中文	英文	示例	说明
打印()	print()	打印("Hello, World!")	打印输出给定的内容
取长度()	len()	string = "Hello, World!" length = 取长度(string) 打印(length) # 输出: 13	返回对象的长度或元素个数
输入()	input()	name = 输入("请输入您的姓名:") 打印("您的姓名是: " + name)	接收用户输入并返回作为字符串
生成范围()	range()	循环 num 在 生成范围(1, 5): 打印(num) # 输出: 1 2 3 4	生成一个指定范围内的整数序列
字符串型()	str()	number = 42 string = 字符串型(number) 打印(string) # 输出: '42'	将对象转换为字符串
整数型()	int()	string = "42" number = 整数型(string) 打印(number) # 输出: 42	将对象转换为整数
浮点数型()	float()	string = "3.14" number = 浮点数型(string) 打印(number) # 输出: 3.14	将对象转换为浮点数
取类型()	type()	number = 42 打印(取类型(number)) # 输出: int	返回对象的类型
列表型()	list()	string = "Hello" char_list = 列表型(string) 打印(char_list) # 输出: ['H', 'e', 'l', 'l', 'o']	将可迭代对象转换为列表
元组型()	tuple()	list_data = [1, 2, 3] tuple_data = 元组型(list_data) 打印(tuple_data) # 输出: (1, 2, 3)	将可迭代对象转换为元组
字典型()	dict()	person = 字典型(name='Alice', age=25) 打印(person) # 输出: {'name': 'Alice', 'age': 25}	创建一个字典对象
集合型()	set()	numbers = [1, 2, 3, 2, 1] unique_numbers = 集合型(numbers)	创建一个集合对象（去重）

中文	英文	示例	说明
		打印(unique_numbers) # 输出: {1, 2, 3}	
求和()	sum()	numbers = [1, 2, 3, 4, 5] total = 求和(numbers) 打印(total) # 输出: 15	返回可迭代对象的总和
取最大值()	max()	numbers = [1, 2, 3, 4, 5] maximum = 取最大值(numbers) 打印(maximum) # 输出: 5	返回可迭代对象的最大值
取最小值()	min()	numbers = [1, 2, 3, 4, 5] minimum = 取最小值(numbers) 打印(minimum) # 输出: 1	返回可迭代对象的最小值
取绝对值()	abs()	number = -42 absolute = 取绝对值(number) 打印(absolute) # 输出: 42	返回数值的绝对值
替换()	**replace()**	st = "i want a apple" st = st.替换("apple", "mice") 打印(st) # 输出: mice i want a	替代字符串中的某一些子串为另一些字
四舍五入()	**round()**	number = 3.14159 rounded = 四舍五入(number, 2) 打印(rounded) # 输出: 3.14	返回一个数值的四舍五入值
去除空格()	**strip()**	st = " hello " st = st.去除空格() 打印(st+"end") # 输出: helloend	去除字符串前面和后面的空格
排序()	**sorted**()	numbers = [5, 2, 4, 1, 3] sorted_numbers = 排序(numbers) 打印(sorted_numbers) # 输出: [1, 2, 3, 4, 5]	返回一个排序后的可迭代对象
反转()	**reversed**()	numbers = [1, 2, 3, 4, 5] reversed_numbers = 列表型(反转(numbers)) 打印(reversed_numbers) # 输出: [5, 4, 3, 2, 1]	返回一个反转后的可迭代对象

中文	英文	示例	说明
组合()	**zip()**	names = ['Alice', 'Bob', 'Charlie'] ages = [25, 30, 35] zipped = 列表型(组合(names, ages)) 打印(zipped) # 输出: [('Alice', 25), ('Bob', 30)]	将多个可迭代对象按索引位置组合成元组
枚举()	**enumerate()**	names = ['Alice', 'Bob', 'Charlie'] 循环 index, name 在 枚举(names): 打印(f"Name at index {index}: {name}")	返回可迭代对象中元素的索引和值
存在为真()	**any()**	numbers = [0, 1, 2, 3] 打印(存在为真(numbers)) # 输出: True	判断可迭代对象中是否存在任何为真的元素
全部为真()	**all()**	numbers = [0, 1, 2, 3] 打印(全部为真(numbers)) # 输出: False	判断可迭代对象中所有元素是否都为真
设置属性()	**setattr()**	类 Person: name = "" person = Person() 设置属性(person, "name", "Alice") 打印(person.name) # 输出: Alice	设置对象的属性值
删除属性()	**delattr()**	类 Person: name = "Alice" person = Person() 删除属性(person, "name") 打印(hasattr(person, "name")) # 输出: False	删除对象的属性
乘方()	**pow()**	result = 乘方(2, 3) 打印(result) # 输出: 8	返回数值的指定乘方
除法取商余()	**divmod()**	quotient, remainder = 除法取商余(10, 3) 打印(quotient, remainder) # 输出: 3 1	返回两个数值的商和余数
过滤()	**filter()**	numbers = [1, 2, 3, 4, 5] even_numbers = 列表型(过滤(匿名函数 x:x%2==0,numbers)) 打印(even_numbers) # 输出: [2, 4]	使用函数过滤可迭代对象中的元素

中文	英文	示例	说明
截取()	**slice()**	numbers = [0, 1, 2, 3, 4, 5] sliced = numbers[截取(2, 5)] 打印(sliced) # 输出: [2, 3, 4]	返回一个截取对象，用于截取操作
可否调用()	**callable()**	函数 say_hello(): 打印("Hello!") 打印(可否调用(say_hello)) # 输出: True	检查对象是否可调用（函数、方法等）
获取属性()	getattr()	类 Person: name = "Alice" person = Person() name = 获取属性(person, "name") 打印(name) # 输出: Alice	返回对象的属性值
写入()	write()	file = 打开("example.txt", "w") file.写入("Hello, World!") file.关闭 ()	将内容写入文件
加入成员()	append()	numbers = [1, 2, 3] numbers.加入成员(4) 打印(numbers) # 输出: [1, 2, 3, 4]	在列表末尾添加元素
追加数组()	extend()	numbers = [1, 2, 3] more_numbers = [4, 5, 6] numbers.追加数组(more_numbers) 打印(numbers) # 输出: [1, 2, 3, 4, 5, 6]	将可迭代对象中的元素添加到列表末尾
插入()	insert()	numbers = [1, 2, 3] numbers.插入(1, 4) 打印(numbers) # 输出: [1, 4, 2, 3]	在指定索引处插入元素
映射()	map()	numbers = [1, 2, 3, 4, 5] squared_numbers = 列表型(映射(匿名函数 x:x**2,numbers)) 打印(squared_numbers) # 输出: [1, 4, 9, 16, 25]	使用函数对可迭代对象中的每个元素进行映射
累积计算()	reduce()	从 functools 导入 累积计算 numbers = [1, 2, 3, 4, 5]	使用函数对可迭代对象中的元素进行累积计算

中文	英文	示例	说明
		product = 累积计算(匿名函数 x, y: x * y, numbers) 打印(product) # 输出: 120	
打开()	open()	file = 打开("example.txt", "r") content = file.读取() 打印(content) file.关闭()	打开文件并返回文件对象
关闭()	close()	file = 打开("example.txt", "r") content = file.读取() file.关闭()	关闭文件
读取()	read()	file = 打开("example.txt", "r") content = file.读取() 打印(content) file.关闭()	读取文件内容
删除()	remove()	numbers = [1, 2, 3, 2, 4] numbers.删除(2) 打印(numbers) # 输出: [1, 3, 2, 4]	移除列表中第一个匹配的元素
弹出()	pop()	numbers = [1, 2, 3] popped = numbers.弹出(1) 打印(popped) # 输出: 2 打印(numbers) # 输出: [1, 3]	移除并返回指定索引处的元素
查找索引()	index()	numbers = [1, 2, 3, 2, 4] index = numbers.查找索引(2) 打印(index) # 输出: 1	返回第一个匹配元素的索引
计数()	count()	numbers = [1, 2, 3, 2, 4] count = numbers.计数(2) 打印(count) # 输出: 2	返回元素在列表中的出现次数
排序()	sort()	numbers = [5, 2, 4, 1, 3] numbers.排序() 打印(numbers) # 输出: [1, 2, 3, 4, 5]	对列表进行排序
倒序()	reverse()	numbers = [1, 2, 3, 4, 5] numbers.倒序() 打印(numbers) # 输出: [5, 4, 3, 2, 1]	反转列表中的元素顺序

中文	英文	示例	说明
生成随机数()	random.random()	导入 random 打印(random.生成随机数()) # 输出: 0.2203627	来生成随机数
延迟()	time.sleep()	导入 time time.延迟(5) 打印('hello') # hello 会延迟 5 秒后输出	让程序停止一段时间
列出目录()	listdir()	path = r'D:/images' dirs = os.列出目录(path) 循环 file 在 dirs: 打印(file) # 输出 images 下所有文件列表	显示当前目录下的文件
字符型()	chr()	char = 字符型(65) 打印(char) # 输出: 'A'	返回指定 Unicode 代码的字符
取字符编码()	ord()	code = 取字符编码('A') 打印(code) # 输出: 65	返回字符的 Unicode 代码
转二进制()	bin()	binary = 转二进制(10) 打印(binary) # 输出: '0b1010'	将整数转换为二进制字符串
转十六进制()	hex()	hexadecimal = 转十六进制(16) 打印(hexadecimal) # 输出: '0x10'	将整数转换为十六进制字符串
转八进制()	oct()	octal = oct(8) 打印(octal) # 输出: '0o10'	将整数转换为八进制字符串
字节数组型()	bytearray()	my_array = 字节数组型([0, 1, 2, 3]) 打印(my_array) # 输出: b'\x00\x01\x02\x03'	创建一个可变的字节数组对象
字节集型()	bytes()	my_bytes = 字节集型([0, 1, 2, 3]) 打印(my_bytes) # 输出: b'\x00\x01\x02\x03'	创建一个不可变的字节数组对象
转 ASCII()	ascii()	text = "Hello, 你好" ascii_text = 转 ASCII(text) 打印(ascii_text) # 输出: 'Hello, \u4f60\u597d'	返回一个表示对象的可打印字符串

中文	英文	示例	说明
执行代码()	exec()	code = ''' 循环 i 在 生成范围(5): 打印(i) ''' 执行代码(code) # 输出: 0 1 2 3 4	执行动态生成的 Python 代码
格式化字符串()	format()	name = "Alice" age = 25 formatted = 格式化字符串("Name: {}, Age: {}", name, age) 打印(formatted) # 输出: "Name: Alice, Age: 25"	根据指定的格式进行字符串格式化
逻辑型()	bool()	打印(逻辑型(0)) # 输出: False 打印(逻辑型(1)) # 输出: True	将对象转换为布尔值（True/False）
设置断点()	breakpoint()	x = 5 设置断点() # 程序暂停进入调试器	在代码中插入断点，用于调试

类方法()	classmethod()	类 MyClass:
		    @类方法
		    函数 cls_method(cls):
		        返回 "Called"
		打印(MyClass.cls_method())
			将方法转换为类方法，自动传入类作为第一个参数

编译代码()	compile()	code = 编译代码('打印("Hello")', '', 'exec') 执行代码(code)	将源代码编译为代码对象，供 执行代码() 或 求值() 使用
复数型()	complex()	c = 复数型(3, 4) 打印(c) # 输出: (3+4j)	创建复数对象
取成员()	dir()	打印(取成员(字符串型)) # 列出 字符串型 类的所有属性和方法	返回对象的属性、方法列表
求值()	eval()	result = 求值("2 + 3 * 4") 打印(result) # 输出: 14	执行字符串形式的 Python 表达式并返回结果
固定集合型()	frozenset()	fs = 固定集合型([1, 2, 3]) 打印(fs) # frozenset({1, 2, 3})	创建不可变的集合对象

中文	英文	示例	说明
全局变量()	globals()	x = 10 打印(全局变量()['x']) # 输出: 10	返回当前全局符号表（字典）
是否有成员()	hasattr()	类 A: 跳过 a = A() 打印(是否有成员(a, 'name')) # False	检查对象是否具有指定名称的属性
取哈希值()	hash()	打印(取哈希值("hello")) # 输出一个整数（如 -1267278887）	返回对象的哈希值（用于字典、集合等）
取帮助()	help()	取帮助(取长度) # 显示 取长度() 函数的帮助文档	调出交互式帮助系统或显示对象文档
取对象 ID()	id()	a = [1, 2] 打印(取对象 ID(a)) # 输出内存地址（整数）	返回对象的唯一标识（内存地址）
是否为子类()	issubclass()	类 A: 跳过 类 B(A): 跳过 打印(是否为子类(B, A)) # True	检查一个类是否是另一个类的子类
创建迭代器()	iter()	lst = [1, 2, 3] it = 创建迭代器(lst) 打印(next(it)) # 1	返回可迭代对象的迭代器

取局部变量()	locals()	函数 func():
		    x = 5
		    打印(取局部变量())
		func() # {'x': 5}
			返回当前局部符号表（字典）

内存视图型()	memoryview()	data = b"hello" mv = 内存视图型(data) 打印(mv[0]) # 104 ('h' 的 ASCII)	创建内存视图对象，高效访问内存数据
取下一项()	next()	it = iter([10, 20]) 打印(取下一项(it)) # 10 打印(取下一项(it)) # 20	获取迭代器的下一个元素
对象型()	object()	obj = 对象型() 打印(obj) # <object object at 0x...>	创建一个基础对象（所有类的基类实例）

中文	英文	示例	说明
属性描述符()	property()	类 C: 函数 **init**(self): self._x = 0 x = 属性描述符(匿名函数 self: self._x) c = C() 打印(c.x) # 0	创建属性访问器（常用于类中）
取规范文本()	repr()	打印(取规范文本("hello")) # "'hello'"	返回对象的“官方”字符串表示（通常可 eval 还原）
静态方法()	staticmethod()	类 Math: @静态方法 函数 add(a, b): 返回 a + b 打印(Math.add(2, 3)) # 5	将方法定义为静态方法（不接收 self 或 cls）
父对象()	super()	类 Parent: 函数 greet(self): 返回 "Hi" 类 Child(Parent): 函数 greet(self): 返回 父对象().greet() + " 从 child" 打印(Child().greet()) # "Hi from child"	调用父类的方法或属性
元组型()	tuple()	t = 元组型([1, 2, 3]) 打印(t) # (1, 2, 3)	将可迭代对象转换为不可变元组
取变量()	vars()	类 A: 函数 init(self): self.x = 1 a = A() 打印(取变量(a)) # {'x': 1}	返回对象的 **dict** 属性（存储实例变量的字典）
异步迭代器()	aiter()	异步 函数 main(): ait = 异步迭代器(async_iterable) 打印(等待 异步下一项(ait)) asyncio.run(main())	返回异步可迭代对象的异步迭代器
异步下一项()	anext()	异步 函数 main(): ait = 异步迭代器(async_iterable) val = 等待 异步下一项(ait, "end") asyncio.run(main())	获取异步迭代器的下一个值，可设默认值