笔记

> 版权：智泊AI
>
> 作者：Jeff

一、格式化字符串：

f-string占位符（重点掌握）

Python 3.6以后引入了一个新的格式化字符串的方法：f-string（formatted string），它可以直接把变量写在字符串中，使得格式化的字符串看起来很直观

```
a = "hello"
b = "ikun哥"
print(a, b)

# 需求：打印内容 "hello,你是ikun哥吗？"

# 方式1：  不推荐
print(a, ",你是", b, "吗？", sep='')

# 方式2：f-string语法，推荐
print(f'{a},你是{b}吗？')
```

百分号占位符

在Python中，占位符是一种特殊的标记或占据字符串中的位置，用于表示在运行时将某个值插入到这个位置。占位符通常以百分号（%）开头，后跟一个字母或字母组合，表示不同的数据类型。Python中常用的占位符有：

- %s：字符串占位符，用于插入字符串
- %d：整数占位符，用于插入整数
- %f：浮点数占位符，用于插入浮点数
- %%：百分号占位符，用于插入百分号符号

```
# 占位符：
#   %s : 字符串
#   %d : 整数
#   %f : 小数 , %.2f表示保留2位小数

name = "杰伦"
age = 40
salary = 10000.4567

print('大家好！我是杰伦,今年40岁，我年薪10000.4567万')

# 百分号占位符写法：
```

```
print('大家好！我是%s,今年%d岁，我年薪%.2f万' % (name, age, salary))

# f-string写法：
print(f'大家好！我是{name},今年{age}岁，我年薪{salary}万')
```

花括号占位符（了解）

花括号占位符是从Python 3.6版本开始引入的一种新语法

```
name = "Jack"
age = 25
print("My name is {} and I am {} years old.".format(name, age))

# 也可以定义名称a,b,然后在format中给a和b传值
print("My name is {a} and I am {b} years old.".format(a=name, b=age))
print("My name is {b} and I am {a} years old.".format(b=name, a=age))
```

练习：

```
# 请输入您的姓名，年龄，身高，体重，期中姓名是字符串，年龄是整数，身高和体重是小数类型，
# 要求分别使用上面3种占位符方式输出内容："大家好，我是xxx，今年xxx岁，我身高是xx.xcm，体重是xx.xkg"
#       例如："大家好，我是Jack，今年25岁，我身高是177.5cm，体重是75.2kg"
```

二、Python中的数据类型介绍

> 计算机： 顾明思议就是可以做数学运算的机器, 因此计算机理所当然的可以处理各种数据, 但是计算机能处理的远远不止数值，还有文本,图形,音频,视频网页等各种数据，不同数据需要定义不同的数据类型
>
> int, float【数字类型：整型int，浮点型[小数]float，复数类型complex】， 如： 100，3.14
>
> str【字符串】， 如："hello"，'张三'
>
> bool【布尔类型】： True真（1）， Flase假（0）
>
> NoneType【空值】: None
>
> list【列表】 类似c语言的数组array， 如： [1, 2, 3]
>
> tuple【元组】 不可改变的列表，如： (1, 2, 3)
>
> dict【字典】由键值对组成的，如： {"name": "张三"，"age": 30}
>
> set【集合】(了解)，如： {1, 2, 3}
>
> bytes【字节】二进制， 如：b'hello'

三、表达式和运算符

1.表达式

> 操作数和运算符组成, 比如: 1 + 1
>
> 作用: 表达式可以求值，也可以给变量赋值

2.运算符【掌握】

2.1. 算术运算符

> `+  -  *【乘法】  /【除法】  %【求余，取模】  **【求幂,次方】  //【取整】`
>
> 代码演示：

```
num1 = 5
num2 = 3
print(num1 + num2)
print(num1 - num2)
print(num1 * num2)
print(num1 / num2)  #浮点型：1.6666666666666667    默认精度16位
print(num1 % num2)  #2
print(num1 ** num2) # 5的3次方
print(num1 // num2) # 获取浮点数的整数部分(向下取整)

# 除了+和-之外，其他的算术运算符都是相同的优先级
# 出现优先级，解决办法使用括号
print((2 ** 5) * 3)
```

2.2. 赋值运算符

> 简单赋值运算符：= 给一个变量进行赋值
>
> 复合赋值运算符：+= -= %= /= ... 给一个变量进行赋值，同时给变量进行相应的运算
>
> 代码演示：

```
# 简单
num1 = 10
# 注意：在赋值运算符中，先计算等号右边的表达式，然后将计算的结果赋值给等号左边的变量
num2 = num1 + 10
print(num2)

# 复合运算符
num3 = 10
num3 += 100   #等价于num3 = num3 + 100
print(num3)
```

2.3. 关系【条件，比较】运算符

> 作用：比较大小，得到结果为布尔值【如果表达式成立，则返回True，如果不成立，则返回False】
>
> ```
> >     <     >=     <=     ==【等号】     !=  【不等于】
> ```
>
> 使用场景：if语句，循环
>
> 代码演示：
>
> ```
> x = 3
> y = 5
> print(x > y)    #False
> print(x < y)
>
> print(x == y)
> print(x != y)
>
> print(x >= y)  #False
> print(x <= y)  #True
> ```

2.4. 逻辑运算符

> and : 与, 并且
>
> or： 或, 或者
>
> not：非, 取反

短路操作【扩展】：

and短路操作:

- 2边都为True则为True, 有1边为False则为False
- 从左往右: 判断每个值是否为False,如果为False则直接返回该值,否则继续判断第二个数...

```
print(0 and 4)  # 0
print(None and 3 and 5)  # None
print(3 and [] and 6)  # []
print(10 and print(666) and print(888))  # 666   None
```

or短路操作:

- 2边都为False则为False, 有1边为True则为True
- 从左往右: 判断每个值是否为True,如果为True则直接返回该值,否则继续判断第二个数...

```
print(0 or 4)  # 4
print(3 or 4)  # 3
print(-2 or print(666) or print(888))
```

```
bool隐式判断：
  数值：   0为假,其他为真
  字符串：空字符串('')是假,其他为真
  None：  None就是假
  list：  空列表([])是假,其他为真
  tuple：空元组()是假,其他为真
  dict：  空字典({})为假,其他为真
```

2.5. 成员运算符和身份运算符

> 成员运算符:
>
> in, not in
>
> 身份运算符:
>
> is, is not