的python板子

Python内置函数

abs()	**dict()**	**help()**	**min()**	**setattr()**
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	**import**()
complex()	hasattr()	max()	round()	reload()
delattr()	hash()	memoryview()	set()	参数可以help查一下

Python math 模块

Python **math** 模块提供了许多对浮点数的数学运算函数。**math** 模块下的函数，返回值均为浮点数，除非另有明确说明。

如果你需要计算复数，请使用 cmath 模块中的同名函数。要使用 math 函数必须先导入, 查看 math 模块中的内容:

```
import math  dir(math)
```

math 模块常量

常量	描述
math.e	返回欧拉数 (2.7182...)
math.inf	返回正无穷大浮点数
math.nan	返回一个浮点值 NaN (not a number)
math.pi	π 一般指圆周率。 圆周率 PI (3.1415...)

常量	描述
math.tau	数学常数 τ = 6.283185...，精确到可用精度。Tau 是一个圆周常数，等于 2π，圆的周长与半径之比。

math 模块方法

方法	描述
math.acos(x)	返回 x 的反余弦，结果范围在 0 到 pi 之间。
math.acosh(x)	返回 x 的反双曲余弦值。
math.asin(x)	返回 x 的反正弦值，结果范围在 -pi/2 到 pi/2 之间。
math.asinh(x)	返回 x 的反双曲正弦值。
math.atan(x)	返回 x 的反正切值，结果范围在 -pi/2 到 pi/2 之间。
math.atan2(y, x)	返回给定的 X 及 Y 坐标值的反正切值，结果是在 -pi 和 pi 之间。
math.atanh(x)	返回 x 的反双曲正切值。
math.ceil(x)	将 x 向上舍入到最接近的整数
math.comb(n, k)	返回不重复且无顺序地从 n 项中选择 k 项的方式总数。
math.copysign(x, y)	返回一个基于 x 的绝对值和 y 的符号的浮点数。
math.cos()	返回 x 弧度的余弦值。
math.cosh(x)	返回 x 的双曲余弦值。
math.degrees(x)	将角度 x 从弧度转换为度数。
math.dist(p, q)	返回 p 与 q 两点之间的欧几里得距离，以一个坐标序列（或可迭代对象）的形式给出。 两个点必须具有相同的维度。
math.erf(x)	返回一个数的误差函数
math.erfc(x)	返回 x 处的互补误差函数
math.exp(x)	返回 e 的 x 次幂，Ex， 其中 e = 2.718281... 是自然对数的基数。
math.expm1()	返回 Ex - 1， e 的 x 次幂，Ex，其中 e = 2.718281... 是自然对数的基数。这通常比 math.e ** x 或 pow(math.e, x) 更精确。
math.fabs(x)	返回 x 的绝对值。
math.factorial(x)	返回 x 的阶乘。 如果 x 不是整数或为负数时则将引发 ValueError。
math.floor()	将数字向下舍入到最接近的整数
math.fmod(x, y)	返回 x/y 的余数
math.frexp(x)	以 (m, e) 对的形式返回 x 的尾数和指数。 m 是一个浮点数， e 是一个整数，正好是 x == m * 2**e 。 如果 x 为零，则返回 (0.0, 0) ，否则返回 0.5 <= abs(m) < 1 。

方法	描述
math.fsum(iterable)	返回可迭代对象 (元组, 数组, 列表, 等)中的元素总和，是浮点值。
math.gamma(x)	返回 x 处的伽马函数值。
math.gcd()	返回给定的整数参数的最大公约数。
math.hypot()	返回欧几里得范数，sqrt(sum(x**2 for x in coordinates))。 这是从原点到坐标给定点的向量长度。
math.isclose(a,b)	检查两个值是否彼此接近，若 a 和 b 的值比较接近则返回 True，否则返回 False。。
math.isfinite(x)	判断 x 是否有限，如果 x 既不是无穷大也不是 NaN，则返回 True ，否则返回 False 。
math.isinf(x)	判断 x 是否是无穷大，如果 x 是正或负无穷大，则返回 True ，否则返回 False 。
math.isnan()	判断数字是否为 NaN，如果 x 是 NaN（不是数字），则返回 True ，否则返回 False 。
math.isqrt()	将平方根数向下舍入到最接近的整数
math.ldexp(x, i)	返回 x * (2**i) 。 这基本上是函数 math.frexp() 的反函数。
math.lgamma()	返回伽玛函数在 x 绝对值的自然对数。
math.log(x[, base])	使用一个参数，返回 x 的自然对数（底为 e ）。
math.log10(x)	返回 x 底为 10 的对数。
math.log1p(x)	返回 1+x 的自然对数（以 e 为底）。
math.log2(x)	返回 x 以 2 为底的对数
math.perm(n, k=None)	返回不重复且有顺序地从 n 项中选择 k 项的方式总数。
math.pow(x, y)	将返回 x 的 y 次幂。
math.prod(iterable)	计算可迭代对象中所有元素的积。
math.radians(x)	将角度 x 从度数转换为弧度。
math.remainder(x, y)	返回 IEEE 754 风格的 x 除于 y 的余数。
math.sin(x)	返回 x 弧度的正弦值。
math.sinh(x)	返回 x 的双曲正弦值。
math.sqrt(x)	返回 x 的平方根。
math.tan(x)	返回 x 弧度的正切值。
math.tanh(x)	返回 x 的双曲正切值。
math.trunc(x)	返回 x 截断整数的部分，即返回整数部分，删除小数部分

日历（Calendar）模块

此模块的函数都是日历相关的，例如打印某月的字符月历。

星期一是默认的每周第一天，星期天是默认的最后一天。更改设置需调用calendar.setfirstweekday()函数。模块包含了以下内置函数：

序号	函数及描述
1	**calendar.calendar(year,w=2,l=1,c=6)** 返回一个多行字符串格式的 year 年年历，3 个月一行，间隔距离为 c。 每日宽度间隔为w字符。每行长度为 21* W+18+2* C。**l** 是每星期行数。
2	**calendar.firstweekday( )** 返回当前每周起始日期的设置。默认情况下，首次载入 calendar 模块时返回 0，即星期一。

3	**calendar.isleap(year)**
	是闰年返回 True，否则为 False。
	
	>>> import calendar
	>>> print(calendar.isleap(2000))
	True
	>>> print(calendar.isleap(1900))
	False

4	**calendar.leapdays(y1,y2)** 返回在Y1，Y2两年之间的闰年总数。
5	**calendar.month(year,month,w=2,l=1)** 返回一个多行字符串格式的year年month月日历，两行标题，一周一行。每日宽度间隔为w字符。每行的长度为7* w+6。l是每星期的行数。
6	**calendar.monthcalendar(year,month)** 返回一个整数的单层嵌套列表。每个子列表装载代表一个星期的整数。Year年month月外的日期都设为0;范围内的日子都由该月第几日表示，从1开始。

7	**calendar.monthrange(year,month)**
	返回两个整数。第一个是该月的星期几，第二个是该月有几天。星期几是从0（星期一）到 6（星期日）。
	>>> import calendar
	>>> calendar.monthrange(2014, 11)
	(5, 30)
	解释：5 表示 2014 年 11 月份的第一天是周六，30 表示 2014 年 11 月份总共有 30 天。

8	**calendar.prcal(year, w=0, l=0, c=6, m=3)** 相当于 print (calendar.calendar(year, w=0, l=0, c=6, m=3))。
9	**calendar.prmonth(theyear, themonth, w=0, l=0)** 相当于 print(calendar.month(theyear, themonth, w=0, l=0))。

序号	函数及描述
10	**calendar.setfirstweekday(weekday)** 设置每周的起始日期码。0（星期一）到6（星期日）。
11	**calendar.timegm(tupletime)** 和time.gmtime相反：接受一个时间元组形式，返回该时刻的时间戳（1970纪元后经过的浮点秒数）。
12	**calendar.weekday(year,month,day)** 返回给定日期的日期码。0（星期一）到6（星期日）。月份为 1（一月） 到 12（12月）。

常用数据结构

1. 列表（List）
列表是Python中最常用的数据结构之一，支持动态大小，能够以简单的方式存储多个项目。以下是一些基本操作：

```
fruits = ['apple', 'banana', 'cherry']
# 添加元素
fruits.append('orange')
# 删除元素
fruits.remove('banana')
# 查找元素
if 'apple' in fruits:
    print("Apple is in the list!")
```

新手易踩坑：在使用列表时，注意索引从0开始。如果你试图访问一个超出范围的索引，Python会抛出IndexError。

2. 字典（Dictionary）
字典是一种无序的键值对集合，适合用于快速查找。它的查找速度非常快，通常是O(1)。

```
student = {'name': 'Alice', 'age': 20}
# 添加元素
student['grade'] = 'A'
# 删除元素
del student['age']
# 查找元素
print(student.get('name'))
```

新手易踩坑：字典的键必须是不可变类型（如字符串、数字、元组），而值可以是任意类型。

3. 集合（Set）
集合是一种无序且不重复的元素集合，适合用于去重和集合运算。

```
numbers = {1, 2, 3, 4, 5}
# 添加元素
numbers.add(6)
# 删除元素
numbers.discard(3)
# 集合运算
even_numbers = {2, 4, 6}
print(numbers.intersection(even_numbers))
```

新手易踩坑：集合中的元素是无序的，因此不能通过索引访问。

4. 栈（Stack）
栈是一种后进先出（LIFO）的数据结构。我们可以使用列表来实现栈的功能。

```
stack = []
# 入栈
stack.append(1)
# 出栈
top = stack.pop()
```

新手易踩坑：在使用栈时，确保在出栈之前栈中有元素，否则会抛出IndexError。

5. 队列（Queue）
队列是一种先进先出（FIFO）的数据结构。我们可以使用collections模块中的deque来实现队列。

```
from collections import deque
queue = deque()
# 入队
queue.append(1)
# 出队
first = queue.popleft()
```

新手易踩坑：使用列表实现队列时，出队操作会导致O(n)的时间复杂度，因此推荐使用deque。

常用算法
1. 排序算法
排序是数据处理中的基本操作之一。Python内置的sorted()函数和列表的sort()方法都可以轻松实现排序

```
sorted_list = sorted([3, 1, 4, 1, 5, 9])# 使用sorted()函数
numbers = [3, 1, 4, 1, 5, 9]# 使用sort()方法
numbers.sort()
```

新手易踩坑：注意sorted()返回一个新列表，而sort()是在原地排序。

2. 查找算法
查找算法用于在数据结构中查找特定元素。最常用的查找算法是线性查找和二分查找。

```
# 线性查找
def linear_search(arr, target):
    for index, value in enumerate(arr):
        if value == target:
            return index
    return -1
# 二分查找
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

新手易踩坑：在使用二分查找时，确保数据是有序的，否则结果将不可靠。