变量，常量

* 赋值运算符：

数据类型

* 字符串：
  + 方法：title()、lower()、upper()、lstrip()、rstrip()、strip()
  + f字符串 => format()方法
* 整型、浮点型
  + int()函数、float()函数
* 布尔类型：
* type()函数

列表：

* 查询：
* 修改：
* 添加
  + append($item)方法
  + insert($index, $item)方法
* 删除
  + del $list[$index]语句
  + pop()/pop($index)方法
  + remove($item)方法
* 排序
  + 永久排序：sort(reverse=True|False)方法
  + 临时排序：sorted(reverse=True|False)函数
* 反转：reverse()方法
* 列表长度：len()函数
* list()函数
* range($start\_include, $end\_exclude, $step)函数
* min()、max()、sum()函数
* 切片：$list[${from\_include}:${to\_exclude}:${step}]
  + 复制列表副本: $list[:]
* 元组，不可变列表：

字典：

* 查询：$dict[$key]
* 更新/新增：$dict[$key] = $value
* 删除：del $dict[$key]语句
* set()函数：返回一个不包含重复元素的集合

if语句：

* 比较运算符：
* 逻辑运算符：and、or
* 查询特定是否在(in)/不在(not in)列表中：`if $item in $list`, `if $item not in $list`
* 判断列表是否为空：`if $list`等价于`if len($list) > 0`

for语句

* 列表循环
  + for $item in $list
  + for $item in $slice
  + for $item in $tuple
* 字典循环
  + for $key, $value in {$dict}.items()
  + for $key in $dict
  + for $key in {$dict}.keys()
  + for $value in {$dict}.values()

元组：固定长度、不可变的Python对象序列，逗号分隔序列值。

* `($item1, $item2, ...)`标识、tuple()函数
* `$tuple[$index]`
* 运算符
* 运算符
* count()方法
* 拆包

# 变量交换

a, b = 1, 2

b, a = a, b

print(f"a={a}, b={b}") # a=2, b=1

# 遍历

seq = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]

"""

1, 2, 3

4, 5, 6

7, 8, 9

"""

for a, b, c in seq:

print(f"{a}, {b}, {c}")

# \*rest语法

x1, y1, z1, \*rest = (11, 12, 13, 14, 15, 16)

print(f"x={x1}, y={y1}, z={z1}, rest={rest}") # x=11, y=12, z=13, rest=[14, 15, 16]

x2, y2, z2, \*\_ = (1, 2, 3, 4, 5, 6)

print(f”x={x2}, y={y2}, z={z2}”) # x=1, y=2, z=3

列表：可变的Python对象序列。

* `[$item1, $item2, ...]`标识、list()函数
* `$list[$index]`
* append()方法、insert()方法
* pop()方法、remove()方法、del关键字
* in/not in关键字：检查列表中是否含有对应元素
* 运算符、extend()方法
* 运算符
* sort()方法
* count()方法
* 切片：`$list[$from\_index:$to\_index:$step]`

seq = [7, 2, 3, 7, 5, 6, 0, 1]

# 列表反转

print(seq) # [7, 2, 3, 7, 5, 6, 0, 1]

print(seq[::-1]) # [1, 0, 6, 5, 7, 3, 2, 7]

字典：哈希表、关联数组。

* `{$key1: $value1, $key2: $value2, ...}`标识、dict()函数
* 字典的键必须是不可变对象，即可以哈希化（通过hash()函数检查）
* `$dict[$key]`
* get()方法
* setdefault()方法
* pop()方法、del关键字
* update()方法：合并字典
* in/not in关键字：检查字典中是否含有对应键
* items()方法、keys()方法、values()方法

集合：无序且元素唯一的容器。

* `{$item1, $item2, ...}`标识、set()函数
* 集合的元素必须是不可变的，即可以哈希化（通过hash()函数检查）
* 当且仅当集合中的元素都相同时，集合才相等
* add($item)方法
* clear()方法
* remove($item)方法、pop()方法
* 集合操作常用方法

|  |  |  |
| --- | --- | --- |
| 方法 | 替代方法 | 描述 |
| x.union(y) | x | y | 集合x和y的并集 |
| x.update(y) | x |= y | 将x设置为集合x和y的并集 |
| x.intersection(y) | x & y | 集合x和y的交集 |
| x.intersection\_update(y) | x &= y | 将x设置为集合x和y的交集 |
| x.difference(y) | x – y | 在集合x中不在集合y中的元素 |
| x.difference\_update(y) | x -= y | 将x设置为在集合x中不在集合y中的元素 |
| x.symmetric\_difference(y) | x ^ y | 所有在集合x或y中，但不是同时在x、y中的元素 |
| x.symmetric\_difference\_update(y) | x ^= y | 将x设置为所有在集合x或y中，但不是同时在x、y中的元素 |
| x.issubset(y) |  | 集合x是否是集合y的子集 |
| x.issuperset(y) |  | 集合x是否是集合y的超集 |

推导式

* 列表：`[$expr *for* $val *in* $collection *if* $condition]`
* 集合：`{$expr *for* $val *in* $collection *if* $condition}`
* 字典：``{$key-expr: $value-expr *for* $val *in* $collection *if* $condition}`
* 嵌套

all\_data = [["John", "Emily", "Michael", "Mary", "Steven"],

["Maria", "Juan", "Javier", "Natalia", "Pilar"]]

# ['Steven']

print([name for names in all\_data for name in names if name.count("e") >= 2])

函数：代码组织和复用。

* def关键字声明函数，return关键字返回值
* 位置参数、关键字参数，关键字参数必须跟在位置参数后面
* 命名空间，本地命名空间、全局变量
* global关键字

a = None

def func\_global\_exam():

global a

a = [i \*\* 2 for i in range(1, 11)]

print(a)

func\_global\_exam() # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

print(a) # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

* 返回多个值

def func\_return\_multiple\_values(x):

# 返回的是元组(x, x\*\*2, x\*\*3)

return x, x\*\*2, x\*\*3

# 元组拆包

val, square, cube = func\_return\_multiple\_values(3)

print(f"value={val}, square={square}, cube={cube}") # value=3, square=9, cube=27

* 函数是对象

import re

def remove\_punctuation(val):

return re.sub("[!#?]", "", val)

# 将函数作为对象使用：去除空格，正则替换，首字母大写

clean\_ops = [str.strip, remove\_punctuation, str.title]

def clean\_strings(strValues, ops):

result = []

for strValue in strValues:

for op in ops:

strValue = op(strValue)

result.append(strValue)

return result

strings = [" Alabama ", "Georgia! ", "Georgia", "georgia", "FlOrIda", "south carolina##", "West virginia? "]

# ['Alabama', 'Georgia', 'Georgia', 'Georgia', 'Florida', 'South Carolina', 'West Virginia']

print(clean\_strings(strings, clean\_ops))

* 匿名（lambda）函数：使用关键字lambda定义，通过单个语句生成函数，其结果是返回值

equiv\_anon = lambda x: x \*\* 2

print(equiv\_anon(3)) # 9

exam\_list = ["abc", "abcd", "aabb", "aaaa", "eeffcc"]

# 根据字符串中不同字母的数量对一个字符串集合进行排序

exam\_list.sort(key=lambda x: len(set(list(x))))

print(exam\_list) # ['aaaa', 'aabb', 'abc', 'eeffcc', 'abcd']

* 任意数量的实参：`\*params`

def show\_names(\*names):

print(names)

show\_names("Remilia", "Flandre", "Cirno") # ('Remilia', 'Flandre', 'Cirno')

* 任意数量的关键字实参：`\*\*params`

def build\_profile(first, last, \*\*user\_info):

profile = dict({"first\_name": first.title(), "last\_name": last.title()})

for key, value in user\_info.items():

profile[key] = value

return profile

# {'first\_name': 'Remilia', 'last\_name': 'Scarlet', 'address': 'gensokyo', 'sex': 'female', 'age': 500}

print(build\_profile("remilia", "scarlet", address="gensokyo", sex="female", age=500))

* 迭代器协议，一种令对象可遍历的通用方式，`iter()`。迭代器是一种用于在上下文中向Python解释器生成对象的对象，大部分以列表或列表型对象为参数的方法都可以接收任意的迭代器对象。

some\_dict = {"a": 1, "b": 2, "c": 3}

# 迭代器

dict\_iterator = iter(some\_dict)

print(dict\_iterator) # <dict\_keyiterator object at 0x1055ae2c0>

print(tuple(dict\_iterator)) # ('a', 'b', 'c')

* 生成器是构造新的可遍历对象的一种非常简洁的方式，`yield`关键字代替`return`关键字。生成器表达式与列表推导式类似，只需将中括号替换成圆括号即可。

def squares(n=10):

# 等价于`return (i \*\* 2 for i in range(1, n+1))`

for i in range(1, n+1):

yield i \*\* 2

gen = squares()

print(gen) # <generator object squares at 0x105452820>

for n in gen: # 1 4 9 16 25 36 49 64 81 100

print(n, end=" ")

错误和异常处理：`try/excep/else/finally`

f = open(file\_path, 'w')

try:

write\_to\_file(f)

# 可以通过将多个异常类型写成元组的方式同时捕获多个异常（小括号是必不可少的）

# except (TypeError, ValueError):

except:

print("Failed")

else:

print("Succeeded")

finally:

f.close()

文件与操作系统

Python内置函数

* `range()`：数字序列
* `max()`、`min()`、`sum()`
* `tuple()`：元组
* `list()`：列表
* `dict()`：字典
* `set()`：集合
* `map()`
* `iter()`：生成一个迭代器
* `sorted()`：对序列临时排序。
* `enumerate()`：返回(index, value)元组的序列。
* `zip()`：将序列的元素配对，新建一个元组构成的列表，列表的长度由最短的序列决定。

# 行转列, 列转行

row1 = (1, 2, 3)

row2 = (4, 5, 6)

cols = list(zip(row1, row2))

"""

(1, 4)

(2, 5)

(3, 6)

"""

for col in cols:

print(col)

row3, row4 = zip(\*cols)

"""

(1, 2, 3)

(4, 5, 6)

"""

print(f"{row3}\n{row4}")

* `reversed()`：将序列的元素倒序排列。
* `hash()`：计算hash值，如果对象类型不可哈希化，则返回异常。
* `open()`：打开指定路径的文件

Python内置模块

* bisect模块：二分搜索，需要保证列表已经排序。
  + bisect()函数：找到元素应该被插入的位置。
  + insort()函数：将元素插入到相应的位置。
* collections模块
  + deque类：双端队列。
  + defaultdict()函数：具有默认值的字典。
* re模块：正则表达式
  + sub()函数
* functools模块
  + partial()函数：柯里化
* itertools模块