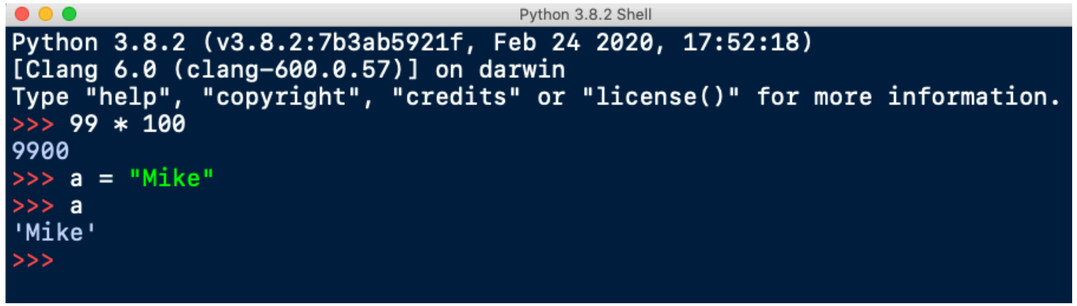
1

# 交互模式与脚本模式

## 交互模式：

1、打开IDLE即为交互模式

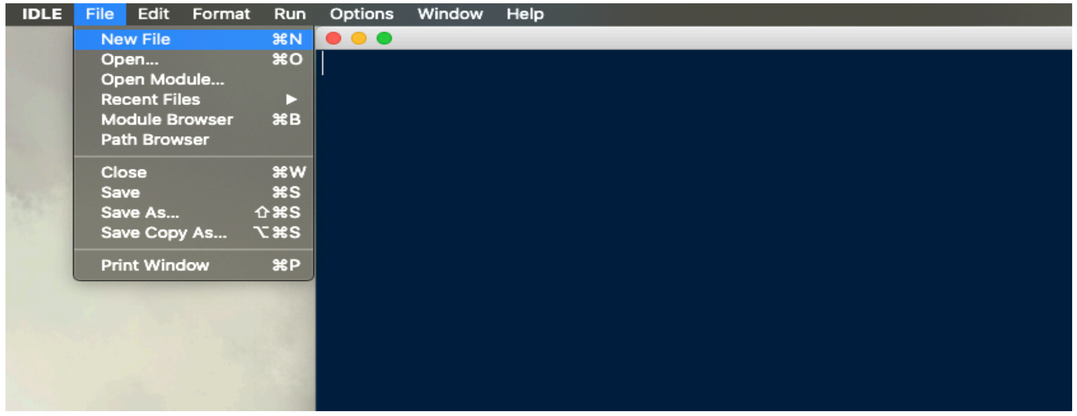
2、能及时交互，无需等待所有代码完成




## 脚本模式：

1、“File”->"New File" 打开

2、不能及时交互，需等待所有代码完成



1

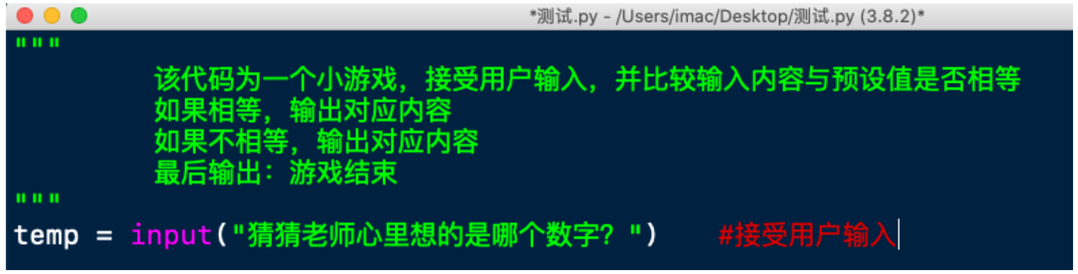
# 单行和多行注释

## 单行注释：

在代码的后方、或者新的一行中使用#号进行注释，#之后的内容不会被执行。注释内容多为描述代码功能。


## 多行注释：

在新的一行中，前后使用三个双引号，进行多行内容的注释。如右图

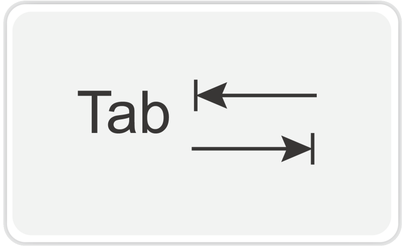


1

# 缩进与BIF

## 缩进：

使用TAB键或四次空格键进行缩进




## BIF：

BIF = 内置函数Built-in Functions

在IDLE中输入dir(__builtins__)可以看到Python提供的内置函数列表。

help()这个内置函数可以显示某个BIF的功能描述。

2

# 变量及其命名规则

## 变量：

1、变量就是一个名字，变量中存储了数据

2、变量中的数据可以改变

3、使用变量之前，必须先进行赋值

```
>>> name = "Mike"
>>> print(name)
Mike
```

## 命名规则：

1、由字母、数字、下划线组成

2、不能以数字开头；不能和关键字重名

3、Python区分大小写

4、等号(=)是赋值号，左变量，右数据

```
#正确命名
number = 10
_name  = "Mike"
name2 = "David"

#错误命名
1number = 10
print = 10

#更佳
number_one  = 12
```

2

# 字符串与长字符串

## 字符串：

1、字符串和数字是截然不同的

2、字符串用引号括起来

3、Python不区分单引号和双引号，但必须成对

4、如果字符串中需要出现单引号或者双引号

使用与外围引号相反的引号

外围用双引号，内部用单引号，反之亦然

```
#正确命名
10
"list"
'list'

#错误命名
"no
'no

#不相等
"10" == '10'     #相等
"10" != 10       #不相等

#转义:使用相反引号
a = "this's Mr.Mike"
```

## 长字符串：

1、使用前后各三个双引号包裹内容

```
poetry = """
                从明天起，做一个幸福的人
                喂马、劈柴，周游世界
                从明天起，关心粮食和蔬菜
                我有一所房子，面朝大海，春暖花开
                从明天起，和每一个亲人通信
                告诉他们，我的幸福
            """
```

3

# 条件分支

## 条见分支：

1、根据不同的条件执行不同的任务

2、if和elif后面都需要有条件判断语句

else后面不需要

3、elif 是else if的缩写，在不满足前面的if情况之下，再次进行判断

4、else是不满足前面的if和elif之外的所有情况

5、判断是否相等，需要使用两个等号==

```
if 2 != 1 :
        print("true")
############################
if 10 > 9 :
        print("true")
else:
        print("False")
############################
if 10 > 9 :
        print("True")
elif 10 > 7 :
        print("true")
else:
        print("False")
```

3

# While循环

**while循环：**

1、while的执行的前提是while语句后的条件判断语句成立

2、while语句后可以跟条件判断或者直接True/False。

3、while True为无限循环，可以通过点击窗口关闭按钮停止，或者使用ctrl+c中断。

4、当条件为真时，执行循环体，直到条件不为真

```
while True :
        print("Hello world")

while 1 < 2 :
        print("Hello world")

flag= 0
while flag < 3 :
    print(flag)
    flag += 1
```

3

# 随机模块

**random模块：**

1、random模块，也叫随机模块，可以帮我们进行随机的操作。

2、在使用random模块之前，需要先进行导入

import random

3、随机一个范围内的整数：

random.randint(start,end)

其中random是模块名，randint是获取随机整数的方法，start是范围的起点，end是终点。获得的随机数在起点和终点之间，注意：随机包含起点和终点

```
import random
a = random.randint(0,10)
#a的取值范围为0到10
b = random.randint(1,2)
#b的取值范围为1或者2
```

4

数据类型

## Int\Float\String\Bool:

1、**Int**：整数型，没有小数点的数

2、**Float**：浮点型，只要有小数点的数都是浮点型

3、**String**：字符型，使用双引号或单引号包裹的内容都属于字符符，单个内容我们称为字符，多个内容我们称为字符串。

4、**Bool**:布尔型，bool型只有两个：True或者False，在Python中，T和F需要大写，布尔类型为条件判断的结果，分别为：成立或不成立，正确或错误

```
#整数
number = 10
#浮点型
number_two = 0.5
number_one = 1.5e20 #E记法
#bool布尔类型 -True False
condition = True
condition = False
#字符型
string1 = "双引号字符串"
string2 = '单引号字符串'
```

4

# 类型转换

## int()\float()\str()：

1、**int()**:将内容转换为整数型

注意：如果是字符串，字符串中只能有整型数字

2、**float()**:将内容转换为浮点型

注意：如果是字符串，字符串中只能有整型数字或浮点型数据

3、**str()**:将内容转换为字符型

```
#整数
number = 10
#浮点型
number_two = 0.5
number_one = 1.5e20 #E记法
#bool布尔类型 -True False
condition = True
condition = False
#字符型
string1 = "双引号字符串"
string2 = '单引号字符串'
```

5

# 获取类型信息

**type\isinstance：**

1、可以使用type()方法来查看数据的类型。

2、isinstance()：判断某个数据与指定数据类型是否匹配

3、type更多在我们调试过程中使用，如果想要进行比较的话，因为type返回给我们的信息比较复杂，不便于在程序当中使用，所以isinstance返回True/False，可以更好的服务我们。

```
>>> a = 10
>>> b = "Mike"
>>> type(a)
<class 'int'>
>>> isinstance(b,str)
True
```

5

# 算数操作符

## 算数操作符：

+ ： 加法

- ： 减法，(一元操作符表示负数)

* ： 乘法

/ ： 除法(真正的除法，结果为浮点数)

% ： 取余数，求模

** ： 幂运算(3**2 = 3*3)

// ： 地板除法(返回比商小的最大整数舍弃小数部分)

注意：

当一个表达式存在着多个运算符的时候，严格按照优先级规定的级别来进行运算。

在适当的地方加上括号，会是更好的方案。()的优先级最高。

5

# 自增运算符

**自增运算符：**

+= ： a += 1 # a=a+1

-= ： a -= 1 # a=a-1

*= ： a *= 1 # a=a*1

/= ： a /= 1 # a=a/1

%= ： a %= 1 # a=a%1

```
10 += 2   #自增运算10+2
10 -= 2   #自减运算10-2
10 *= 2   #自乘运算10*2
10 /= 2   #自除运算10/2
10 %= 2   #自求余运算10%2
```

6



# 比较操作符

1、比较操作符，也叫作关系运算符。

2、判断表达式的关系是否成立，返回一个布尔类型的值(True、False)

3、6种关系：

<、 <=：小于、小于等于

>、 >=：大于、大于等于

==、 !=：恒等于、不等于

```
2 > 1   #大于      ---True
2 >= 1  #大于等于  ---True
2 < 1   #小于      ---False
2 <= 1  #小于等于  ---False
2 == 1  #恒等于    ---False
2 != 1  #不等于    ---False
```

6

# 逻辑操作符

## 与、或、非

1、and：逻辑与，前后都为真，结果才为真。

2、or：逻辑或，前后有一个为真，结果就为真。

3、not：否，非，一元操作符，真变假，假变真。

```
2 > 1 and 2 != 1
#2大于1，并且2不等于1，前后两者条件都同时成立，返回true

2 > 1 or 2 != 1
#2大于1，或者2不等于1，前后两者条件有一个成立，返回true

not 2 > 1
#2大于1为真True，not取反，则真变假，返回False
```

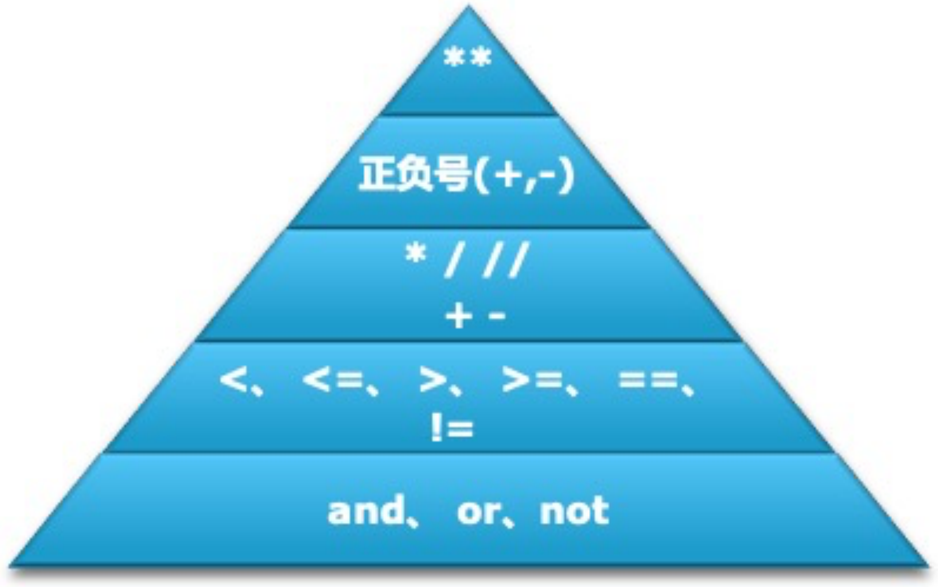
3<4<5，在Python中是行得通的，相当于3<4 and 4<5

6

优先级

**优先级：**

1、在存在多种操作符的时候，程序会有自己的处理优先级，当大家在编写代码时，尽量采用括号来人为区分处理的优秀级。



7



## 三元操作符

1、a = x if 条件 else y

当条件为True时，a的值是x，否则为y

2、三元操作符一般应用于满足某条件赋值A，否者B的情况，可以有效的简洁我们的代码，建议大家在熟悉之后多多使用！

```
x = 12
y = 13
small = 0
#常规写法
if x < y :
    small = x
else :
    small = y
#三元操作符
small = x if x < y else y
```

7



## Assert断言

1、断言assert:当后面的条件为假的时候，程序会自动崩溃并抛出AssertionError的异常。

2、assert可以帮助我们对程序作出可控崩溃。也可以帮助我们在debug时进行数据的验证

```
>>> assert 10 > 9
>>> assert 9  > 10
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    assert 9  > 10
AssertionError
```

8



## for 循环

1、for后面需跟目标，目标可以理解为只在for循环中起作用的变量，in 序列，目标会依此到序列中取出一个值

2、当我们的目标不会在循环体中使用时，可以使用下划线_来进行占位。

```
for "目标" in "序列":
    "循环体"
```

8

# 内建函数range()

**range([strat,]stop[,step = 1])**

1、range常与for循环搭配使用。range的作用是生成一个数据范围序列，如右图示。

2、range共有三个参数：

**start**:数据的启始点,(可有可无，当有时需在后方加入逗号，若无，默认从0开始)

**stop**：数据的终止点,目标取值会取到stop-1(必须有)

**step**：步长，取值间隔,(可有可无，如无，默认为1，若有，则需要在前面加入逗号。)

```
for item in range(3):
    print(item)
#结果为：0，1，2。取到stop-1

for i in range(1,4):
    print(i)
#结果为：1，2，3。range(start,stop)

for a in range(1,6,2):
    print(a)
#结果为：1，3，5，步长为1，取一次间隔一次
#    range(start,stop,step)
```

备注：序列，有序的数列，详见16

9

Break

## break语句

1、break的作用是终止循环，跳出循环体。

2、break用于在循环中，满足某条件，就跳出循环不再循环的情况。

```
flag = 0
while True :
    print(flag)
    flag += 1
    if flag == 12:
        break
#结果：0 ～ 11，12及12之后的数字不会被打印
```

注意⚠️：在break之后的同级代码不会被执行。

9



## continue语句

1、continue的作用是终止本次循环

2、跳过本次循环尚未执行的语句，开始下一次的循环判断。

3、break是跳出，代表所有循环结束

continue是跳过，代表本次循环结束，后续循环继续

```
for item in range(10):
    if item == 5:
        continue
    else:
        print(item)
#结果：打印出0 ～ 9 ，不包括5
```

注意⚠️：在continue之后的同级代码不会被执行。

## 创建列表List

1、中括号括起来一堆数据，用逗号隔开

2、使用list()或 [] 创建一个空列表

3、数据元素可以是任何的类型，包括另一个列表。

```
#创建空列表
data1 = []
data2 = list()

#数据元素可以是多种类型
data3 = [1,3.13,"string"]

#数据元素是元祖也可以。
data4 = [1,[2,3,4,5]]
```

10



## 列表元素增加：

1、append()方法：追加元素到列表末尾

2、extend()方法：把另一个列表追加到末尾

3、insert(index,data)方法：在index位置插入一个data。

索引index：总是从0开始

```
data = [1,2,3]#初始值

data.append(4)
#结果为：【1, 2, 3, 4】

data.extend([5,6])
#结果为：【1, 2, 3, 4, 5, 6】

data.insert(0,10)
#结果为：【10, 1, 2, 3, 4, 5, 6]
```

10



## 列表元素获取：

1、通过索引也就是下标直接访问

2、用for循环遍历整个列表

3、通过索引不仅可以进行查，也可以改

```
data = [1,2,3]#初始值

print(data[1])
#打印结果为:2

data[0] = 0
#结果为：【0，2，3】

for i in data:
    print(i)
#依此打印0，2，3，每次只打印一个元素。
```