Python 基础

1.python 的运算符

1.1 算数运算符

+

-

*：一个小妙招。例如输出 100 个“！%“。可以写“!%” * 100。拼接字符串。

/

//：整除

%：取余

**：幂运算

1.2 比较运算符

==：等于

!=：不等于

>

<

>=

<=

1.3 逻辑运算符

and，or，not

2.Pycharm 安装、快捷方式、缩进、统一增加#

2.1linux 版本安装：

tar -zxvf pycharm-edu-3.5.1.tar.gz

#解压后直接可以使用了，建议移动解压文件夹到/opt 目录下，这样所有用户都可以用了。

pycharm_path/pycharm.sh

#找到文件夹中的 pycharm.sh 文件执行即可使用

2.2 软连接:

在 ubuntu 中，用户程序启动的快捷方式都放在/usr/share/applications 目录下。因此我们可以建立一个软连接放在此目录下即可。

#ln -s [pycharm/pycharm.sh 路径] /usr/share/applications/pycharm.desktop

2.3 Tab 增加 Shift+Tab 减少缩进

2.4 Ctrl+/ 可以为选中的代码统一加上注释‘#’

3.python 数据类型

字符串（str）、列表、元组、字典

整型（int）、浮点型（float）、布尔型（bool）、复数型（complex）

3.1type(name)

```
#  查看变量类型
```

3.2 列表

```
列表定义：[1, 2, ‘a’, ‘&’]
列表索引从 0 开始
```

3.2.1 列表排序

```
列表.sort()升序排序
列表.sort(reverse = T)降序
列表.reverse()反向，逆序
```

3.2.2 列表增加

```
列表.insert(索引，数据) 在指定位置后增加元素
列表.append(数据)  在末尾增加元素
列表.extend(列表 2)  将列表 2 追加到列表末尾
```

3.2.3 列表修改

```
列表[索引]= 数据 修改指定索引的数据
```

3.2.4 列表删除

```
del 列表[索引]  删除指定索引的数据
列表.remove(数据)  删除第一次出现的指定数据
列表.pop 删除末尾的数据
列表.clear  清空列表
```

3.2.5 列表统计

```
len(列表)  统计列表长度
列表.count(数据)  统计数据在列表中出现的次数
```

3.3 字符串

3.3.1 字符串判断

```
'hello'.isidentifier()  #判断 hello 是不是合法字符串
'\t'.isspace()  #判断\t 是不是空白字符串（空白字符串有空格，回车，水平制表符）
'abc'.isalpha()  #判断字符串是不是全部由字母组成
'1234'.isdecimal()  #判断字符串是否全部由十进制数字组成
'123'.isnumeric()  #判断字符串是否全部由数字组成，罗马数字，汉字数字也是数字，Ⅱ，二
'123abc'.isalnum()  #判断字符串是否全部由数字和字母组成，汉字属于字母
```

3.3.2 字符串内容对齐

```
a='python'
#居中对齐,center
print(a.center(8,'*'))  #*python*,第一参数指定宽度，第二个参数指定填充符号，默认空格
#左对齐,ljust
print(a.ljust(8,'*'))  #python**,第一参数指定宽度，第二个参数指定填充符号，默认空格
#右对齐,rjust
print(a.rjust(8,'*'))  #**python,第一参数指定宽度，第二个参数指定填充符号，默认空格
```

3.3.3 字符串分割

```
a='python hello|world'  #使用指定的分隔符从左侧开始分割字符串，split
lst=a.split()  #默认采用空格作为分隔符,分割结果为列表
lst=a.split(seq='|')  #使用 seq=指定分隔符，分割结果为列表
lst=a.split(seq=' ',maxsplit=1)  #使用 seq=指定分隔符，使用 maxsplite=指定最大分割次数，达到最大分割次数后，后面的字符串将作为一个整体
#使用指定分隔符从右侧开始分割字符串，rsplit,使用方法和 splite 完全一样，只是从右侧开始分割而已
```

3.3.4 字符串合并

```
#join（）将列表或者元组中的字符串合并成一个新字符串,注意必须是列表或者元组
a=['hello','python']
print('|'.join(a))  #使用|作为连接符，把 a 列表中的字符串连接起来，注意‘|’.join，是一个点
print(' '.join(a))  #使用空格作为连接符，把 a 列表中的字符串连接起来
print('*'.join('python'))  #p*y*t*h*o*n
```

3.3.5 字符串大小写转换

```
a='PYTHON'    b='python'    c='PYThon'
b1=b.upper()    #upper,把所有字母全部转换为大写 #转成大写后产生一个新字符串
a1=a.lower()    #把所有字母全部转换为小写转成大写，产生一个新字符串
c.swapcase()  #把大写转小写，把小写转大写
c.capitalize() #capitalize,把第一个字符转换为大写，其余字符全部小写
```

3.3.6 字符串查找

```
s='hello,hello'
print(s.find('lo'))      #查找 lo 第一次出现的位置，找不到返回-1
print(s.rfind('lo'))     #查找 lo 最后一次出现的位置，找不到返回-1
```

3.3.7 字符串切片

```
a='pythonhello'
b=a[:5]  #从 0 号位开始切，切到 4 号位。pytho
e=a[1:6:2]  #[start:stop:step]  step 就是等差值，可以截取 2，4，6 号位的字符，不必连续截取
f=a[::-1]  #step 为负数时，则从末尾开始截取
```

3.3.8 字符串替换

```
a='hello world'
print(a.replace('hello', 'java'))  #用 java 去替换 python，第一个数是原字符串，第二个参数是要替换成的字符串
b='hello python python python'
print(b.replace('python', 'java', 2))  #仅替换前两个 python，第一个数是原字符串，第二个参数是要替换成的字符串，第三个参数是替换几次
####特别注意，replace 这种方法替换会产生一个新字符串，因此会伴随字符串末尾的\n 符难以去除
re.sub('原字符串'，'新字符串'，变量名)  # name = re.sub('.bam\n', '', i)  这种方法会直接修改原字符，不会新建字符串。
```

3.3.9 格式化字符串

```
#格式化字符串--方法三：使用 f 声明格式化字符串
print(f'我叫{name},今年{age}岁了')   #注意 f 的位置

#固定字符串宽度，例如固定整数的宽度
print('%10d' % 99)  #'' % 99 是固定格式，%d 表示整数，%10d 表示代指的整数宽度设为占 10 个字符，前面多了 8 个空格
print('%.3f' % 3.1415926)  #保留三位小数，%f 代指小数，%.3f 保留三位小数
print('%10.3f' % 3.1415926)  #同时表示宽度和精度，宽度 10，精度保留三位小数
```

3.4 元组

定义元组：tuple = (1, 2, 'a')
定义完就不能修改，其他的和列表基本一致。

3.5 字典

键值对。键必须是唯一的（键就类似于索引一样）
使用{}定义，元素是无序的，列表是有序的。
字典.key() 返回所有键组成的一个列表
字典.vakues() 返回所有值组成的一个列表
字典.items() 返回所有键值对（key，values）元组组成的一个列表（列表中的每个元素是一个元组，一个键值对）
字典['key'] #字典取值
字典['key'] = 18 #字典增加键值对，如果键已经存在，则变成修改键值对
字典.pop['key'] #删除键值对

4 python 内置公共方法

字符串，列表，元组，字典都可以统一使用的方法就是公共方法。

4.1 内置公共函数

len(item) 统计变量中元素个数
del(item) 删除指定变量
max(item) 返回容器中元素最大值，若果是字典，只针对 key
min(item) 返回容器中元素最小值，若果是字典，只针对 key
#字符串比较时：'a' > 'A' > '0' ACSⅡ码排序

4.2 切片操作

变量[start:stop:step] 支持字符串，列表，元组

4.3 公共运算符

+ : [1,2] + [3,4]，会创建新列表
* : ['HI'] * 5
in : 3 in [1,2,3]
not in
> < >= <= ==.都支持字符串，列表，元组

4.4 完整 for 循环

```
for x in range(1,10):
    执行 1
else:
    执行 2
###else 只有在遍历完全后才会执行
```

5.常用循环与函数

5.1 input 函数

```
# filepath = input("请输入文件路径：")
# number = int(input("请输入数字：")) 可以修改变量类型，默认都是字符串。
# number = float(input('请输入另一个数字：'))
```

5.2 if 语句

```
if [要判断的条件]:
    条件成立时执行内容
else:
    条件不成立执行内容
```

5.3 if elif 语句

```
if [判断条件 1]:
    执行内容 1
elif [判断条件 2]:
    执行内容 2
…
else:
    执行内容 n
```

5.4 if 嵌套

```
if [判断条件 1]:
    if [判断条件 2]:
        执行内容 1
    else:
        执行内容 2
else:
    执行内容 3
```

5.5 import 导入 python 包

5.5.1 随机数包 random

```
# import random
# 输入 random. 再按 Tab 键，可以把 random 包中所有工具全部列出来。
# random.randint(a, b)  返回[a,b]区间内的整数
```

5.6 循环 while

5.6.1 while 基本循环

```
i = 0
while 100 > i:
    执行内容
    i = i + 1
```

5.6.2 while break 循环

```
    i = 0
    while [判断条件 1]:
        i = i + 1
        执行内容 1
        if [判断条件 2]:
            执行内容 2
        else：
            break
    print(‘over’)
```

5.6.3 while continue 循环

```
continue 会直接此次循环跳出来并重新回到 while 循环条件判断。
i = 0
while [条件 1]:
    执行内容 1
    if [条件 2]:
        i = i + 1
        continue #特别注意 continue 后循环计数是否正常，别死循环了
print(‘over’)
```

5.6.4 while 嵌套

```
while 条件 1：
    while 条件 2：
        执行内容 2
    执行内容 1
print(‘over’)
```

5.7 print 函数 print(‘*’, end = ‘’)指定内容末尾符号

```
#end = ‘’可以指定输出内容末尾是否加什么东西。默认加换行符。
```

6. 转义字符

\\ ： 反斜杠

\’ ： 单引号

\” ： 双引号

\n ： 换行符

\t ： 制表符

\r ： 回车

7.函数

7.1 定义函数

7.1.1 具体格式

```
def num():
    执行
    return (x,y)    #可以使用元组来接收多个函数返回值
gl_x, gl_y = num()    #可以使用多个变量接受函数的多个返回值，要求两者数量一致
```

7.1.2 Pycharm 调试工具

```
F8 单步执行
F7 遇到函数时，F7 可以进入函数内部单步执行
```

7.1.3 函数的标准注释文档

在函数定义的下方，使用连续三对引号进行注释。并且可以在函数使用的地方 Ctrl+Q 快速查看函数注释信息

7.2 函数传参

```
def sum_num(num1, num2):
    result = num1 + num2
    print(f’{num1} + {num2} = {result}’)
sum_num(10, 20)
```

7.2.1 形参和实参

形参：定义函数时传递的参数，为了让函数可以接受外部数据而设置的。例如 num1，num2

实参：实际函数调用时给的参数，例如 10，20

7.2.2 函数返回值 return

```
def sum_num(num1, num2):
    return num1+num2
result = sum_num(10, 20)  #调用函数，使用一个变量接收结果即可。
```

这样可以在函数外部知道函数返回结果，否则只有函数内部知道结果，我们无法利用结果。

其他小技巧

1. a,b = b,a

交换赋值，利用的是元组的性质，右边相当于（b,a），括号省略了，然后就是利用元组性质，同时给多个变量赋值