43 | 2B | + | 加号 |
| 0010 1100 | 54 | 44 | 2C | , | 逗号 |
| 0010 1101 | 55 | 45 | 2D | - | 减号/破折号 |
| 0010 1110 | 56 | 46 | 2E | . | 句号 |
| 00101111 | 57 | 47 | 2F | / | 斜杠 |
| 00110000 | 60 | 48 | 30 | 0 | 数字0 |
| 00110001 | 61 | 49 | 31 | 1 | 数字1 |
| 00110010 | 62 | 50 | 32 | 2 | 数字2 |
| 00110011 | 63 | 51 | 33 | 3 | 数字3 |
| 00110100 | 64 | 52 | 34 | 4 | 数字4 |
| 00110101 | 65 | 53 | 35 | 5 | 数字5 |
| 00110110 | 66 | 54 | 36 | 6 | 数字6 |
| 00110111 | 67 | 55 | 37 | 7 | 数字7 |
| 00111000 | 70 | 56 | 38 | 8 | 数字8 |
| 00111001 | 71 | 57 | 39 | 9 | 数字9 |
| 00111010 | 72 | 58 | 3A | : | 冒号 |
| 00111011 | 73 | 59 | 3B | ; | 分号 |
| 00111100 | 74 | 60 | 3C | < | 小于 |
| 00111101 | 75 | 61 | 3D | = | 等号 |
| 00111110 | 76 | 62 | 3E | > | 大于 |
| 00111111 | 77 | 63 | 3F | ? | 问号 |
| 01000000 | 100 | 64 | 40 | @ | 电子邮件符号 |
| 01000001 | 101 | 65 | 41 | A | 大写字母A |
| 01000010 | 102 | 66 | 42 | B | 大写字母B |
| 01000011 | 103 | 67 | 43 | C | 大写字母C |
| 01000100 | 104 | 68 | 44 | D | 大写字母D |
| 01000101 | 105 | 69 | 45 | E | 大写字母E |
| 01000110 | 106 | 70 | 46 | F | 大写字母F |
| 01000111 | 107 | 71 | 47 | G | 大写字母G |
| 01001000 | 110 | 72 | 48 | H | 大写字母H |
| 01001001 | 111 | 73 | 49 | I | 大写字母I |
| 01001010 | 112 | 74 | 4A | J | 大写字母J |
| 01001011 | 113 | 75 | 4B | K | 大写字母K |
| 01001100 | 114 | 76 | 4C | L | 大写字母L |
| 01001101 | 115 | 77 | 4D | M | 大写字母M |
| 01001110 | 116 | 78 | 4E | N | 大写字母N |
| 01001111 | 117 | 79 | 4F | O | 大写字母O |
| 01010000 | 120 | 80 | 50 | P | 大写字母P |
| 01010001 | 121 | 81 | 51 | Q | 大写字母Q |
| 01010010 | 122 | 82 | 52 | R | 大写字母R |
| 01010011 | 123 | 83 | 53 | S | 大写字母S |
| 01010100 | 124 | 84 | 54 | T | 大写字母T |
| 01010101 | 125 | 85 | 55 | U | 大写字母U |
| 01010110 | 126 | 86 | 56 | V | 大写字母V |
| 01010111 | 127 | 87 | 57 | W | 大写字母W |
| 01011000 | 130 | 88 | 58 | X | 大写字母X |
| 01011001 | 131 | 89 | 59 | Y | 大写字母Y |
| 01011010 | 132 | 90 | 5A | Z | 大写字母Z |
| 01011011 | 133 | 91 | 5B | [ | 开方括号 |
| 01011100 | 134 | 92 | 5C | \ | 反斜杠 |
| 01011101 | 135 | 93 | 5D | ] | 闭方括号 |
| 01011110 | 136 | 94 | 5E | ^ | 脱字符 |
| 01011111 | 137 | 95 | 5F | \_ | 下划线 |
| 01100000 | 140 | 96 | 60 | ` | 开单引号 |
| 01100001 | 141 | 97 | 61 | a | 小写字母a |
| 01100010 | 142 | 98 | 62 | b | 小写字母b |
| 01100011 | 143 | 99 | 63 | c | 小写字母c |
| 01100100 | 144 | 100 | 64 | d | 小写字母d |
| 01100101 | 145 | 101 | 65 | e | 小写字母e |
| 01100110 | 146 | 102 | 66 | f | 小写字母f |
| 01100111 | 147 | 103 | 67 | g | 小写字母g |
| 01101000 | 150 | 104 | 68 | h | 小写字母h |
| 01101001 | 151 | 105 | 69 | i | 小写字母i |
| 01101010 | 152 | 106 | 6A | j | 小写字母j |
| 01101011 | 153 | 107 | 6B | k | 小写字母k |
| 01101100 | 154 | 108 | 6C | l | 小写字母l |
| 01101101 | 155 | 109 | 6D | m | 小写字母m |
| 01101110 | 156 | 110 | 6E | n | 小写字母n |
| 01101111 | 157 | 111 | 6F | o | 小写字母o |
| 01110000 | 160 | 112 | 70 | p | 小写字母p |
| 01110001 | 161 | 113 | 71 | q | 小写字母q |
| 01110010 | 162 | 114 | 72 | r | 小写字母r |
| 01110011 | 163 | 115 | 73 | s | 小写字母s |
| 01110100 | 164 | 116 | 74 | t | 小写字母t |
| 01110101 | 165 | 117 | 75 | u | 小写字母u |
| 01110110 | 166 | 118 | 76 | v | 小写字母v |
| 01110111 | 167 | 119 | 77 | w | 小写字母w |
| 01111000 | 170 | 120 | 78 | x | 小写字母x |
| 01111001 | 171 | 121 | 79 | y | 小写字母y |
| 01111010 | 172 | 122 | 7A | z | 小写字母z |
| 01111011 | 173 | 123 | 7B | { | 开花括号 |
| 01111100 | 174 | 124 | 7C | | | 垂线 |
| 01111101 | 175 | 125 | 7D | } | 闭花括号 |
| 01111110 | 176 | 126 | 7E | ~ | 波浪号 |
| 01111111 | 177 | 127 | 7F | DEL (delete) | 删除 |

随着计算机的发展. 以及普及率的提高. 流行到欧洲和亚洲. 这时ASCII码就不合适了. 比如: 中文汉字有几万个. 而ASCII最多也就256个位置. 所以ASCII不行了. 怎么办呢? 这时, 不同的国家就提出了不同的编码用来适用于各自的语言环境. 比如, 中国的GBK, GB2312, BIG5, ISO-8859-1等等. 这时各个国家都可以使用计算机了.

GBK, 国标码占用2个字节. 对应ASCII码 GBK直接兼容. 因为计算机底层是用英文写的. 你不支持英文肯定不行. 而英文已经使用了ASCII码. 所以GBK要兼容ASCII.

这里GBK国标码. 前面的ASCII码部分. 由于使用两个字节. 所以对于ASCII码而言. 前几位都是0.

字母A:0100 0001        # ASCII 1kb

字母A:0000 0000 0100 0001 # 国标码 2kb

字母A: 0000 0000 0000 1000 0000 0000 0100 0001 # unicode 4kb

国标码的弊端: 只能中国用. 到了非洲就垮了. 所以国标码不满足我们的使用. 这时提出了一个万国码Unicode. Unicode一开始设计是每个字符两个字节. 设计完了. 发现我大中国汉字依然无法进行编码. 只能进行扩充. 扩充成32位也就是4个字节. 这回够了. 但是. 问题来了. 中国字9万多. 而unicode可以表示41亿多的文字. 根本用不了. 太浪费了. 于是乎, 就提出了新的UTF编码.可变长度编码叫UTF, 我们用的最多的就是UTF-8.

GBK: 每个字符占2个字节, 16位.

UTF-8: 每个字符最少占8位. 每个字符占用的字节数不定.根据文字内容进行具体编码. 比如. 英文. 就一个字节就够了. 汉字占3个字节. 这时即满足了中文. 也满足了节约. 也是目前使用频率最高的一种编码

英文: 1byte, 8bit

欧洲: 2byte, 16bit

中文: 3byte, 24bit

单位转换:

8bit = 1byte

1024byte = 1KB

1024KB = 1MB

1024MB = 1GB

1024GB = 1TB

1024TB = 1PB

1024PB = 1EB

1024EB = 1ZB

1024ZB = 1YB

1024YB = 1NB

1024NB = 1DB

常用到TB就够了

结论:

1. ascii : 8bit, 主要存放的是英文, 数字, 特殊符号

2. gbk: 16bit, 主要存放中文和亚洲字符. 兼容ascii

3. unicode: 16bit和32bit两个版本. 平时我们用的是16bit这个版本. 全世界所有国家的文字信息. 缺点: 浪费空间(传输和存储)

4. utf-8 : 可变长度unicode, 英文: 8bit, 欧洲文字: 16bit, 中文24bit. 一般数据传输和存储的时候使用.

5. 以上所有编码必须兼容ascii .

**3.2 bytes类型**

在python程序中, 当我们的程序运行起来之后. 内存中存储的字符串默认使用的是unicode. 目的是unicode定长. 好处理. 但是, 如果涉及到字符串存储和传输, 就必须要进行编码. 编码成utf-8或者gbk进行传递. 原因是unicode实在是太浪费空间了.

编码: encode()

s = "中国"

bs = s.encode("utf-8")

print(bs) # b'\xe4\xb8\xad\xe5\x9b\xbd'

上例中, 我们看到这样一串特殊的内容b'\xe4\xb8\xad\xe5\x9b\xbd' 这个东西在python中被称为bytes. 我们可以通过type来查看数据类型

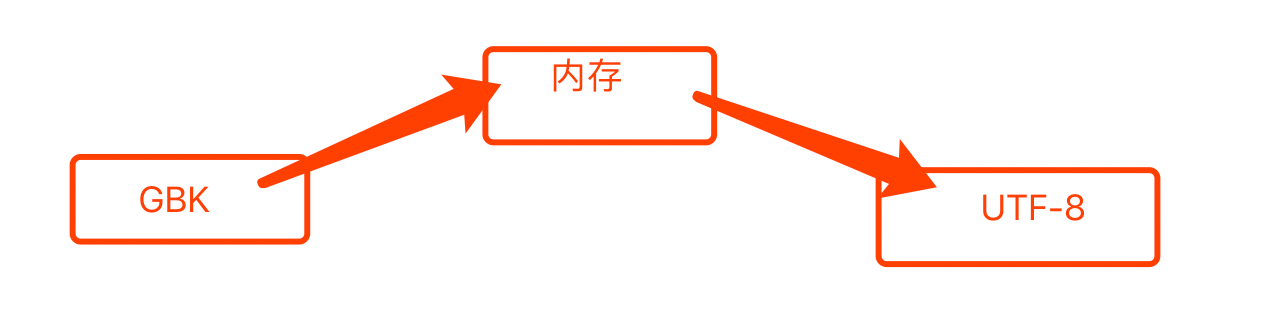
print(type(b'\xe4\xb8\xad\xe5\x9b\xbd')) # <class 'bytes'>

这里每一个\x表示一个字节. 数一数就能发现 两个汉字, 对应6个字节. 是符合UTF-8的编码规则的. 那如果编码成GBK呢?

print(s.encode("gbk")) # b'\xd6\xd0\xb9\xfa'

数一数, 4个字节. 也符合gbk的标准.

从上面的结果中我们就能发现, gbk和utf-8是不兼容的. 必须通过程序进行转换. 此时, 我们需要这样做



此时问题来了. 如何把编码之后的bytes转化回我们的字符串?

decode()

bs = b'\xe4\xb8\xad\xe5\x9b\xbd'

s = bs.decode("utf-8") # 只能用utf-8解码

print(s) # 中国

# 重新编码成gbk

bss = s.encode("gbk")

print(bss) # b'\xd6\xd0\xb9\xfa'

总结: 编码用encode() 解码用decode()

**四. 基础数据类型bool**

bool我们已经使用了很长时间了. 它主要的作用就是用来做条件判断, 所以bool类型的变量没有什么操作 这里要给大家聊的是bool类型和其他数据类型之间的转换问题

    转换问题:

        str => int       int(str)

        int => str       str(int)

        int => bool    bool(int).  0是False 非0是True

        bool=>int      int(bool)   True是1, False是0

        str => bool    bool(str)  空字符串是False,  不空是True

        bool => str    str(bool)  把bool值转换成相应的"值"

# 各种数据类型转化成bool

print(bool(0)) # False

print(bool(1)) # True

print(bool(-1)) # True

print(bool("")) # False

print(bool(" ")) # True

print(bool("哈哈")) # True

print(bool([])) # False

print(bool([1,2,3])) # True

print(bool({})) # False

while 1:

pass

结论: 所有表示空的东西都是False

基本数据类型之间的转化, 想变成谁, 就用谁把数据括起来

**五. 基础数据类型str**

把字符连成串. 在python中用', ", ''', """引起来的内容被称为字符串.

字符串: 把多个字符连成串

**5.1 切片和索引**

**5.1.1 索引.**

索引就是下标, 切记, 下标从0开始

#     0123456 7 8

s1 = "python最牛B"

print(s1[0])    # 获取第0个

print(s1[1])

print(s1[2])

print(s1[3])

print(s1[4])

print(s1[5])

print(s1[6])

print(s1[7])

print(s1[8])

# print(s1[9])    # 没有9, 越界了. 会报错

print(s1[-1])   # -1 表示倒数.

print(s1[-2])   # 倒数第二个

**5.1.2 切片**

我们可以使用下标来截取部分字符串的内容

    基本语法: str[start: end]

    规则: 顾头不顾尾, 从start开始截取. 截取到end位置. 但不包括end

s2 = "python最牛B"

print(s2[0:3])  # 从0获取到3. 不包含3. 结果: pyt

print(s2[6:8])  # 结果 最牛

print(s2[6:9])  # 最大是8. 但根据顾头不顾腚, 想要取到8必须给9

print(s2[6:10])  # 如果右边已经过了最大值. 相当于获取到最后

print(s2[4:])   # 如果想获取到最后. 那么最后一个值可以不给.

print(s2[-1:-5])    # 从-1 获取到 -5 这样是获取不到任何结果的. 从-1向右数. 你怎么数也数不到-5

print(s2[-5:-1])    # 牛b, 取到数据了. 但是. 顾头不顾腚. 怎么取最后一个呢?

print(s2[-5:])  # 什么都不写就是最后了

print(s2[:-1])  # 这个是取到倒数第一个

print(s2[:])    # 原样输出

     跳着截取

# 跳着取, 步长

print(s2[1:5:2])    # 从第一个开始取, 取到第5个,每2个取1个, 结果: yh, 分析: 1:5=> ytho => yh

print(s2[:5:2])     # 从头开始到第五个. 每两个取一个

print(s2[4::2])     # 从4开始取到最后. 每两个取一个

print(s2[-5::2])    # 从-5取到最后.每两个取一个

print(s2[-1:-5])    # -1:-5什么都没有. 因为是从左往右获取的.

print(s2[-1:-5:-1])  # 步长是-1. 这时就从右往左取值了

print(s2[-5::-3])   # 从倒数第5个开始. 到最开始. 每3个取一个, 结果oy

步长: 如果是整数, 则从左往右取. 如果是负数. 则从右往左取. 默认是1

切片完整语法:

       str[start:end:step]

    start: 起始位置

    end: 结束位置

    step: 步长

**5.2 字符串的相关操作方法(记结论)**

切记, 字符串是不可变的对象, 所以任何操作对原字符串是不会有任何影响的

1. 大小写转来转去

s1.capitalize()

print(s1)   # 输出发现并没有任何的变化. 因为这里的字符串本身是不会发生改变的. 需要我们重新获取

ret1 = s1.capitalize()

print(ret1)

# 大小写的转换

ret = s1.lower()    # 全部转换成小写

print(ret)

ret = s1.upper()     # 全部转换成大写

print(ret)

# 应用, 校验用户输入的验证码是否合法

verify\_code = "abDe"

user\_verify\_code = input("请输入验证码:")

if verify\_code.upper() == user\_verify\_code.upper():

    print("验证成功")

else:

    print("验证失败")

ret = s1.swapcase()     # 大小写互相转换

print(ret)

# 不常用

ret = s1.casefold()     # 转换成小写, 和lower的区别: lower()对某些字符支持不够好. casefold()对所有字母都有效. 比如东欧的一些字母

print(ret)

s2 = "БBß"  # 俄美德

print(s2)

print(s2.lower())

print(s2.casefold())

# 每个被特殊字符隔开的字母首字母大写

s3 = "sylar chuchu,baoheizi\*展昭\_麻花藤"

ret = s3.title()       # Sylar chuchu,baoheizi\*展昭\_麻花藤

print(ret)

# 中文也算是特殊字符

s4 = "sylar最喜欢heihei\_bao"     # Sylar最喜欢Heihei\_Bao

print(s4.title())

2. 切来切去

# 居中

s5 = "周杰伦"

ret = s5.center(10, "\*")   # 拉长成10, 把原字符串放中间.其余位置补\*

print(ret)

# 去空格

s7 = "   heihei bao   haha "

ret = s7.strip()    # 去掉左右两端的空格

print(ret)

ret = s7.lstrip()   # 去掉左边空格

print(ret)

ret = s7.rstrip()   # 去掉右边空格

print(ret)

# 应用, 模拟用户登录. 忽略用户输入的空格

username = input("请输入用户名:").strip()

password = input("请输入密码: ").strip()

if username == 'ren' and password == '123':

    print("登录成功")

else:

    print("登录失败")

s7 = "abcdefgabc"

print(s7.strip("abc"))  # defg 也可以指定去掉的元素,

# 字符串替换

s8 = "bao\_heihei\_nezha\_huhu"

ret = s8.replace('bao', '包')    # 把bao替换成包

print(s8)   # bao\_heihei\_nezha\_huhu 切记, 字符串是不可变对象. 所有操作都是产生新字符串返回

print(ret)  # 包\_heihei\_nezha\_huhu

ret = s8.replace('i', 'SB', 1)     # 把i替换成SB, 替换1个

print(ret)  # bao\_heSBhei\_nezha\_huhu

# 字符串切割

s9 = "bao\_heihei\_nezha\_huhu"

lst = s9.split("\_")     # 字符串切割, 根据\_进行切割

print(lst)

lst = ['周杰伦', '王力宏', '麻花藤']

s = "\_".join(lst)

print(s) # 周杰伦\_王力宏\_麻花藤

s10 = """诗人

学者

感叹号

渣渣"""

print(s10.split("\n"))  # 用\n切割

# 坑

s11 = "哈哈包黑黑呵呵包黑黑吼吼包黑黑"

lst = s11.split("银王")   # ['', '哈哈', '呵呵', '吼吼', ''] 如果切割符在左右两端. 那么一定会出现空字符串.深坑请留意

print(lst)

3. 格式化输出(了解)

# 格式化输出

s12 = "我叫%s, 今年%d岁了, 我喜欢%s" % ('sylar', 18, '周杰伦')  # 之前的写法

print(s12)

s12 = "我叫{}, 今年{}岁了, 我喜欢{}".format("周杰伦", 28, "周润发")    # 按位置格式化

print(s12)

s12 = "我叫{0}, 今年{2}岁了, 我喜欢{1}".format("周杰伦", "周润发", 28)     # 指定位置

print(s12)

s12 = "我叫{name}, 今年{age}岁了, 我喜欢{singer}".format(name="周杰伦", singer="周润发", age=28)     # 指定关键字

print(s12)

4. 查找

s13 = "我叫sylar, 我喜欢python, java, c等编程语言."

ret1 = s13.startswith("sylar")   # 判断是否以sylar开头

print(ret1)

ret2 = s13.startswith("我叫sylar")    # 判断是否以我叫sylar开头

print(ret2)

ret3 = s13.endswith("语言")   # 是否以'语言'结尾

print(ret3)

ret4 = s13.endswith("语言.")  # 是否以'语言.'结尾

print(ret4)

ret7 = s13.count("a")   # 查找"a"出现的次数

print(ret7)

ret5 = s13.find("sylar")    # 查找'sylar'出现的位置

print(ret5)

ret6 = s13.find("tory")     # 查找'tory'的位置, 如果没有返回-1

print(ret6)

ret7 = s13.find("a", 8, 22)  # 切片找

print(ret7)

ret8 = s13.index("sylar")   # 求索引位置. 注意. 如果找不到索引. 程序会报错

print(ret8)

5. 条件判断

# 条件判断

s14 = "123.16"

s15 = "abc"

s16 = "\_abc!@"

# 是否由数字组成

print(s14.isdigit())

print(s15.isdigit())

print(s16.isdigit())

# 是否由数字组成, 不包括小数点

print(s14.isdigit())

print(s15.isdecimal())

print(s16.isnumeric())  # 这个比较牛B. 中文都识别.

# 练习. 用算法判断某一个字符串是否是小数

s17 = "-123.12"

s17 = s17.replace("-", "")  # 替换掉负号

if s17.isdigit():

    print("是整数")

else:

    if s17.count(".") == 1 and not s17.startswith(".") and not s17.endswith("."):

        print("是小数")

    else:

        print("不是小数")

6. 计算字符串的长度

s18 = "我是你的眼, 我也是a"

ret = len(s18)  # 计算字符串的长度

print(ret)

注意: len()是python的内置函数. 所以访问方式也不一样. 你就记着len()和print()一样就行了

7. 迭代

    我们可以使用for循环来便利(获取)字符串中的每一个字符

    语法:

        for 变量 in 可迭代对象:

            pass

        可迭代对象: 可以一个一个往外取值的对象

s19 = "大家好, 我是VUE, 前端的小朋友们. 你们好么?"

# 用while循环

index = 0

while index < len(s19):

    print(s19[index])   # 利用索引切片来完成字符的查找

    index = index + 1

# for循环, 把s19中的每一个字符拿出来赋值给前面的c

for c in s19:

    print(c)

关于in

'''

    in有两种用法:

        1. 在for中. 是把每一个元素获取到赋值给前面的变量.

        2. 不在for中. 判断xxx是否出现在str中.

'''

print('VUE' in s19)

# 练习, 计算在字符串"I am sylar, I'm 5 years old, I have 2 dogs!"中出现了多少个数字

s20 = "I am sylar, I'm 14 years old, I have 2 dogs!"

count = 0

for c in s20:

    if c.isdigit():

        count = count + 1

print(count)

结论:

字符串索引和切片

索引从0开始

切片:

s[start: end: step]

start: 起始位置

end: 结束位置, 取不到

step: 步长.

1. upper(), 把字符串中所有的字母都变成大写. 主要使用在忽略大小写的时候用

2. strip(), 默认去掉左右两端的空白, 包括\n, \t, 空格.

3. replace(), 字符串替换

4. split(), 字符串切割. 得到字符串列表

5. join(), 把列表重新组合成字符串

6. startswith(), 判断是否以xxxx开头

7. find(), 查找xxxx

8. count(), 数数, 查看xxx出现的次数

9. isdigit(), 判断该字符串是否是由数字组成

10. len(), 字符串长度, 它是一个内置函数, 直接len(数据)即可

for 变量 in 可迭代对象:

循环体

**六. 基础数据类型list**

列表是python的基础数据类型之一 ,其他编程语言也有类似的数据类型. 比如JS中的数组, java中的数组等等. 它是以[ ]括起来, 每个元素用' , '隔开而且可以存放各种数据类型:

列表: 能装东西的东西

lst = [1, '哈哈', "吼吼", [1,8,0,"百度"], ("我","叫", "元", "组"), "abc", {"我叫":"dict字典"},{"我叫集合","集合"}]

列表相比于字符串. 不仅可以存放不同的数据类型. 而且可以存放大量的数据. 32位python可以存放:  536870912个元素, 64位可以存放:  1152921504606846975个元素.而且列表是有序的(按照你保存的顺序),有索引, 可以切片方便取值.

**6.1 列表的索引和切片**

列表和字符串一样也拥有索引:

lst = ["麻花藤", "王剑林", "马芸", "周鸿医", "向华强"]

print(lst[0])   # 获取第一个元素

print(lst[1])

print(lst[2])

lst[3] = "流动强"  # 注意. 列表是可以发生改变的. 这里和字符串不一样

print(lst)  # ['麻花藤', '王剑林', '马芸', '流动强', '向华强']

s0 = "向华强"

s0[1] = "美"  # TypeError: 'str' object does not support item assignment 不允许改变

print(s0)

     列表的切片:

lst = ["麻花藤", "王剑林", "马芸", "周鸿医", "向华强"]

print(lst[0:3])     # ['麻花藤', '王剑林', '马芸']

print(lst[:3])      # ['麻花藤', '王剑林', '马芸']

print(lst[1::2])    # ['王剑林', '周鸿医'] 也有步长

print(lst[2::-1])   # ['马芸', '王剑林', '麻花藤'] 也可以倒着取

print(lst[-1:-3:-2])    # 倒着带步长

**6.2 列表的相关操作**

1. 增加, 注意, list和str是不一样的. list可以发生改变. 所以直接就在原来的对象上进行了操作

lst = ["麻花藤", "林俊杰", "周润发", "周芷若"]

print(lst)

lst.append("wusir")

print(lst)

lst = []

while True:

    content = input("请输入你要录入的员工信息, 输入Q退出:")

    if content.upper() == 'Q':

        break

    lst.append(content)

print(lst)

lst = ["麻花藤", "张德忠", "孔德福"]

lst.insert(1, "刘德华")    # 在1的位置插入刘德华. 原来的元素向后移动一位

print(lst)

# 迭代添加

lst = ["王志文", "张一山", "苦海无涯"]

lst.extend(["麻花藤", "麻花不疼"])

print(lst)

2. 删除

    pop, remove, clear, del

lst = ["麻花藤", "王剑林", "李嘉诚", "王富贵"]

print(lst)

deleted = lst.pop()         # 删除最后一个

print("被删除的", deleted)

print(lst)

el = lst.pop(2)     # 删除2号元素

print(el)

print(lst)

lst.remove("麻花藤")   # 删除指定元素

print(lst)

# lst.remove("哈哈")    # 删除不存在的元素会报错

# # print(lst)

lst.clear()     # 清空list

print(lst)

# 切片删除

del lst[1:3]

print(lst)

3. 修改

    索引切片修改

# 修改

lst = ["得儿得儿", "太牛", "胡辣汤", "王者荣耀", "科科"]

lst[1] = "太污"   # 把1号元素修改成太污

print(lst)

lst[1:4:3] = ["麻花藤", "哇靠"]     # 切片修改也OK. 如果步长不是1, 要注意. 元素的个数

print(lst)

lst[1:4] = ["压了个嘿嘿龟儿子"]  # 如果切片没有步长或者步长是1. 则不用关心个数

print(lst)

4. 查询, 列表是一个可迭代对象, 所以可以进行for循环

for el in lst:

    print(el)

5. 其他操作

lst = ["小白", "小黑", "腾哥", "马总", "日天", "小白"]

c = lst.count("小白")     # 查询太白出现的次数

print(c)

lst = [1, 11, 22, 2]

lst.sort()          # 排序. 默认升序

print(lst)

lst.sort(reverse=True)  # 降序

print(lst)

lst = ["小白", "小黑", "腾哥", "马总", "日天", "白天"]

print(lst)

lst.reverse()

print(lst)

l = len(lst)    # 列表的长度

print(l)

6. 列表的嵌套, 一层一层的看就好.

lst = [1, "小白", "小黑黑", ["马虎疼", ["可口可乐"], "王剑林"], 'kounijiwa']

# 找到小黑黑

print(lst[2])

# 找到小白和小黑黑

print(lst[1:3])

# 找到小白的白字

print(lst[1][1])

# 将kounijiwa拿到. 然后首字母大写. 再扔回去

s = lst[4]

s = s.capitalize()

lst[4] = s

print(lst)

# 简写

lst[4] = lst[4].capitalize()

print(lst)

# 把小白换成小黑

lst[1] = lst[1].replace("白", "黑")

print(lst)

# 把马虎疼换成马化疼

lst[3][0] = lst[3][0].replace("虎", "化")

print(lst[3][0])

lst[3][1].append("雪碧")

print(lst)

7. range

range可以帮我们获取到一组数据. 通过for循环能分别获取到这些数据

for num in range(10):

    print(num)

for num in range(1, 10, 2):

    print(num)

for num in range(10, 1, -2):    # 反着来, 和切片一样

    print(num)

range最大的作用是可以循环出列表中每一个元素的索引

lst = ["周杰伦", "马虎疼", "疼不疼", "傻不傻"]

for i in range(len(lst)):

print(i, lst[i])

总结:

1. append() 增加数据, insert() 插入

2. remove() 删除数据, pop()

3. list[index] = value 修改数据

4. for循环 循环查看

for item in lst:

循环

for i in range(len(lst)):

i

lst[i]

**七. 基础数据类型tuple**

元组: 俗称不可变的列表.又被成为只读列表, 元组也是python的基本数据类型之一, 用小括号括起来, 里面可以放任何数据类型的数据,  查询可以. 循环也可以. 切片也可以. 但就是不能改.

tu = (1, "太白", "李白", "太黑", "怎么黑")

print(tu)

print(tu[0])

print(tu[2])

print(tu[2:5])  # 切片之后还是元组

# for循环遍历元组

for el in tu:

    print(el)

# 尝试修改元组

# tu[1] = "马虎疼"   # 报错 'tuple' object does not support item assignment

tu = (1, "哈哈", [], "呵呵")

# tu[2] = ["fdsaf"]   # 这么改不行

tu[2].append("麻花藤")     # 可以改了. 没报错

tu[2].append("王剑林")

print(tu)

关于不可变, 注意: 这里元组的不可变的意思是子元素不可变. 而子元素内部的子元素是可以变, 这取决于子元素是否是可变对象.

元组中如果只有一个元素. 一定要添加一个逗号, 否则就不是元组

tu = (1,)

print(type(tu))

**八. 基础数据类型dict**

**8.1 字典的简单介绍**

字典(dict)是python中唯一的一个映射类型.他是以{ }括起来的键值对组成. 在dict中key是唯一的. 在保存的时候, 根据key来计算出一个内存地址. 然后将key:value保存在这个地址中. 这种算法被称为hash算法, 所以, 切记, 在dict中存储的key:value中的key'必须是可hash的, 如果你搞不懂什么是可哈希, 暂时可以这样记, 可以改变的都是不可哈希的,

那么可哈希就意味着不可变. 这个是为了能准确的计算内存地址而规定的.

已知的可哈希(不可变)的数据类型: int, str, tuple, bool

不可哈希(可变)的数据类型: list, dict, set

语法 :

    {key1: value1, key2: value2....}

注意: key必须是不可变(可哈希)的. value没有要求.可以保存任意类型的数据

# 合法

dic = {123: 456, True: 999, "id": 1, "name": 'sylar', "age": 18, "stu": ['帅哥', '美女'], (1, 2, 3): '麻花藤'}

print(dic[123])

print(dic[True])

print(dic['id'])

print(dic['stu'])

print(dic[(1, 2, 3)])

# 不合法

# dic = {[1, 2, 3]: '周杰伦'}   # list是可变的. 不能作为key

# dic = {{1: 2}: "哈哈哈"}     # dict是可变的. 不能作为key

dic = {{1, 2, 3}: '呵呵呵'}    # set是可变的, 不能作为key

dict保存的数据不是按照我们添加进去的顺序保存的. 是按照hash表的顺序保存的. 而hash表不是连续的. 所以不能进行切片工作. 它只能通过key来获取dict中的数据

**8.2 字典增删改查和其他操作**

1. 增加

dic = {}

dic['name'] = '周润发'     # 如果dict中没有出现这个key, 就会新增一个key-value的组合进dict

dic['age'] = 18

print(dic)

# 如果dict中没有出现过这个key-value. 可以通过setdefault设置默认值

dic.setdefault('李嘉诚')   # 也可以往里面设置值.

dic.setdefault("李嘉诚", "房地产")    # 如果dict中已经存在了. 那么setdefault将不会起作用

print(dic)

2. 删除

ret = dic.pop("jay")

print(ret)

del dic["jay"]

print(dic)

# 清空字典中的所有内容

dic.clear()

3. 修改

dic = {"id": 123, "name": 'sylar', "age": 18}

dic1 = {"id": 456, "name": "麻花藤", "ok": "wtf"}

dic.update(dic1)    # 把dic1中的内容更新到dic中. 如果key重名. 则修改替换. 如果不存在key, 则新增.

print(dic)

print(dic1)

4. 查询

print(dic['name'])

# print(dic['sylar'])     # 报错

print(dic.get("ok"))

print(dic.get("sylar"))      # None

print(dic.get("sylar", "牛B"))   # 牛B

5. 循环

dic = {"id": 123, "name": 'sylar', "age": 18, "ok": "科比"}

for k in dic:

print(k)

print(dic.keys())   # dict\_keys(['id', 'name', 'age', 'ok']) 不用管它是什么.当成list来用就行

for key in dic.keys():

    print(key)

print(dic.values())     # dict\_values([123, 'sylar', 18, '科比']) 一样. 也当list来用

for value in dic.values():

    print(value)

print(dic.items())  # dict\_items([('id', 123), ('name', 'sylar'), ('age', 18), ('ok', '科比')]) 这个东西也是list. 只不过list中装的是tuple

for key, value in dic.items():  # ?? 这个是解构

    print(key, value)

# 解构

a, b = 1, 2

print(a, b)

(c, d) = 3, 4

print(c, d)

e, f = [1, 2, 3]    # 解构的时候注意数量必须匹配

print(e, f)

**8.3 字典的嵌套**

# 字典的嵌套

dic1 = {

    "name": "汪峰",

    "age": 18,

    "wife": {

        "name": '章子怡',

        "age": 28

    },

    "children": ['第一个毛孩子', '第二个毛孩子'],

    "desc": '峰哥不会告我吧. 没关系. 我想上头条的'

}

print(dic1.get("wife").get("name"))

print(dic1.["children"])

print(dic1.get("children")[1])

**九. 基础数据类型set**

set集合是python的一个基本数据类型. 一般不是很常用. set中的元素是不重复的.无序的.里面的元素必须是可hash的(int, str, tuple,bool), 我们可以这样来记. set就是dict类型的数据但是不保存value, 只保存key. set也用{}表示

set1 = {'1','周杰伦',2,True,[1,2,3]} # 报错

set2 = {'1','周杰伦',2,True,{1:2}}  # 报错

set3 = {'1','周杰伦',2,True,(1,2,[2,3,4])}  # 报错

s = {"周杰伦", "周杰伦", "周星星"}

print(s)

结果:

{'周星星', '周杰伦'}

使用这个特性.我们可以使用set来去掉重复

# 给list去重复

lst = [45, 5, "哈哈", 45, '哈哈', 50]

lst = list(set(lst))    # 把list转换成set, 然后再转换回list

print(lst)

最主要的操作: 去重复, 交,并,差

s1 = {"刘能", "赵四", "皮长山"}

s2 = {"刘科长", "冯乡长", "皮长山"}

# 交集

# 两个集合中的共有元素

print(s1 & s2)  # {'皮长山'}

print(s1.intersection(s2))  # {'皮长山'}

# 并集

print(s1 | s2)  # {'刘科长', '冯乡长', '赵四', '皮长山', '刘能'}

print(s1.union(s2))     # {'刘科长', '冯乡长', '赵四', '皮长山', '刘能'}

# 差集

print(s1 - s2)  # {'赵四', '刘能'} 得到第一个中单独存在的

print(s1.difference(s2))     # {'赵四', '刘能'}

**十. 深浅copy**

lst1 = ["金毛狮王", "紫衫龙王", "白眉鹰王", "青翼蝠王"]

lst2 = lst1

print(lst1)

print(lst2)

lst1.append("杨逍")

print(lst1)

print(lst2)

结果:

['金毛狮王', '紫衫龙王', '白眉鹰王', '青翼蝠王', '杨逍']

['金毛狮王', '紫衫龙王', '白眉鹰王', '青翼蝠王', '杨逍']

dic1 = {"id": 123, "name": "谢逊"}

dic2 = dic1

print(dic1) # {'id': 123, 'name': '谢逊'}

print(dic2) # {'id': 123, 'name': '谢逊'}

dic1['name'] = "范瑶"

print(dic1) # {'id': 123, 'name': '范瑶'}

print(dic2) # {'id': 123, 'name': '范瑶'}

对于list, set, dict来说, 直接赋值. 其实是把内存地址交给变量. 并不是复制一份内容. 所以. lst1的内存指向和lst2是一样的. lst1改变了, lst2也发生了改变

**10.1 浅拷贝**

lst1 = ["何炅", "杜海涛","周渝民"]

lst2 = lst1.copy()

lst1.append("李嘉诚")

print(lst1)

print(lst2)

print(id(lst1), id(lst2))

结果:

两个lst完全不一样. 内存地址和内容也不一样. 发现实现了内存的拷贝

lst1 = ["何炅", "杜海涛","周渝民", ["麻花藤", "马芸", "周笔畅"]]

lst2 = lst1.copy()

lst1[3].append("无敌是多磨寂寞")

print(lst1)

print(lst2)

print(id(lst1[3]), id(lst2[3]))

结果:

['何炅', '杜海涛', '周渝民', ['麻花藤', '马芸', '周笔畅', '无敌是多磨寂寞']]

['何炅', '杜海涛', '周渝民', ['麻花藤', '马芸', '周笔畅', '无敌是多磨寂寞']]

4417248328 4417248328

浅拷贝. 只会拷贝第一层. 第二层的内容不会拷贝. 所以被称为浅拷贝

**10.2 深拷贝**

import copy

lst1 = ["何炅", "杜海涛","周渝民", ["麻花藤", "马芸", "周笔畅"]]

lst2 = copy.deepcopy(lst1)

lst1[3].append("无敌是多磨寂寞")

print(lst1)

print(lst2)

print(id(lst1[3]), id(lst2[3]))

结果:

['何炅', '杜海涛', '周渝民', ['麻花藤', '马芸', '周笔畅', '无敌是多磨寂寞']]

['何炅', '杜海涛', '周渝民', ['麻花藤', '马芸', '周笔畅']]

4447221448 4447233800

都不一样了.  深度拷贝. 把元素内部的元素完全进行拷贝复制. 不会产生一个改变另一个跟着改变的问题

**十一. 知识点补充**

**11.1. 列表和字典循环的时候不能删除**

先看一段代码

lst = ["张无忌", "张翠山", "灭绝师太", "胡辣汤"]

for name in lst:

if name.startswith("张"):

lst.remove(name)

print(lst) # ["张翠山", "灭绝师太", "胡辣汤"]

为什么会这样呢? 原因是: 当删除掉第一个元素之后. 后面的元素就向前移动了一次. 而for循环还要向后走一次. 完美错过了"张翠山"这个元素. 那怎么办呢? 我们需要把要删除的内容先保存在一个新列表中, 然后循环这个新列表. 去删除原来的数据列表

lst = ["张无忌", "张翠山", "灭绝师太", "胡辣汤"]

new\_lst = []

for name in lst:

if name.startswith("张"):

new\_lst.append(name)

for name in new\_lst:

lst.remove(name)

print(lst) # ["灭绝师太", "胡辣汤"]

这样删除就没有问题了

结论: python中的列表和字典在循环的时候. 不能删除自身中的元素. 列表虽然不报错. 但是删不干净. 对于字典, 直接报错. 不让删. 解决方案都一样, 把要删除的内容保存在一个新列表中, 循环新列表, 删除老列表.

**11.2. is和==的区别**

a = [1, 2, 3]

b = [1, 2, 3]

print(a == b) # True

print(a is b) # False

结论:

== 判断的是内容. ==> 两个人长的是不是一样?

is 判断的是内存地址. ==> 两个人是不是同一个人

此结论不适合字符串. 这里涉及到小数据池的内容. 不用纠结. 暂时咱们先不用了解小数据池. 只需知道is和==的区别就好

**11.3 while...else**

    while 条件:

        循环体

    else: 循环在正常情况跳出之后会执行这里

index = 1

while index < 11:

    if index == 8:

        # break

        pass

    else:

        print(index)

    index = index+1

else:

print("你好")

注意: 如果循环是通过break退出的. 那么while后面的else将不会被执行, 只有在while条件判断是假的时候才会执行这个else

i = 0

while i < 10:

if i == 7:

break

print(i)

else:

print("么么哒") # 这句话不会被打印, 因为i数到7, 就会break. break的是while...else这个整体.

pass: 不表示任何内容. 为了代码的完整性.  占位而已

**十二. 文件操作**

**1.1. 初识文件操作**

    使用python来读写文件是非常简单的操作. 我们使用open()函数来打开一个文件, 获取到文件句柄. 然后通过文件句柄就可以进行各种各样的操作了. 根据打开方式的不同能够执行的操作也会有相应的差异.

    打开文件的方式: r, w, a, r+, w+, a+, rb, wb, ab, r+b, w+b, a+b 默认使用的是r(只读)模式

**1.2. 只读操作(r, rb)**

f = open("护士少妇嫩模.txt",mode="r", encoding="utf-8")

content = f.read()

print(content)

f.close()

需要注意encoding表示编码集. 根据文件的实际保存编码进行获取数据, 对于我们而言. 更多的是utf-8.

    rb. 读取出来的数据是bytes类型, 在rb模式下. 不能选择encoding字符集.

f = open("护士少妇嫩模.txt",mode="rb" )

content = f.read()

print(content)

f.close()

结果:

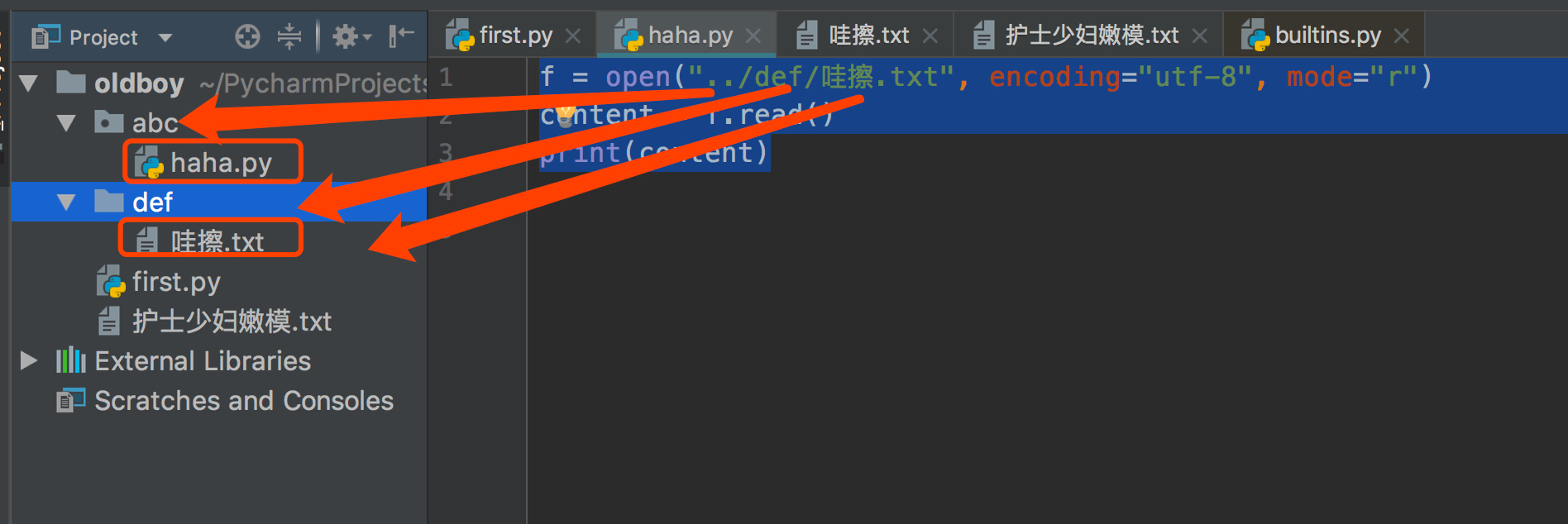
b'\xe6\xaf\x85\xe5\x93\xa5, \xe5\xa4\xaa\xe7\x99\xbd, wuse\n\xe5\x91\xb5\xe5\x91\xb5\n\xe6\x97\xa5\xe5\xa4\xa9'

    rb的作用: 在读取非文本文件的时候. 比如读取MP3. 图像. 视频等信息的时候就需要用到rb. 因为这种数据是没办法直接显示出来的.  在后面我们文件上传下载的时候还会用到. 还有. 我们看的直播. 实际上都是这种数据.

绝对路径和相对路径:

绝对路径:从磁盘根目录开始一直到文件名.

相对路径:同一个文件夹下的文件. 相对于当前这个程序所在的文件夹而言. 如果在同一个文件夹中. 则相对路径就是这个文件名. 如果在上一层文件夹. 则要../



我们更推荐大家使用相对路径. 因为在我们把程序拷贝给别人使用的时候. 直接把项目拷贝走就能运行. 但是如果用绝对路径. 那还需要拷贝外部的文件.

读取文件的方法:

read()  将文件中的内容全部读取出来. 弊端: 占内存. 如果文件过大.容易导致内存崩溃

f = open("../def/哇擦.txt", mode="r", encoding="utf-8")

content = f.read()

print(content)

结果:

友谊地久天长,

爱一点,

可惜我是水瓶座

一生中最爱

2. read(n) 读取n个字符. 需要注意的是. 如果再次读取. 那么会在当前位置继续去读而不是从头读, 如果使用的是rb模式. 则读取出来的是n个字节

f = open("../def/哇擦.txt", mode="r" encoding="utf-8")

content = f.read(3)

print(content)

结果:

友谊地

f = open("../def/哇擦.txt", mode="rb")

content = f.read(3)

print(content)

结果:

b'\xe5\x8f\x8b'

f = open("../def/哇擦.txt", mode="r", encoding="utf-8")

content = f.read(3)

content2 = f.read(3)

print(content)

print(content2)

结果:

友谊地

久天长

3. readline() 一次读取一行数据, 注意: readline()结尾, 注意每次读取出来的数据都会有一个\n 所以呢. 需要我们使用strip()方法来去掉\n或者空格

f = open("../def/哇擦.txt", mode="r", encoding="utf-8")

content = f.readline()

content2 = f.readline()

content3 = f.readline()

content4 = f.readline()

content5 = f.readline()

content6 = f.readline()

print(content)

print(content2)

print(content3)

print(content4)

print(content5)

print(content6)

结果:

友谊地久天长,

爱一点,

可惜我是水瓶座

一生中最爱

4. readlines()将每一行形成一个元素, 放到一个列表中. 将所有的内容都读取出来. 所以也是. 容易出现内存崩溃的问题.不推荐使用

f = open("../def/哇擦.txt", mode="r", encoding="utf-8")

lst = f.readlines()

print(lst)

for line in lst:

    print(line.strip())

5. 循环读取. 这种方式是组好的. 每次读取一行内容.不会产生内存溢出的问题.

f = open("../def/哇擦.txt", mode="r", encoding="utf-8")

for line in f:

    print(line.strip())

f.close()

注意: 读取完的文件句柄一定要关闭   f.close()

**1.3. 写模式(w, wb)**

写的时候注意. 如果没有文件. 则会创建文件, 如果文件存在. 则将原件中原来的内容删除, 再写入新内容

f = open("小娃娃", mode="w", encoding="utf-8")

f.write("金毛狮王")

f.flush()    # 刷新. 养成好习惯

f.close()

尝试读一读

f = open("小娃娃", mode="w", encoding="utf-8")

f.write("金毛狮王")

f.read()    # not readable 模式是w. 不可以执行读操作

f.flush()

f.close()

wb模式下. 可以不指定打开文件的编码. 但是在写文件的时候必须将字符串转化成utf-8的bytes数据

f = open("小娃娃", mode="wb")

f.write("金毛狮王".encode("utf-8"))

f.flush()

f.close()

**1.4. 追加(a, ab)**

    只要是a或者ab, a+ 都是在文件的末尾写入. 不论光标在任何位置.

    在追加模式下. 我们写入的内容会追加在文件的结尾.

f = open("小娃娃", mode="a", encoding="utf-8")

f.write("麻花藤的最爱")

f.flush()

f.close()

    ab模式自己试一试就好了

**1.5. 读写模式(r+, r+b)**

    对于读写模式. 必须是先读. 因为默认光标是在开头的. 准备读取的. 当读完了之后再进行写入. 我们以后使用频率最高的模式就是r+

正确操作是:

f = open("小娃娃", mode="r+", encoding="utf-8")

content = f.read()

f.write("麻花藤的最爱")

print(content)

f.flush()

f.close()

结果:

正常的读取之后, 写在结尾

错误操作:

f = open("小娃娃", mode="r+", encoding="utf-8")

f.write("哈哈")

content = f.read()

print(content)

f.flush()

f.close()

结果: 将开头的内容改写成了"哈哈", 然后读取的内容是后面的内容.

所以记住: r+模式下. 必须是先读取. 然后再写入

还有一些其他的带b的操作. 就不多赘述了. 就是把字符换成字节. 仅此而已

**1.6. 修改文件以及另一种打开文件的方式(重点)**

    文件修改: 只能将文件中的内容读取到内存中, 将信息修改完毕, 然后将源文件删除, 将新文件的名字改成老文件的名字.

# 文件修改

import os

with open("小娃娃", mode="r", encoding="utf-8") as f1,\

     open("小娃娃\_new", mode="w", encoding="UTF-8") as f2:

    content = f1.read()

    new\_content = content.replace("冰糖葫芦", "大白梨")

    f2.write(new\_content)

os.remove("小娃娃")    # 删除源文件

os.rename("小娃娃\_new", "小娃娃")     # 重命名新文件

弊端: 一次将所有内容进行读取. 内存溢出. 解决方案: 一行一行的读取和操作

import os

with open("小娃娃", mode="r", encoding="utf-8") as f1,\

     open("小娃娃\_new", mode="w", encoding="UTF-8") as f2:

    for line in f1:

        new\_line = line.replace("大白梨", "冰糖葫芦")

        f2.write(new\_line)

os.remove("小娃娃")    # 删除源文件

os.rename("小娃娃\_new", "小娃娃")     # 重命名新文件

文件练习题:

文件a1.txt内容

序号 部门 人数 平均年龄 备注

1 python 30 26 单身狗

2 Linux 26 30 没对象

3 运营部 20 24 女生多

.......

通过代码，将其构建成这种数据类型：

[{'序号':'1','部门':Python,'人数':30,'平均年龄':26,'备注':'单身狗'},......]

使用的知识点:

1. 文件读取

2. 字符串

3. 字典

4. 列表

# 读取规则的文件

f = open("g.txt", mode="r", encoding="utf-8")

head\_str = f.readline()

# 把头处理成列表

head\_list = head\_str.split() # 默认的split()用空白切割

# print(head\_list)

lst = []

for line in f:

line = line.strip() # 去掉左右两端空白

data\_list = line.split() # 切割

# print(data\_list)

dic = {}

for i in range(len(head\_list)): # 拿到索引

dic[head\_list[i]] = data\_list[i] # 向字典中填充数据

lst.append(dic)

print(lst)

f.close()

**重点整合**

1. 格式化输出:

%s 表示 占位 字符串

%d 表示 占位 数字

f"{变量}"

2. 简单运算符

and 并且, 左右两端同时为真. 结果才是真

or 或者, 左右两端有一个是真, 结果就是真

not 非, 非真既假, 非假既真

顺序: () => not => and => or

3. 编码初识(ascii,unicode,utf-8,gbk等历史)以及bytes

ascii : 8bit, 1byte, 主要存放的是英文, 数字, 特殊符号

gbk: 16bit, 2byte, 主要存放中文和亚洲字符. 兼容ascii

unicode: 16bit和32bit两个版本. 平时我们用的是16bit这个版本. 全世界所有国家的文字信息. 缺点: 浪费空间(传输和存储)

utf-8 : 可变长度unicode, 英文: 8bit, 欧洲文字: 16bit, 中文24bit. 一般数据传输和存储的时候使用.

encode() 编码. 字符串转化成字节bytes.

decode() 解码. 把字节bytes转化成字符串.

4. 基础数据类型bool

所有表示空的东西都是False

5. 基础数据类型str

索引, 从0开始

切片: [start: end: step] 顾头不顾尾

字符串不论做什么操作. 原字符串是不可以发生改变的. 都是返回一个新的字符串

upper(), 把字符串中所有的字母都变成大写. 主要使用在忽略大小写的时候用

strip(), 默认去掉左右两端的空白, 包括\n, \t, 空格.

replace(), 字符串替换

split(), 字符串切割. 得到字符串列表

join(), 把列表组合成一个字符串

startswith(), 判断是否以xxxx开头

find(), 查找xxxx

count(), 数数, 查看xxx出现的次数

isdigit(), 判断该字符串是否是由数字组成

len() 字符串长度 内置函数

for c in s:

循环字符串

6. 基础数据类型list

append() 追加

insert() 插入

remove() 删除

pop() 删除元素

list[index] = value 修改

for item in list:

item 元素

for i in range(len(列表)):

i 索引

列表[i] 元素

7. 基础数据类型tuple

只读列表.

如果tuple中只有一个元素. 那么必须在末尾添加一个 逗号

8. 基础数据类型dict

字典是以k:v的形式存储数据的.

k必须是可哈希的数据类型

dict[k]

dict.get(k)

dict.pop(k) 删除

for k in dict:

pass

拿到字典中的每一个k和v

for k, v in dict.items():

pass

10.深浅copy

= 并没有创建新的列表, 左右两端使用的是同一个列表

浅拷贝: 只拷贝第一层内容

深拷贝: 内外层数据全部拷贝

11. 知识点补充

is 判断的是左右两端的内存地址, is None, is not None

== 判断左右两端的内容是否一致

12. 文件操作:

f = open(文件, mode="模式" encoding="编码")

模式:

r: 只读

w: 只写

a: 追加写

+: 扩展

b: 字节(非文本文件)

读取文件最好的方案

with open() as f:

for line in f:

xxxxx

修改文件:

1. 创建一个文件副本.

2. 把源文件中的内容读取到内存.

3. 然后在内存中进行修改.

4. 修改之后保存在文件副本中.

5. 把源文件删除

6. 把文件副本更改名称为源文件的名称

**练习题**

dic1 = {

    'name':['jay',2,3,5],

    'job':'teacher',

    'school':{'sylar':['python1','python2',100]}

    }

1，将name对应的列表追加一个元素'太黑'。

2，将name对应的列表中的jay首字母大写。

3，school对应的字典加一个键值对'tory':['linux','java']。

4，将oldboy对应的字典中的dic1对应的列表中的job删除。

使用for循环对s="abcdefg"进行循环，每次打印的内容是每个字符加上sb， 例如：asb, bsb，csb,...gsb。

计算用户输入的内容中有几个整数（以个位数为单位）。

如：content = input("请输入内容：") # 如fhdal234slfh98769fjdla

制作趣味模板程序需求：等待⽤户输⼊名字、地点、爱好，根据⽤户的名字和爱好进⾏任意现实

如：敬爱可亲的xxx，最喜欢在xxx地⽅⼲xxx

写代码，完成下列需求：

用户可持续输入（用while循环），用户使用的情况：

输入A，则显示走大路回家，然后在让用户进一步选择：

是选择公交车，还是步行？

选择公交车，显示10分钟到家，并退出整个程序。

选择步行，显示20分钟到家，并退出整个程序。

输入B，则显示走小路回家，并退出整个程序。

输入C，则显示绕道回家，然后在让用户进一步选择：

是选择游戏厅玩会，还是网吧？

选择游戏厅，则显示 ‘一个半小时到家，爸爸在家，拿棍等你。’并让其重新输入A，B,C选项。

选择网吧，则显示‘两个小时到家，妈妈已做好了战斗准备。’并让其重新输入A，B,C选项。

1. 请用代码实现：

li = ["jolin", "eric", "rain"]

利用下划线将列表的每一个元素拼接成字符串"jolin\_eric\_rain"

2. 利用for循环和range打印出下面列表的索引和元素。

li = ["jolin", "sep", "kongkong", "tony", "kalido"]

3. 利用for循环和range 找出50以内能被3整除的数，并将这些数存入到一个新列表中。

4. 利用for循环和range，将1-30的数字一次添加到一个列表中，并循环这个列表，将能被3整除的数改成\*

开发敏感词语过滤程序，提示用户输入评论内容，如果用户输入的内容中包含特殊的字符：

敏感词列表 li = ["苍老师", "东京热", "武藤兰", "波多野结衣"]

则将用户输入的内容中的敏感词汇替换成等长度的\*（苍老师就替换\*\*\*），并添加到一个列表中；

如果用户输入的内容没有敏感词汇，则直接添加到上述的列表中。

av\_catalog = {

"欧美":{

"www.youporn.com": ["很多免费的,世界最大的","质量一般"],

"www.pornhub.com": ["很多免费的,也很大","质量比yourporn高点"],

"letmedothistoyou.com": ["多是自拍,高质量图片很多","资源不多,更新慢"],

"x-art.com":["质量很高,真的很高","全部收费,屌丝请绕过"]

},

"日韩":{

"tokyo-hot":["质量怎样不清楚,个人已经不喜欢日韩范了","verygood"]

},

"大陆":{

"1024":["全部免费,真好,好人一生平安","服务器在国外,慢"]

}

}

a,给此 ["很多免费的,世界最大的","质量一般"]列表第二个位置插入一个 元素：'量很大'。

b,将此 ["质量很高,真的很高","全部收费,屌丝请绕过"]列表的 "全部收费,屌丝请绕过" 删除。

c,将此 ["质量很高,真的很高","全部收费,屌丝请绕过"]列表的 "全部收 费,屌丝请绕过" 删除。

d,将此["质量怎样不清楚,个人已经不喜欢日韩范了","verygood"]列表的 "verygood"全部变成大写。

e,给 '大陆' 对应的字典添加一个键值对 '1048' :['一天就封了']

f,删除此"letmedothistoyou.com": ["多是自拍,高质量图片很多","资源不多,更新慢"]键值对。

g,给此["全部免费,真好,好人一生平安","服务器在国外,慢"]列表的第一个元素，加上一句话：'可以爬下来'

有字符串"k:1|k1:2|k2:3|k3:4" 处理成字典 {'k':1,'k1':2....}

有如下值li= [11,22,33,44,55,66,77,88,99,90]，

将所有大于 66 的值保存至字典的第一个key中，

将小于 66 的值保存至第二个key的值中。

即： {'k1': 大于66的所有值列表, 'k2': 小于66的所有值列表}

**作业题:**

**1. 小试牛刀**

车牌区域划分, 现给出以下车牌. 根据车牌的信息, 分析出各省的车牌持有量.

cars = ['鲁A32444', '鲁B12333', '京B8989M', '黑C49678', '黑C46555', '沪B25041', '黑C34567']

locations = {'沪': '上海', '京': '北京', '黑': '黑龙江', '鲁': '山东', '鄂': '湖北', '湘': '湖南'}

结果:

{"上海":1, "北京":1, "黑龙江":3, "山东":2}

**2. 真正的挑战**

数据结构：

goods = [

{"name": "电脑", "price": 1999},

{"name": "鼠标", "price": 10},

{"name": "游艇", "price": 20},

{"name": "美女", "price": 998},

......

]

提示:

shoppingCart = [

{"name": "电脑", "price": 1999, "count":1},

{"name": "美女", "price": 998, "count":3},

]

功能要求：

1、启动程序后，输入用户名密码后，让用户输入余额，

2、然后打印商品列表

页面显示 序号 + 商品名称 + 商品价格，如：

1 电脑 1999

2 鼠标 10

…

n 购物车结算

2、用户输入商品编号或n

3、用户选择商品后，检测余额是否够，够就直接扣款，不够就提醒

4、用户输入n结束购买过程.打印出该用户购买的商品信息, 数量, 单价. 以及余额

**超纲算法题(相当于脑筋急转弯, 千万别当真):**

    本题只是为了练习思维逻辑用的. 和作业无关.

请使用循环语句完成以下图案的打印:

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

上期超纲算法题答案:

计算: 1-3+5-7+9…99 = ?

i = 1

s = 0

fu = 1

while i <= 99:

s = s + i \* fu

fu = -fu # 本题灵魂

i += 2 # i 是奇数