语句

行

1. 物理行：程序员编写代码的行。
2. 逻辑行：python 解释器需要执行的指令。
3. 建议一个逻辑行在一个物理行上。
4. 如果一个物理行中使用多个逻辑行，需要使用分号；隔开。
5. 如果逻辑行过长，可以使用隐式换行或显式换行。

隐式换行：所有括号的内容换行,称为隐式换行

括号包括: () [] {} 三种

显式换行：通过折行符 \ (反斜杠)换行，必须放在一行的末尾，目的是告诉解释器,下一行也是本行的语句。

"""

```
    行
    10:40
"""
# 三个物理行，三个逻辑行
a = 10
b = 20
c = a + b
# 一个物理行，三个逻辑行(不建议)
a = 10;b = 20;c = a + b
# 三个物理行，一个逻辑行(适用于一行太长时使用)
# 折行符
d = 1+\
    2+3+\
    4+5
# 括号是天然的折行符
e = 1+(2+3
      +4)*5
```

pass 语句

通常用来填充语法空白。

选择语句

If elif else 语句

1. 作用:
   让程序根据条件选择性的执行语句。
2. 语法:
   if 条件 1:
   语句块 1
   elif 条件 2:
   语句块 2
   else:
   语句块 3
3. 说明:
   elif 子句可以有 0 个或多个。
   else 子句可以有 0 个或 1 个，且只能放在 if 语句的最后。

```
"""
    选择语句
        if 条件bool:
            满足条件执行的代码
        else:
            不满足条件执行的代码
         if 条件1:
            满足条件1执行的代码
         elif 条件2:
            满足条件2执行的代码
         else:
            以上都不满足执行的代码
    练习:exercise01.py ~ exercise06.py
"""
sex = input("请输入性别：")
# 缩进不是tab，而是四个空格.
if sex == "男":
    print("您好，先生！")
elif sex == "女":
    print("您好，女士！")
else:
    print("性别未知")
# 调试：让程序在指定的行中断，逐语句执行.
#       查看程序执行过程，查看变量的取值。
# 步骤：
# 1. 在可能出错的行，加断点.
```

```
# 2. 开始调试Shift + F9.
# 3. 逐语句执行 F8.
# 4. .....
# 5. 停止调试Ctrl + F2
```

if 语句的真值表达式

if 100:
print("真值")
等同于
if bool(100):
print("真值")

条件表达式

语法：变量 = 结果 1 if 条件 else 结果 2
作用：根据条件(True/False) 来决定返回结果 1 还是结果 2。
"""

```
    if的真值表达式
        if 变量:
            变量存在数据则执行
    条件表达式：
        有选择性的为变量进行赋值
    练习:exercise07.py
"""
number = 10
if number != 0:
    print("不是零")
# 1. if的真值表达式
if number:
    # if bool(number):
    print("不是零")
# 2. 条件表达式：
if input("请输入性别：") == "男":
    sex_id = 1
else:
    sex_id = 0
sex_id = 1 if input("请输入性别：") == "男" else 0
print(sex_id)
```

循环语句

while 语句

1. 作用:
   可以让一段代码满足条件，重复执行。
2. 语法:
   while 条件:
   满足条件执行的语句
   else:
   不满足条件执行的语句
3. 说明:
   else 子句可以省略。
   在循环体内用 break 终止循环时,else 子句不执行。

"""

```
    循环语句
        while 条件:
            循环体
"""
# 死循环：循环条件永远满足
while True:
    str_usd = input("请输入美元：")
    int_usd = int(str_usd)
    rmb = int_usd * 6.86
    print("人民币是："+ str(rmb))
    if input("按下q键退出:") == "q":
        break # 退出循环
```

"""

```
    循环语句
        while 执行预定次数
"""
count = 0
while count < 5:  # 0  1  2  3  4
    print("跑圈")
    # print(count)
    count += 1
```

append 语句

语法: 文件名.append('需要添加的元素')

作用:

添加在列表 ,等的末尾,且不会影响列表中的其他元素.

2.在列表中插入元素

insert 语句

语法: 文件名.insert(文件位置,”需要添加的元素”)

在索引出添加空间,并将原宿储存到这个位置,这种操作将列表中即有的每个元素都右移一位.

*3. del 语句

从列表中删除元素,删除之后将再无法使用.

语法: del 文件名[元素位置](从 0 开始数)

*4.pop 语句

将元素从列表中删除,并接着使用它的值.

语法: 文件名.pop()

5.根据值删除元素

语法: 文件名.remove(‘需要删除的元素’)

```
'''
    列表
'''
# 输入数据 -- 创建列表
list = ['laber','mary','kit']
print(list)
# 删除指定位置元素
del list[1]
# 在末尾添加元素
list.append('bob')
print(list)
# 在指定位置添加元素
list.insert(1,'meimei')
list.insert(-1,'meili')
list.insert(0,'wumei')
# 在末尾添加元素
list.append('ligee')
print(list)
# 删除末尾元素
popped_list = list.pop()
print(list)
# 删除末尾元素的输出
print(popped_list)
# 删除指定元素
list.remove('laber')
print(list)
```