1. 整数（int）:

整数是不带小数点的数字。例如：

x = 5

y = -10

2. 浮点数（float）:

浮点数包含小数点的数字。例如：

pi = 3.14

radius = 2.0

3. 字符串（str）:

字符串是由字符组成的文本序列。可以使用单引号或双引号表示。例如：

message = "Hello, World!"

4. 列表（list）:

列表是有序、可变的元素序列。可以包含不同类型的元素。例如：

fruits = ['apple', 'orange', 'banana']

访问元素：

可以使用索引来访问列表中的元素，索引从0开始：

print(fruits[0])

切片（Slicing）：

通过切片可以获取列表的子集：

sequence[start:stop:step]

start：切片开始的位置，默认为序列的开头（索引0）。

stop：切片结束的位置（不包含该位置的元素）。如果省略，则默认为序列的末尾。

step：切片的步长，即每次跳过元素的间隔。默认为1。

修改元素：

可以通过索引直接修改列表中的元素：

fruits[1] = 'grape'

方法：

append(element)：

将元素追加到列表的末尾。

fruits = ['apple', 'orange']

fruits.append('banana')

print(fruits) # 输出: ['apple', 'orange', 'banana']

insert(index, element)：

在指定位置插入元素。

fruits = ['apple', 'orange']

fruits.insert(1, 'banana')

print(fruits) # 输出: ['apple', 'banana', 'orange']

remove(element)：

移除列表中第一个匹配的元素。

fruits = ['apple', 'orange', 'banana']

fruits.remove('orange')

print(fruits) # 输出: ['apple', 'banana']

index(element[, start[, end]])：

返回列表中第一个匹配元素的索引，可指定搜索范围。

fruits = ['apple', 'orange', 'banana']

index = fruits.index('orange')

print(index) # 输出: 1

len() 函数来获取列表的长度，即列表中包含的元素个数

my\_list = [1, 2, 3, 4, 5] # 使用 len() 函数获取列表长度

length = len(my\_list)

print(f"The length of the list is: {length}")

5. 元组（tuple）:

元组与列表类似，但是一旦创建，其元素就不可更改。用小括号表示。例如：

coordinates = (3, 4)

6. 集合（set）:

集合是无序的、不重复的元素集合。用大括号表示。例如：

unique\_numbers = {1, 2, 3, 4, 5}

7. 字典（dict）:

字典是键-值对的集合，用于存储相关数据。键必须是唯一的，而值可以是任意类型的数据。例如：

person = {'name': 'John', 'age': 30, 'city': 'New York'}

keys()：

返回包含字典所有键的视图。

视图是一种对象，它提供了对其他数据结构（比如字典）的动态查看。这意味着当原始数据结构发生变化时，视图会自动更新以反映这些变化。

person = {'name': 'John', 'age': 30}

keys\_view = person.keys()

print(keys\_view) # 输出: dict\_keys(['name', 'age'])

访问元素：

字典的元素可以通过键来访问：

print(my\_dict['name']) # 输出 'Alice'

修改和添加元素：

# 修改元素值

my\_dict['age'] = 26

# 添加新的键值对

my\_dict['gender'] = 'Female'

values()：

返回包含字典所有值的视图。

person = {'name': 'John', 'age': 30}

values\_view = person.values()

print(values\_view) # 输出: dict\_values(['John', 30])

items()：

返回包含字典所有键值对的视图。

person = {'name': 'John', 'age': 30}

items\_view = person.items()

print(items\_view) # 输出: dict\_items([('name', 'John'), ('age', 30)])

my\_dict = {'name': 'John', 'age': 30, 'city': 'New York'} # 使用 items() 与 for 循环获取键和值

for key, value in my\_dict.items():

print(f"Key: {key}, Value: {value}")

函数：

在编程中，函数是一段可重复使用的代码块，它接受输入参数，执行特定的任务，然后返回一个结果。函数提供了一种将代码模块化和抽象化的方式，使得程序更易于理解、维护和重用。

print函数：

print("Hello, World!")

打印多个值：

name = "Alice"

age = 25

print("Name:", name, "Age:", age)

f-string（formatted string literals）是一种方便的字符串格式化方法

1. 插入变量：

使用 f-string 时，可以直接在字符串中插入变量，变量名用花括号 {} 包裹：

name = "Alice"

age = 25

print(f"Name: {name}, Age: {age}")

这将输出：Name: Alice, Age: 25

2. 插入表达式：

在 f-string 中，你也可以插入表达式，这些表达式会在运行时计算：

x = 10

y = 20

print(f"The sum of {x} and {y} is {x + y}")

这将输出：The sum of 10 and 20 is 30

3. 格式化数字：

你可以使用格式说明符控制数字的显示格式，例如限制小数点后的位数：

pi = 3.14159print(f"Value of pi: {pi:.2f}")

这将输出：Value of pi: 3.14

格式说明符的一般结构是 :[填充字符][对齐方式][宽度][.精度][数据类型]

格式说明符的一般结构是:[填充字符][对齐方式][宽度][.精度][数据类型]，其中每一部分都是可选的。以下是格式说明符的各部分详细说明：

填充字符（Fill Character）:

使用指定的字符填充字段的空白部分。默认情况下，填充字符为空格。

例如，:>10 将在字段宽度不足时使用空格填充，并右对齐。

对齐方式（Alignment）:

<：左对齐

^：居中对齐

>：右对齐

例如，:^10 将居中对齐一个宽度为 10 的字段。

宽度（Width）:

指定字段的最小宽度。如果字段的内容不足宽度，则使用填充字符进行填充。

例如，:>5 将在字段宽度不足时使用空格填充，并右对齐，宽度为 5。

精度（Precision）:

对于浮点数，指定小数点后的位数。

对于字符串，指定字符的最大数量。

例如，: .2f 将浮点数显示为小数点后两位。

数据类型（Type）:

d：整数

f：浮点数

s：字符串

x：十六进制（仅适用于整数）

例如，:s 将保持字符串原样，:d 将将整数作为整数显示。

4. 使用字典和列表：

在 f-string 中，你可以使用字典的键或列表的索引来插入相应的值：

person = {'name': 'John', 'age': 30}

print(f"Name: {person['name']}, Age: {person['age']}")

fruits = ['apple', 'orange', 'banana']print(f"The first fruit is {fruits[0]}")

这将输出：

Name: John, Age: 30

The first fruit is apple

5. 格式化时间：

使用 f-string 可以方便地格式化日期和时间：

from datetime import datetime

today = datetime.today()

print(f"Today is {today:%Y-%m-%d}")

这将输出类似于：Today is 2023-01-12

6. 对齐文本：

通过在花括号中使用冒号和格式化选项，你可以控制文本的对齐方式：

word = "Python"

print(f"{word:<10} is fun!") # 左对齐

print(f"{word:^10} is fun!") # 居中对齐

print(f"{word:>10} is fun!") # 右对齐

这将输出：

Python is fun! # 左对齐

Python is fun! # 居中对齐

Python is fun! # 右对齐

1. 算术运算符：

# 加法

a + b

# 减法

a - b

# 乘法

a \* b

# 除法

a / b

# 取余

a % b

# 取整除法

a // b

# 乘方

a \*\* b

2. 比较运算符：

# 等于

a == b

# 不等于

a != b

# 大于

a > b

# 小于

a < b

# 大于等于

a >= b

# 小于等于

a <= b

3. 赋值运算符：

# 等于

a = b

# 加等于

a += b

# 减等于

a -= b

# 乘等于

a \*= b

# 除等于

a /= b

# 取余等于

a %= b

# 取整除等于

a //= b

# 乘方等于

a \*\*= b

4. 逻辑运算符：

# 与

a and b

# 或

a or b

# 非

not a

1. 条件判断语句 - if, elif, else：

x = 10

if x > 0:

print("x 是正数")

elif x == 0:

print("x 是零")

else:

print("x 是负数")

5. 循环语句 - for, while：

for 循环：

for variable in iterable: # 在这个块中执行操作，使用 variable 代表当前迭代的元素

for i in range(5):

print(i)

在 Python 中，range 是一个用于生成整数序列的内置函数。range 函数有三种形式：

range(stop)：

生成从 0 开始、到 stop-1 结束（不包括 stop）的整数序列。

numbers = range(5)

print(list(numbers)) # 输出 [0, 1, 2, 3, 4]

range(start, stop)：

生成从 start 开始、到 stop-1 结束的整数序列。

numbers = range(2, 8)

print(list(numbers)) # 输出 [2, 3, 4, 5, 6, 7]

range(start, stop, step)：

生成从 start 开始、到 stop-1 结束的整数序列，以 step 为步长。

numbers = range(1, 10, 2)

print(list(numbers)) # 输出 [1, 3, 5, 7, 9]

while 循环：

count = 0

while count < 5:

print(count)

count += 1

6. 中断语句 - break, continue, pass：

break:

break 语句用于终止循环，即使循环条件没有达到结束的条件。通常在满足某个条件时，提前跳出循环。

for i in range(10):

if i == 5:

break

print(i)

continue:

continue 语句用于跳过当前循环中剩余的代码，直接进入下一次循环迭代。

for i in range(10):

if i == 5:

continue

print(i)

pass:

pass 语句是一个空语句，用于在语法上需要语句但不需要任何操作的情况。通常用于占位，保持代码结构完整。

for i in range(5):

pass # 什么也不做，用于占位

****题目：**** 创建一个学生成绩的字典，包含学生姓名和对应的成绩。编写一个程序，计算平均成绩，并输出高于平均成绩的学生姓名。

grades = {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 95, 'Eva': 89}  
sum=0  
for grade in grades.values():  
 sum+=grade  
average=sum/len(grades)  
print(f"平均成绩：{average}")  
for student in grades.keys():  
 if grades[student]>average:  
 print(f"{student}:{grades[student]}")

创建一个购物清单，包含商品名称和价格。编写一个程序，计算购物清单的总价格，并输出清单中最贵的商品。

shopping\_list = {'Apple': 2.5, 'Banana': 1.8, 'Milk': 3.0, 'Bread': 2.2, 'Eggs': 2.5}  
sum=0  
for value in shopping\_list.values():  
 sum+=value  
print(f"sum:{sum}")  
max\_item='Apple'  
max=2.5  
for item,value in shopping\_list.items():  
 if value>max:  
 max\_item=item  
 max=value  
print(f"item:{max\_item},value:{max}")