COMP9021 整理资料

基本知识

In [ ]:

```
# 不可变数据（3 个）：Integer（数字）、String（字符串）、Tuple（元组）；
# 可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）

# 二进制
bin(1000) # 转换成二进制

# 多变量赋值
a = b = c = 1
a, b, c = 1, 2, "runoob"
# 输出各自的类型
print(type(a), type(b), type(c))
# 判断a的类型是不是int类型
isinstance(a, int)
# 数值运算
5 + 4  # 加法
4.3 - 2 # 减法
3 * 7  # 乘法
2 / 4  # 除法，得到一个浮点数
2 // 4 # 除法，得到一个整数
17 % 3 # 取余
2 ** 5 # 乘方
# 尽量使用divmod方法
a,b = divmod(100,3)

# 字符串常用操作
str = 'Runoob'
print (str)           # 输出字符串
print (str[0:-1])     # 输出第一个到倒数第二个的所有字符
print (str[0])        # 输出字符串第一个字符
print (str[2:5])      # 输出从第三个开始到第五个的字符
print (str[2:])       # 输出从第三个开始的后的所有字符
print (str * 2)       # 输出字符串两次
print (str + "TEST") # 连接字符串

# \表示转义字符
print('Ru\noob')

# 字符串常用方法
capitalize()              # 将字符串的第一个字符转换为大写
center(width, fillchar) # 返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，
默认为空格。
count(str, beg= 0,end=len(string)) #返回 str 在 string 里面出现的次数，
# 如果 beg 或者 end 指定则返回指定范围内 str 出现的次数
encode(encoding='UTF-8',errors='strict') # 以 encoding 指定的编码格式编码字符串，
# 如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace'
endswith(suffix, beg=0, end=len(string)) # 检查字符串是否以 obj 结束，
# 如果beg 或者 end 指定则检查指定的范围内是否以 obj 结束，如果是，返回 True,否则返回 False.
expandtabs(tabsize=8)   # 把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是
 8 。
find(str, beg=0 end=len(string)) # 检测 str 是否包含在字符串中，如果指定范围 beg
# 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1
index(str, beg=0, end=len(string)) # 跟find()方法一样，只不过如果str不在字符串中会报一
个异常.
isalnum()                 # 如果字符串至少有一个字符并且所有字符都是字母或数字则返 回 True,
否则返回 False
isalpha()                 # 如果字符串至少有一个字符并且所有字符都是字母则返回 True, 否则返
回 False
isdigit()                 # 如果字符串只包含数字则返回 True 否则返回 False..
islower()                 # 如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写
```

```
的)字符都是小写,
# 则返回 True, 否则返回 False
isnumeric()             # 如果字符串中只包含数字字符, 则返回 True, 否则返回 False
isspace()               # 如果字符串中只包含空白, 则返回 True, 否则返回 False.
istitle()               # 如果字符串是标题化的(见 title())则返回 True, 否则返回 False
isupper()               # 如果字符串中包含至少一个区分大小写的字符, 并且所有这些(区分大小写
的)字符都是大写,
# 则返回 True, 否则返回 False
join(seq)               # 以指定字符串作为分隔符, 将 seq 中所有的元素(的字符串表示)合并为
一个新的字符串
len(string)             # 返回字符串长度
ljust(width[, fillchar]) # 返回一个原字符串左对齐,并使用 fillchar 填充至长度 width 的新
字符串, fillchar 默认为空格。
lower()                 # 转换字符串中所有大写字符为小写.
lstrip()                # 截掉字符串左边的空格或指定字符。
maketrans()             # 创建字符映射的转换表, 对于接受两个参数的最简单的调用方式, 第一个
参数是字符串,
# 表示需要转换的字符, 第二个参数也是字符串表示转换的目标。
max(str)                # 返回字符串 str 中最大的字母。
min(str)                # 返回字符串 str 中最小的字母。
replace(old, new [, max]) # 把 将字符串中的 str1 替换成 str2,如果 max 指定, 则替换不超
过 max 次。
rfind(str, beg=0,end=len(string)) # 类似于 find()函数, 不过是从右边开始查找.
rindex( str, beg=0, end=len(string)) # 类似于 index(), 不过是从右边开始.
rjust(width,[, fillchar]) # 返回一个原字符串右对齐,并使用fillchar(默认空格）填充至长度 w
idth 的新字符串
rstrip()                # 删除字符串字符串末尾的空格.
split(str="", num=string.count(str)) # num=string.count(str)) 以 str 为分隔符截取字
符串, 如果 num 有指定值,
# 则仅截取 num 个子字符串
splitlines([keepends])  # 按照行('\r', '\r\n', \n')分隔, 返回一个包含各行作为元素的列
表,
# 如果参数 keepends 为 False, 不包含换行符, 如果为 True, 则保留换行符。
startswith(str, beg=0,end=len(string)) # 检查字符串是否是以 obj 开头, 是则返回 True,
# 否则返回 False。如果beg 和 end 指定值, 则在指定范围内检查。
strip([chars])          # 在字符串上执行 lstrip()和 rstrip()
swapcase()              # 将字符串中大写转换为小写, 小写转换为大写
title()                 # 返回"标题化"的字符串,就是说所有单词都是以大写开始, 其余字母均为小
写(见 istitle())
translate(table, deletechars="") # 根据 str 给出的表(包含 256 个字符)转换 string 的字
符,
# 要过滤掉的字符放到 deletechars 参数中
upper()                 # 转换字符串中的小写字母为大写
zfill (width)           # 返回长度为 width 的字符串, 原字符串右对齐, 前面填充0
isdecimal()             # 检查字符串是否只包含十进制字符, 如果是返回 true, 否则返回 fals
e。

# 列表操作
list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]
tinylist = [123, 'runoob']

print (list)            # 输出完整列表
print (list[0])         # 输出列表第一个元素
print (list[1:3])       # 从第二个开始输出到第三个元素
print (list[2:])        # 输出从第三个元素开始的所有元素
print (tinylist * 2)    # 输出两次列表
print (list + tinylist) # 连接列表
# 常用方法
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(cmp=None, key=None, reverse=False) # 对原列表进行排序
list.clear()             # 清空列表
list.copy()              # 复制列表


# 元组操作
tuple = ( 'abcd', 786 , 2.23, 'runoob', 70.2  )
tinytuple = (123, 'runoob')

print (tuple)             # 输出完整元组
print (tuple[0])          # 输出元组的第一个元素
print (tuple[1:3])        # 输出从第二个元素开始到第三个元素
print (tuple[2:])         # 输出从第三个元素开始的所有元素
print (tinytuple * 2)     # 输出两次元组
print (tuple + tinytuple) # 连接元组

# 集合操作
student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}
print(student)   # 输出集合，重复的元素被自动去掉
# 成员测试
if 'Rose' in student :
    print('Rose 在集合中')
else :
    print('Rose 不在集合中')
# set可以进行集合运算
a = set('abracadabra')
b = set('alacazam')
print(a)
print(a - b)     # a和b的差集
print(a | b)     # a和b的并集
print(a & b)     # a和b的交集
print(a ^ b)     # a和b中不同时存在的元素

# 字典操作
dict = {}
dict['one'] = "1"
dict[2]     = "2"
tinydict = {'name': 'runoob','code':1, 'site': 'www.runoob.com'}
print (dict['one'])       # 输出键为 'one' 的值
print (dict[2])           # 输出键为 2 的值
print (tinydict)          # 输出完整的字典
print (tinydict.keys())   # 输出所有键
print (tinydict.values()) # 输出所有值

# 类型转换
int(x [,base])            # 将x转换为一个整数
float(x)                  # 将x转换到一个浮点数
complex(real [,imag])     # 创建一个复数
str(x)                    # 将对象 x 转换为字符串
repr(x)                   # 将对象 x 转换为表达式字符串
eval(str)                 # 用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)                  # 将序列 s 转换为一个元组
```

```
list(s)                     # 将序列 s 转换为一个列表
set(s)                      # 转换为可变集合
dict(d)                     # 创建一个字典。d 必须是一个序列 (key,value)元组。
frozenset(s)                # 转换为不可变集合
chr(x)                      # 将一个整数转换为一个字符
ord(x)                      # 将一个字符转换为它的整数值
hex(x)                      # 将一个整数转换为一个十六进制字符串
oct(x)                      # 将一个整数转换为一个八进制字符串

# 字典方法
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
radiansdict.clear()         # 删除字典内所有元素
radiansdict.copy()          # 返回一个字典的浅复制
radiansdict.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
radiansdict.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
radiansdict.items()         # 以列表返回可遍历的(键, 值) 元组数组
radiansdict.keys()          # 返回一个迭代器，可以使用 list() 来转换为列表
radiansdict.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
radiansdict.update(dict2) # 把字典dict2的键/值对更新到dict里
radiansdict.values()        # 返回一个迭代器，可以使用 list() 来转换为列表
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default值。
popitem()                   # 随机返回并删除字典中的一对键和值(一般删除末尾对)。
```

基本算法

In [ ]:

```
# 进制转换
def decode(num,b):
    return ((num == 0) and  "0" ) \
         or (decode(num // b, b).lstrip("0") + "0123456789"[num % b])

# 将一个数字转换成3进制
print(decode(586,3))
```

In [ ]:

```
# 获取给定数字所有的除数
def get_all_divisor(n):
    if n == 1:
        return [1]
    result = set([])
    for i in range(2, int(sqrt(n)) + 1):
        if n % i == 0:
            result.add(i)
            result.add(n // i)
    return result
```

In [ ]:

```
#  求质数
from math import sqrt
from timeit import timeit

def first_sieve_of_primes_up_to(n):
    sieve = list(range(2, n + 1))
    i=0
    while sieve[i] <= round(sqrt(n)):
        k=0
        while True:
            factor = sieve[i] * sieve[i + k]
            if factor > n:
                break
            while factor <= n:
                sieve.remove(factor)
                factor *= sieve[i]
            k += 1
        i += 1
    return sieve

def second_sieve_of_primes_up_to(n):
    sieve = list(range(2, n + 1))
    i=0
    while sieve[i] <= round(sqrt(n)):
        sieve_as_set = set(sieve)
        k=0
        while True:
            factor = sieve[i] * sieve[i + k]
            if factor > n:
                break
            sieve_as_set.remove(factor)
            k += 1
        sieve = sorted(sieve_as_set)
        i += 1
    return sieve

# 利用byte求质数
def get_primes_3(n):
    """ Returns  a list of primes < n for n > 2 """
    sieve = bytearray([True]) * (n//2)
    for i in range(3,int(n**0.5)+1,2):
        if sieve[i//2]:
            sieve[i*i//2::i] = bytearray((n - i*i-1)//(2 * i) + 1)

    return [2, *compress(range(3, n, 2), sieve[1:])]
```

In [ ]:

```
# 判断某一个值是否是质数
from math import sqrt

def is_prime(n):
    # Only used to test odd numbers.
    return all(n % d for d in range(3, round(sqrt(n)) + 1, 2))
print('The solutions are:\n')
# The list of all even i's such that a + i is one of a, b, c, d, e, f.
good_leaps = tuple(sum(range(0, k, 2)) for k in range(2, 13, 2))
for a in range(10_001, 100_000 - good_leaps[-1], 2):
    # i should be in good_leaps iff a + i is prime for i = 0, 2, 4, ..., 30.
    if all(((i in good_leaps) == is_prime(a + i)) for i in range(0, good_leaps[-
1] + 1, 2)):
        for i in good_leaps[: -1]:
            print(a + i, end = '  ')
        print(a + good_leaps[-1])
```

In [ ]:

```
# 测试二进制转化，计算
def check(number,digits_seen_before):
    while number:
        number,digit = divmod(number,10)
        digits_seen_so_far = digits_seen_before | 1 << digts
        if digits_seen_so_far == digits_seen_before:
            return
        digits_seen_before = digits_seen_so_far
    return digits_seen_before
```

In [ ]:

```
# 字符串format的方式
# 通过关键字
print('{名字}今天{动作}'.format(名字='陈某某',动作='拍视频'))#通过关键字
grade = {'name' : '陈某某', 'fenshu': '59'}
print('{name}电工考了{fenshu}'.format(**grade))#通过关键字，可用字典当关键字传入值时，在字典前加**即可

# 通过位置
print('{1}今天{0}'.format('拍视频','陈某某'))#通过位置
print('{0}今天{1}'.format('陈某某','拍视频'))

# 填充和对齐^<>分别表示居中、左对齐、右对齐，后面带宽度
print('{:^14}'.format('陈某某'))
print('{:>14}'.format('陈某某'))
print('{:<14}'.format('陈某某'))
print('{:*<14}'.format('陈某某'))
print('{:&>14}'.format('陈某某'))#填充和对齐^<>分别表示居中、左对齐、右对齐，后面带宽度

# 精度
print('{:.1f}'.format(4.234324525254))
print('{:.4f}'.format(4.1))

# 进制转化，b o d x 分别表示二、八、十、十六进制
print('{:b}'.format(250))
print('{:o}'.format(250))
print('{:d}'.format(250))
print('{:x}'.format(250))

# 千分位分隔符，这种情况只针对与数字
print('{:,}'.format(100000000))
print('{:,}'.format(235445.234235))
```

In [ ]:

```
# 格式化输出杨辉三角等图形时，推荐采用逐行输出的方式
# 逐行打印输出
intervals = [0] * 4
for e in L:
    intervals[e // 5] += 1
for i in range(4):
    if intervals[i] == 0:
        print('There is no element', end = ' ')
    elif intervals[i] == 1:
        print('There is 1 element', end = ' ')
    else:
        print(f'There are {intervals[i]} elements', end = ' ')
    print(f'between {5 * i} and {5 * i + 4}.')
```

In [ ]:

```
# 给定的解决方案，主要是用于除法的各种题型
# multiplications
for x in range(100, 1_000):
    for y in range(10, 100):
        product0 = x * (y % 10)
        if product0 < 1_000:
            continue
        product1 = x * (y // 10)
        if product1 >= 1_000:
            continue
        total = product0 + 10 * product1
        if total >= 10_000:
            continue
        the_sum = x % 10 + y % 10 + product0 % 10 + total % 10
        if x // 10 % 10 + y // 10 + product0 // 10 % 10 + product1 % 10 + total
// 10 % 10 !=\

the_sum:
            continue
        if x // 100 + product0 // 100 % 10 + product1 // 10 % 10 + total // 100
% 10 != the_sum:
            continue
        if product0 // 1_000 + product1 // 100 + total // 1_000 == the_sum:
            print(f'{x} * {y} = {total}, all columns adding up to {the_sum}.')
```

In [ ]:

```
# perfect numbers
import sys
try:
    N = int(input('Input an integer: '))
except ValueError:
    print('Incorrect input, giving up.')
    sys.exit()

for i in range(2, N + 1):
    # 1 divides i, so counts for one divisor.
    # It is enough to look at 2, ..., i // 2 as other potential divisors.
    if 1 + sum(j for j in range(2, i // 2 + 1) if i % j == 0) == i:
        print(i, 'is a perfect number.')
```

补充题库