基础语法

变量命名

- 只能是一个词；
- 只能包含字母、数字和下划线；
- 不能以数字开头；
- 尽量描述包含的数据内容；
- 不要使用Python函数名和关键字；
- 不要使用Python的保留字符

In [ ]:

```
number = 50
url = 'https://www.baidu.com/'
number_list_2 = [50,10,30]
# 以上number，url和number_list_2都是变量名
```

In [ ]:

```
number
```

In [ ]:

```
# 删除变量 del
del number
```

In [ ]:

```
number
```

保留字符/关键字keyword

and	or	in	not
if	else	elif	break
continue	for	while	try
finally	except	def	classl
global	pass	return	exec
assert	from	import	raise
with	is	yield	async

打印 print()

In [ ]:

```
# 不带引号，计算机需读懂括号里的内容，并打印最终结果
print(1 + 2)
```

In [ ]:

```
# 带单/双引号，计算机无须理解，原样复述引号中内容
print('1 + 2 =')
print('Hello World!')
```

In [ ]:

```
# 同时打印原内容，和运算结果
print('1 + 2 =', 1+2)
```

In [ ]:

```
print(f'{1 + 2}')
print(f'{1 + 2 = }')   # python3.8的新写法
```

In [ ]:

```
# 当打印内容中有单引号时，需使用双引号
print("Let's go!")
```

注释

单行注释：以 # 开头

In [ ]:

```
# 这是单行注释
print('Hello World!')
```

多行注释：用三个单引号 ''' 或者三个双引号 """ 将注释括起来

In [ ]:

```
'''
这是多行注释，用三个单引号
这是多行注释，用三个单引号
这是多行注释，用三个单引号
'''
print('Hello World!')
```

In [ ]:

```
"""
这是多行注释，用三个双引号
这是多行注释，用三个双引号
这是多行注释，用三个双引号
"""
print("Hello World!")
```

基本数据类型

- Python3中有六个标准的数据类型
  - Number（数字）
  - String（字符串）
  - List（列表）
  - Dictionary（字典）
  - Tuple（元组）
  - Set（集合）

数字

数字类型

- Python3 支持 int、float、bool、complex（复数）。
- 在Python 3里，只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。

In [ ]:

```
# 赋值
a, b, c, d = 20, 5.5, True, 4+3j
# 查询变量所指的对象类型 type()
print(type(a), type(b), type(c), type(d))
```

数值运算

In [ ]:

```
5 + 4  # 加法+
```

In [ ]:

```
4.3 - 2 # 减法-
```

In [ ]:

```
3 * 7  # 乘法*
```

In [ ]:

```
2 / 4  # 除法，用单个左下划线“/”，得到一个浮点数
```

In [ ]:

```
2 // 4 # 除法，用两个左下划线“/”，得到一个整数
```

In [ ]:

```
17 % 3 # 取余%
```

In [ ]:

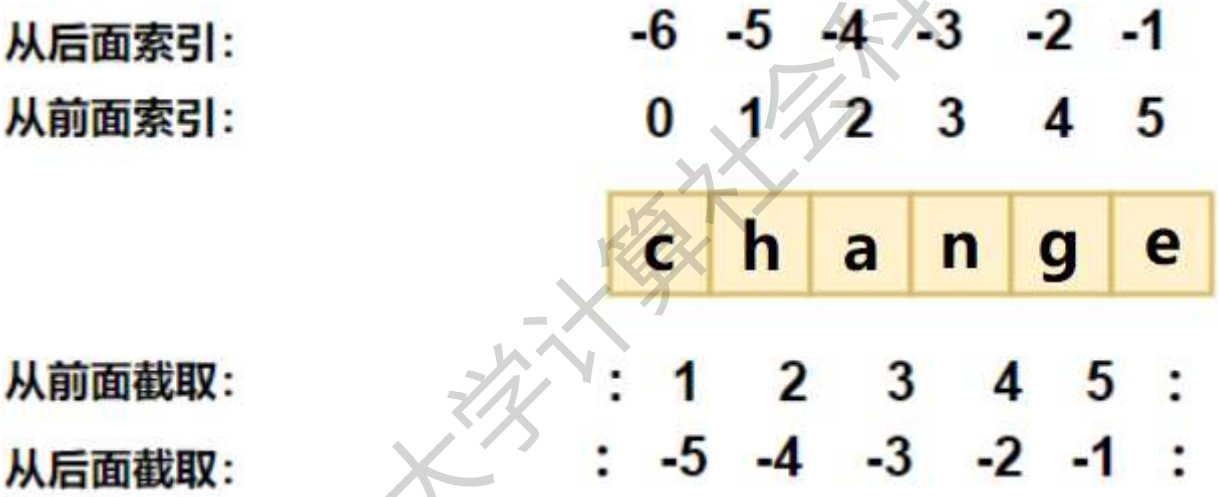
```
2 ** 5 # 乘方**
```

字符串

用单引号 ’ 或双引号 ″ 括起来，例如’Hello World!’

- 索引值以 0 为开始值，-1 为从末尾的开始位置
- 例如h从前面索引为1，从后面索引为-5

字符串索引说明图



In [ ]:

```
x = 'change'
print(x[2])
```

字符串运算

操作符	描述	例子
+	字符串连接	a+b
*	重复字符串	a*2
[]	通过索引获取字符串中的字符	a[1]
[:]	截取字符串中的一部分	a[1:4]
in	成员运算符 - 如果字符串中包含给定的字符返回 True	"h" in a
not in	成员运算符 - 如果字符串中不包含给定的字符返回 True	"M" not in a

注：a为hello，b为world

In [ ]:

```
a='change'
b='world'

print(a+b) # +号连接字符串
print(a*2) # *号重复字符串
print(a[1]) # []索引字符
print(a[1:4]) # [:]截取字符串，第1个到第3个
print("h" in a) # 是否包含
print("M" not in a) # 是否不包含
```

相关函数与方法 split, replace, strip, format

In [ ]:

```
# 查看内建函数
dir(str)
```

字符串相关函数（部分）

函数	描述	例子
split(str='', num=string.count(str))	以str为分隔符截取字符串，如果num有指定值，则分割为num+1个子字符串	str1.split('/')
strip()	去除字符串中的空格	str2.strip()
replace(old, new [, max])	将字符串中的str1替换成 str2,如果max指定，则替换不超过max次	str3.replace('中山市', '广州市')
'{a}{b}'.format(a=1,b=2)	填充字符串	'{a}'.format(a='2021年8月19日')

In [ ]:

```
# 分割 split(str='', num=string.count(str))
# str：分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等，亦可指定其他的
# num：分割次数，默认是-1，即分割所有
str1 = '人类简史：从动物到上帝/尤瓦尔·赫拉利/林俊宏/   中信出 版社  /2014-11/68.00元'
print(str1)

str1_list = str1.split('/')  # 默认分割所有，只要识别到分隔符，就分割
print(str1_list)

print(type(str1_list), '\n')   # 分割后，获得列表

str1_list_2 = str1.split('/',1)   # 分隔次数为2次（num+1）
print(str1_list_2)

print(len(str1_list))  # 查看列表长度
print(len(str1_list_2))
```

In [ ]:

```
# 去除空格 strip()
str2 = ' 中信出版社 '
print(str2)
print(str2.strip())
```

In [ ]:

```
# 替换 replace(old, new [, max])
str3 = '中山大学位于中山市。'   # 实际在广州
print(str3)
print(str3.replace('中山市', '广州市'))
```

In [ ]:

```
# 填充 '{a}{b}'.format(a=1,b=2)
# a和b为待填充的变量，在format中给a和b赋值
# 填充1个变量
str4 = '今天是：{a}'.format(a='2023年8月9日')
print(str4)

# 填充多个变量
str5 = '今天是：{a}，天气：{b}'.format(a='2023年8月9日',b='晴朗')
print(str5)
```

列表

- ** 写在方括号 [] 之间、用逗号分隔开的元素列表 **，元素可以是数字、字符串、甚至是列表等（即嵌套）
- 例子：
  - [1, 2, 3, 4]

- ['北京','上海','广州','深圳']
- [[1, 2, 3, 4], ['北京','上海','广州','深圳']]

列表运算

操作符	描述	例子
+	列表连接	a+b
*	重复列表	a*2
[]	通过索引获取列表中的元素	a[1]
[:]	截取列表中的一部分	a[1:4]
in	成员运算符 - 如果列表中包含给定的元素返回 True	2 in a
not in	成员运算符 - 如果列表中不包含给定的元素返回 True	2 not in a

In [ ]:

```
a=[1, 2, 3, 4]
b=['北京','上海','广州','深圳']

print(a+b) # +号连接列表
print(a*2) # *号重复列表
print(a[1]) # []索引列表中的元素
print(a[1:4]) # [:]截取列表中第1个到第3个元素
print(2 in a) # 是否包含
print(2 not in a) # 是否不包含
```

相关函数与方法 len() append()

函数	描述
len(list)	列表元素个数
max(list)	返回列表元素最大值
min(list)	返回列表元素最小值
list.append(obj)	在列表末尾添加新的对象
list.count(obj)	统计某个元素在列表中出现的次数
list.extend(seq)	在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
list.index(obj)	从列表中找出某个值第一个匹配项的索引位置
list.insert(index, obj)	将对象插入列表
list.pop([index=-1])	移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
list.remove(obj)	移除列表中某个值的第一个匹配项
list.reverse()	反向列表中元素
list.sort(key=None, reverse=False)	对原列表进行排序
list.clear()	清空列表

函数 描述

注：
- list为自设定的list名字，obj为特定的元素对象
- 例如b=['北京','上海','广州','深圳']中，list为b，obj则是其中任一元素，如“广州”

In [ ]:

```
# 列表元素个数 len()
b=['北京','上海','广州','深圳']
len(b)
```

In [ ]:

```
# 在列表末尾添加新的对象 append()
b=['北京','上海','广州','深圳']
b.append('青岛')
b
```

In [ ]:

```
# 生成空列表
c = []
c
```

字典

1. 写在花括号{}里，是无序的“键:值”(key:value)的集合
2. 键值对：即key:value，用冒号(:)分隔，每个键值对间用逗号(,)分隔
3. key和value的要求：在同一个字典中，key可以取任何数据类型，但必须是唯一的，且不可变的，例如字符串、数字；value不要求唯一
4. 字典格式：

d = {key1 : value1, key2 : value2, key3 : value3}

字典处理

In [ ]:

```
# 创建字典
dict0 = {}     # 创建空字典
dict1 = {'book_name':'人类简史：从动物到上帝','author':'尤瓦尔·赫拉利','publisher':'中

print(dict0)
print(dict1)
```

In [ ]:

```
# 访问字典里的值
print(dict1['book_name'])  # 访问book_name的值
print(dict1['author'])  # 访问author的值
```

In [ ]:

```
# 向字典添加/修改/删除键值对
# 添加键值对
dict1 = {'book_name':'人类简史：从动物到上帝','author':'尤瓦尔·赫拉利','publisher':'中
print(dict1)
dict1['translator'] = '林俊宏'
print(dict1)
```

In [ ]:

```
# 修改键值对
dict1['book_name'] = '人类简史'
print(dict1)
```

In [ ]:

```
# 删除键值对
del dict1['translator']
print(dict1)
```


相关函数与方法 len() items() keys() values()

函数	描述
len(dict)	计算字典元素个数，即键的总数
str(dict)	输出字典，以可打印的字符串表示
dict.clear()	删除字典内所有元素
dict.copy()	返回一个字典的浅复制

函数	描述
dict.fromkeys()	创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值
dict.get(key, default=None)	返回指定键的值，如果值不在字典中返回 default值
key in dict	如果键在字典dict里返回true，否则返回false
dict.items()	以列表返回可遍历的(键, 值) 元组数组
dict.keys()	返回一个迭代器，可以使用 list() 来转换为列表
dict.values()	返回一个迭代器，可以使用 list() 来转换为列表
dict.setdefault(key, default=None)	和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
dict.update(dict2)	把字典dict2的键/值对更新到dict里
pop(key[,default])	删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。否则，返回

In [ ]:

```
dict1 = {'book_name':'人类简史：从动物到上帝','author':'尤瓦尔·赫拉利','publisher':'中
print(dict1)
```

In [ ]:

```
# 查看字典长度 len()
len(dict1)
```

In [ ]:

```
# 查看所有键值对 items()
dict1.items()
```

In [ ]:

```
# 遍历并打印所有键值对
for key, value in dict1.items():
    print(key,':', value)
```

In [ ]:

```
# 查看所有的键  keys()
dict1.keys()
```

In [ ]:

```
# 查看所有的值 values()
dict1.values()
```

In [ ]:

```
# 以字典为元素的列表——json文件格式
dict2 = {'book_name':'理想国','author':'柏拉图','publisher':'商务印书馆'}
dict3 = {'book_name':'社会契约论','author':'卢梭','publisher':'商务印书馆'}
dict_list = [dict1, dict2, dict3]
dict_list    # 即json数据格式，是一种网页数据的存储格式，爬虫时涉及
```

元组

1. ** 写在小括号()里，元素之间用逗号隔开 **，元素可以是字符串、数字等
2. 例子：
   - ('张三','男',25)
   - (1, 2, 3, 4, 5)
3. 与列表的对比：
   - 二者相似，不同之处在于列表元素可修改，元组元素** 不可修改 **

In [ ]:

```
tup1 = ('张三','男',25)
print(tup1)

tup2 = (1, 2, 3, 4, 5 )
print(tup2)

print(type(tup1))
```

集合

- ** 写在花括号{}里，元素之间用逗号隔开 **，元素不重复、无序

In [ ]:

```
# 创建集合
set1 = {'上海', '北京', '广州', '深圳'}  # 直接用花括号
print(set1)

set2 = set('abcde') # 用set()
print(set2)

set3 = set() # 空集合，只能用set()，因为{}是用来创建空字典的
print(set3)
```

In [ ]:

```
# 删除重复元素
set4 = {'北京', '广州', '上海', '广州', '深圳', '广州'}
set4
```

In [ ]:

```
# 元素关系测试
set1 = {'上海', '北京', '广州', '深圳'}
'广州' in set1
```

循环语句：while和for

1. 主要包括while和for两种类型
2. 注意** 冒号和缩进 **
3. while和for可以相互嵌套，形成多层嵌套的循环

while语句

- 在给定的判断条件为 true 时执行循环体，否则退出循环体

while condition（条件）：
statements（代码块）

In [ ]:

```
i = 1
while i < 10:
    print(i)
    i = i + 2
```

for语句

1. 遍历任何序列的项目（如列表、字符串），执行循环体
2. 如果遍历数字序列，可使用内置range()函数

for variable in sequence:
statement（代码块）
(注：for 迭代对象 in 任何序列)

In [ ]:

```
b=['北京','上海','广州','深圳']
for j in b:
    print(j)
```

In [ ]:

```
for k in range(1,5):  # 1到4的数字序列
    print(k)
```

其他语句 break continue else pass

1. break语句：

   在语句块执行过程中终止循环，并且 跳出整个循环

In [ ]:

```
for i in range(0,5):
    print(i)
```

In [ ]:

```
for i in range(0,5):
    if i==3:
        break
    print(i)
```

2. continue语句：

   在语句块执行过程中 结束本次循环，回到循环语句的开头，执行下一次循环

In [ ]:

```
for i in range(0,5):
    if i==3:
        continue
    print(i)
```

3. else 语句：

   当条件不满足之后， 结束循环，执行else语句

In [ ]:

```
for i in range(0,5):
    if i==3:
        print(i)
    else:
        print('不等于3')
```

4. pass语句：

   空语句， 什么也不做，只起到占位作用，循环中使用pass不会跳出循环

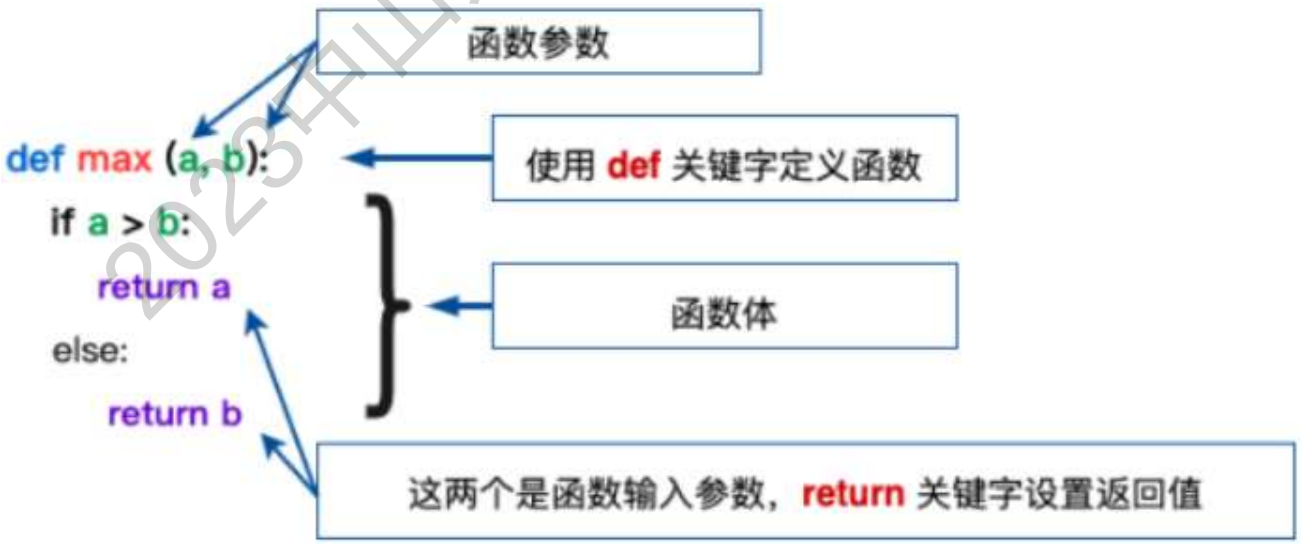
In [ ]:

```
for i in range(0,5):
    if i==3:
        pass
    print(i)
```

函数：输入参数，得到返回值

1. 函数是组织好的，可重复使用的，用来实现单一，或相关联功能的** 代码段 **
2. 好处：函数能提高应用的模块性，和代码的重复利用率
3. 函数类型：
   - 内建函数：Python自带的函数，例如print
   - 自定义函数：我们自己定义的函数
4. 格式如下：

```
def 函数名（参数列表）：
    函数体
```



(注：图片来源于https://www.runoob.com/python3/python3-function.html)

5. ** 定义函数的规则 **：

- 函数代码块** 以def关键词开头，后接函数标识符名称和圆括号 () **。
- 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容** 以冒号 ： 起始，并且缩进 **。
- return [表达式] 结束函数，选择性地** 返回一个值 **给调用方，不带表达式的 return 相当于返回 ** None **。

In [ ]:

```
def hello():  # 定义hello函数，使其打印Hello World!
    print("Hello World!")

hello()       # 调用hello函数
```

In [ ]:

```
# 【函数】中加入【参数】
def get_max(a, b):      # 定义get_max函数，输入a和b参数，使其返回最大值
    if a > b:
        return a
    else:
        return b
```

In [ ]:

```
# 调用get_max函数
a = 3
b = 4
print(get_max(a, b))
```

文件

打开文件 open()

1. 打开文件用open()，返回一个file对象，格式如下：

```
open(file, mode, encoding=utf-8,newline='')
```

- file：要访问的文件名称的字符串值
- mode：文件打开模式，包括只读、写入、追加等，可选，默认模式为只读(r)
- encoding：编码类型，一般使用utf-8
- newline：区分换行符

2. mode参数，常用r,w+,a+
   - ** r模式 **：以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
   - ** w+模式 **：**打开文件用于读写。如果该文件已存在则打开文件，并** 从开头开始编辑，即原有内容会被删除 **。如果该文件不存在，创建新文件。
   - ** a+模式 **：**打开一个文件用于读写，为** 追加模式 **。如果该文件已存在，文件指针将会放在文件的结尾，** 从结尾写入新的数据 **。如果该文件不存在，创建新文件用于读写。

文件方法

函数	描述
f.read(size)	读取一个文件的内容，size为读取的字节数，默认为-1，即读取整个文件
f.readline()	从文件中读取单独的一行（指针所在行）
f.readlines()	读取文件的所有行
f.write(string)	将string写入文件中
f.tell()	返回文件当前位置
f.seek()	移动文件读取指针到指定位置
f.close()	关闭文件

In [ ]:

```
# 写入文件
# 打开文件
file1 = open('output/test.txt', 'w+')    # 如果改成a+
# 写入内容
file1.write('Python 是一门非常好学的语言。\n没错，很简单！！\n')    # \n为换行符，实现换
# 关闭文件
file1.close()
```

In [ ]:

```
# 读取文件
# 打开文件
file2 = open('output/test_2.txt', 'r')
# 读取所有行
for line in file2.readlines():
    print(line)
# 关闭文件
file2.close()
```

实例：csv文件

1. 有一些类型的文件需要用到专门的包，例如csv文件用csv或pandas，excel文件用xlrd，json文件用json
2. 这些包有自己的特殊参数和方法，例如csv的标题行参数fieldnames、word的段落方法paragraphs

文件类型	工具包
txt	--
csv	csv, pandas
excel	xlrd, pandas, openpyxl

** csv文件 **:

- 以纯文本形式存储表格数据的简单文件格式，** 用逗号分隔，但能够呈现像excel表格样式的数据 **

In [ ]:

```
dict3_list = [
    {'book_name':'小王子', 'authors':'圣埃克苏佩里', 'publisher':'人民文学出版社', 'pub_
    {'book_name':'局外人', 'authors':'阿尔贝·加缪', 'publisher':'上海译文出版社', 'pub_
    {'book_name':'人类简史 : 从动物到上帝', 'authors':'尤瓦尔·赫拉利', 'publisher':'中
    {'book_name':'理想国', 'authors':'柏拉图', 'publisher':'商务印书馆', 'pub_time':'198
]
print(dict3_list)
print(len(dict3_list))
```

In [ ]:

```
import csv

# 打开文件
file3 = open(file='output/csv_sample.csv',mode='a+',encoding='utf-8',newline='')
# 文件名为csv_sample.csv，模式为a+，如果已有文件，在末尾追加，如果没，则生成新文件
# 编码为utf-8，需要区分换行符

# 设置标题行fieldnames
fieldnames = ['book_name', 'authors', 'publisher', 'pub_time']
# 要求以字典的形式写入数据，fieldnames注明字典中键的名称，也就是书名、作者这些标题名
writer = csv.DictWriter(file3, fieldnames=fieldnames)
# 将fieldnames设置的标题key写入首行
writer.writeheader()
```

In [ ]:

```
# 循环写入字典列表
for dict3 in dict3_list:
    writer.writerow(dict3)    # 写入一行

file3.close()   # 关闭文件
```

查看帮助文档：help或?

In [ ]:

```
import requests
# 显示在新的窗口
?requests
```

In [ ]:

```
help(requests) # 显示在jupyter notebook的界面中，作为output
```

END