计算思维通识教育

Computational Thinking

第2章 计算设备处理信息-使用编程语言

主讲人：曹轶臻

联系方式：caoyizhen@cuc.edu.cn

Program as you think

CONTENTS

计算机与网络空间安全学院
School of Computer and Cyber Sciences

计算思维通识教育
Computational Thinking

02

编程语言的基本规则

Python

计算机与网络空间安全学院

School of Computer and Cyber Sciences

2-1 编程的基本方法

你为什么要编程？

明确动机和方向，有强烈的学习欲望。首先，要想明白自己学习编程的强烈动机，明确定位，为的是能够让你坚持下来。比如：

1. 通过编程开发脚本，来重复执行某些操作，解放双手，节省时间，提升工作效率。
2. 通过编程，解决自己专业中遇到的复杂问题，或做出符合时代的作品。
3. 想从事这方面的工作，进入IT、互联网、人工智能等行业。
4. 其他……

2-1 编程的基本方法

编程的工具

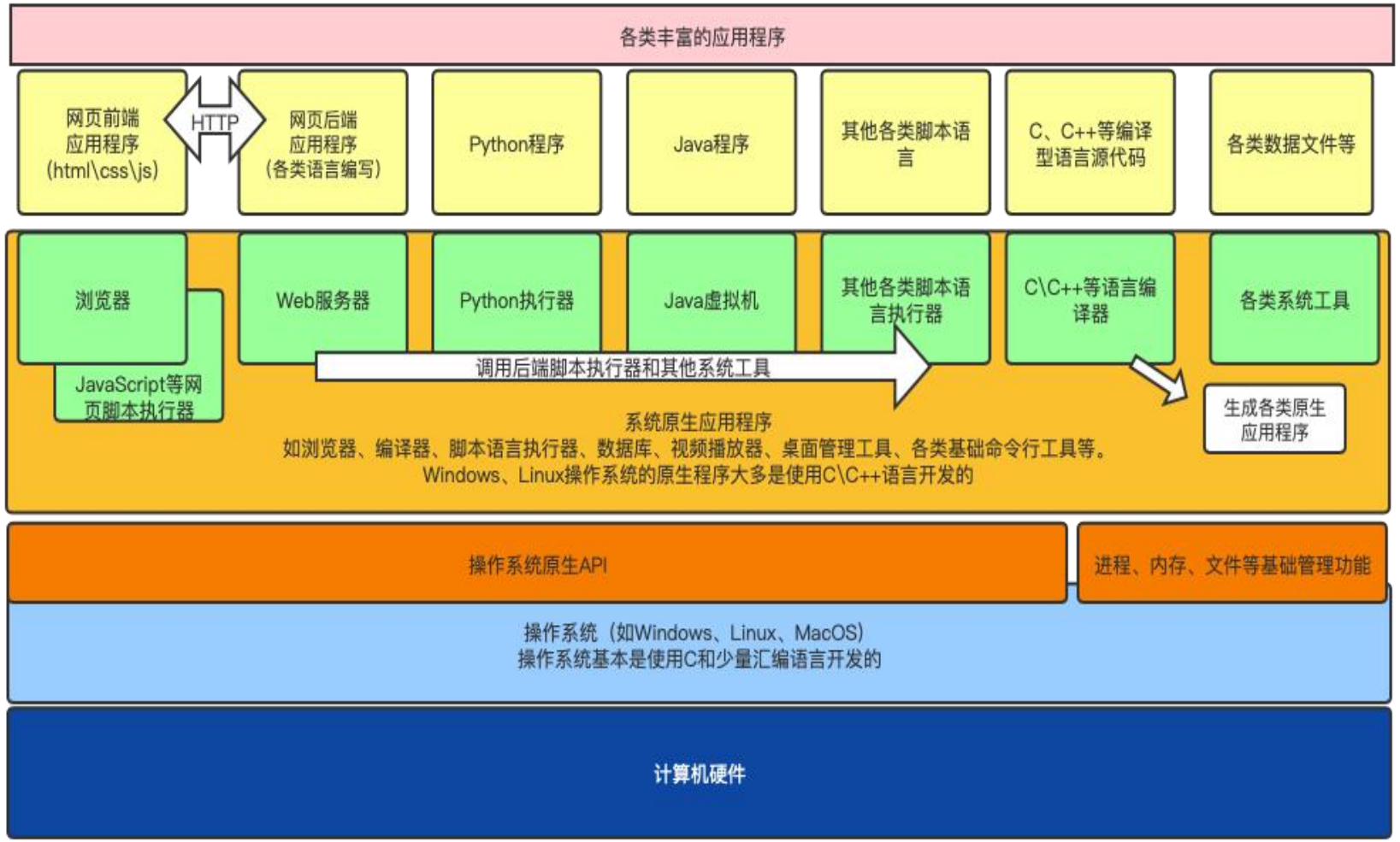
编程工具方面要学会使用以下几类工具：

文本编辑器 Editor	最好有代码高亮和代码提示
调试器 Debugger	很重要和基础的工具
各种程序语言的编译器和解释器 Compiler & Interpreter	将高级语言转成机器可以执行的目标代码
集成开发环境 IDE	集成了上述三种工具等其他辅助工具。
搜索引擎 Search Engine	最基本的就是查阅文档
AI助手 豆包、Kimi、通义灵码等	提供语法建议，解答编程问题，提供编程思路和算法设计，进行代码重构和优化...

2-1 编程的基本方法

编程学习的方向

- 深入学习JavaScript、HTML和CSS。具备Web前端开发能力。
- 学习Python、Java等语言，具备数据分析，科学计算，Web后端等综合开发能力。
- 深入底层，学习C语言、数据结构、计算机体系架构、计算机网络等课程。具备系统级的高级开发能力。



2-1 变量与数据类型

变量（variable）用于在内存中存储数据，用变量的名字就可以访问这块内存中存储的数据，也就是变量的值（value）。

将变量视为一个容器，其中包含可以稍后在程序中更改的数据。
你可以把变量想象成一个用来存放书籍的袋子，并且可以随时更换那本书。

```
number = 10
number = 1.1
```

最初变量number的值为10，
后来改为1.1

```
a, b, c = 5, 3.2, "Hello"
print (a)
print (b)
print (c)
```

为多个变量分配多个值。
变量不仅可以是数字，
还可以是任意数据类型。

2-1 变量与数据类型

程序中的每个值都有一个数据类型。由于在Python中一切都是对象（object）。因此数据类型实际上是类，变量是这些类的实例（对象）。

1. 数字 整数、浮点数在 Python 中被定义为 int、float 类。

```
number = 1
print(number, type(number))
number = 1.0
print(number, type(number))
```

```
1 <class 'int'>
1.0 <class 'float'>
```

1 是整数，1.0 是浮点数

int	float
10	0.0
0o100	15.20
-786	-21.9
80	32.3e+18
-0490	-90.
-0x260	-32.54e100
0x69	70.2E-12

2-1 变量与数据类型

2. 字符串

字符串(String)是Unicode字符序列。使用单引号或双引号来表示字符串。

```
s1 = 'This is a string'
s2 = "这是一个字符串"
s3 = '"Good night",she said'
s4 = '''this is a tiny poetry
and,this is the end'''
```

多行字符串可以使用三引号 ''' 或 """ 来表示

P	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
>>> s = 'abcdef'
>>> s[1:5]
'bcde'
```

可以使用 [头下标:尾下标] 来截取相应的字符串，其中下标从 0 开始

```
str = 'Hello World!'

print(str)  # 输出完整字符串
print(str[0])  # 按索引位置输出字符串中的字符
print(str[2:5])  # 输出字符串中第三个至第六个之间的字符串
print(str[2:])  # 输出从第三个字符开始的字符串
print(str * 2)  # 输出字符串两次
print(str + "TEST")  # 用加号连接字符串
```

2-1 变量与数据类型

3. 列表

列表（List）是有序的元素序列，是 Python 中最常用的数据类型之一，非常灵活。列表中的所有元素不需要属于同一类型（复合数据类型）。

用逗号分隔元素，用括号 [ ] 括起来。

```
a = [1, 2.2, 'python']
```

变量 [头下标:尾下标] 获得列表的切片（或子串），从左到右索引默认 0 开始

```
a = [5,10,15,20,25,30,35,40]
print("a[2] = ", a[2])
print("a[0:3] = ", a[0:3])
print("a[5:] = ", a[5:])
```

```
a[2] = 15
a[0:3] = [5, 10, 15]
a[5:] = [30, 35, 40]
```

```
a = [1, 2, 3]
a[2] = 4
print(a)
```

```
[1, 2, 4]
```

列表是可变的，即列表元素的值可以更改。

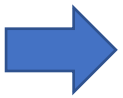
4. 元组

元组（Tuple）是与列表相同的元素的有序序列。唯一的区别是元组是不可变的，一旦创建就无法修改。相当于只读列表。

```
t = (5,'program', [1,2,3])
```

它在括号 () 中定义，其中项目用逗号分隔。

```
list = [ 'hello', 589, 2.11, 'world', 30.5 ]
tuple = ( 'hello', 589, 2.11, 'world', 30.5 )
list[2] = 1000      # 列表是可以修改的
print(list)
tuple[2] = 1000     # 元组的元素只读，不能修改
print(tuple)
```



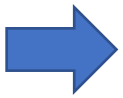
```
['hello', 589, 1000, 'world', 30.5]
Traceback (most recent call last):
  File "C:\pythonProject\datatype.py", line 5, in <module>
    tuple[2] = 1000   # 元组的元素只读不能修改
TypeError: 'tuple' object does not support item assignment
```

计算机与网络空间安全学院
School of Computer and Cyber Sciences

5. 集合

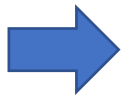
集合（**Set**）是一个无序、不重复的元素集。
Set 由大括号 { } 内用逗号分隔的值定义。

```
a = {1, 2, 2, 3, 3, 3}
print(a)
```



```
{1, 2, 3}
```

```
a = {1, 2, 'hello', 3, 'world', 4}
print(a[2])
```



```
Traceback (most recent call last):
  File "C:\pythonProject\datatype.py", line 2, in <module>
    print(a[2])
TypeError: 'set' object is not subscriptable
```

因为集合是无序的，索引没有意义。
因此，切片运算符 [] 不起作用。

计算机与网络空间安全学院
School of Computer and Cyber Sciences

6. 字典

字典（**Dictionary**）是键值对（**key-value** pairs）的无序集合。通常在有大量数据时使用。字典针对检索数据进行了优化。我们必须知道检索值的键（key）。

```
d = {101:'Tom','ID':'1101052012031700 54'}
print(type(d))
print("d[101] = ", d[101])
print("d['ID'] = ", d['ID'])
```

```
<class 'dict'>
d[101] = Tom
d['ID'] = 1101052012031700 54
```

- 在 Python 中，字典是在大括号 {} 中定义的
- 每个元素都是 key:value 形式的一对。
- key和value可以是任何类型。

2-1 变量与数据类型

数据类型之间的转换

我们可以使用 int()、float()、str()等不同的类型转换函数在不同的数据类型之间进行转换。

```
>>> float('2.5')
2.5
>>> str(25)
'25'
>>> int('1p')
Traceback (most recent call last):
    File "<string>", line 301, in runcode
    File "<interactive input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '1p'
```

```
>>> set([1,2,3])
{1, 2, 3}
>>> tuple({5,6,7})
(5, 6, 7)
>>> list('hello')
['h', 'e', 'l', 'l', 'o']
>>> dict([[1,2],[3,4]])
{1: 2, 3: 4}
>>> dict([(3,26),(4,44)])
{3: 26, 4: 44}
```

```
>>> float(5) 5.0
```

```
>>> int(10.6)
10
>>> int(-10.6)
-10
```

- 可以将一个序列转换为另一个序列
- 要转换为字典，每个元素必须是一对

数据类型之间的转换

函数	作用
int(x, base=10)	将数字或者 base 对应进制的字符串 x，转换成十进制的整数
float(x)	将数字或者字符串 x，转换成浮点数
hex(x)	将整数 x 转换为以'0x'为前缀的小写十六进制字符串
oct(x)	将整数 x 转换为以'0o'为前缀的八进制字符串
bin(x)	将整数 x 转换为以'0b'为前缀的二进制字符串
str(x)	将 x 转换为字符串
ord(x)	将单个字符 x 转换为整数
chr(x)	将整数 x 转换为对应的单个字符

2-2 语句与流程控制

语句与缩进

Python 解释器可以执行的指令称为语句（statement）。

例如，a = 1是一个赋值语句。if语句、for语句、while语句等是其他类型的语句

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

可以使用行继续符(\) 使语句扩展到多行

```
a = (1 + 2 + 3 +
    4 + 5 + 6 +
    7 + 8 + 9)
```

```
colors = ['red',
          'blue',
          'green']
```

在括号中的换行，不需要显式地标出续行符\

```
for(int i=1;i++;i<=10)
{
    printf(i);
    if(i == 5)
    {
        printf("five!");
        break;
    }
}
```

C语言

```
for i in range(1,10):
    print(i)
    if i == 5:
        print('five!')
        break
```

大多数编程语言（如 C、C++ 和 Java）都使用大括号{ }定义代码块（函数体、循环、判断等）。

然而，Python 使用缩进。

缩进使代码看起来更简洁干净。缩进量必须在整个块中保持一致。通常，四个空格用于缩进，也可以用制表符。

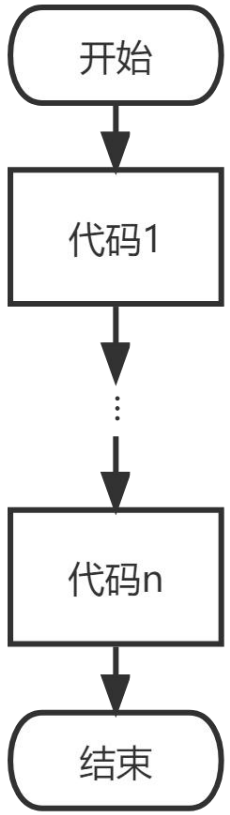
2-2 语句与流程控制

顺序执行语句

先执行第一行，
再执行第二行，
然后执行第三行，
……

```
a = 'hello,world'
b = [1,2,3]
c = 5
d = 5 * c - 2
print(a)
print(b)
print(d)
```

```
hello,world
[1, 2, 3]
23
```



顺序执行

2-2 语句与流程控制

布尔逻辑

- 布尔值（缩写为bool）是一种数据类型，具有两个可能值（通常表示 true 和 false）
- 在各类控制语句中，通过逻辑值检测来决定程序的执行流程。
- 布尔逻辑值检测或者叫真值检测 (Truth Value Testing)。

多个逻辑值可以通过布尔运算得到新的逻辑值。

运算	结果
x or y	如果 x 为 False，那么结果为 y，否则 x
x and y	如果 x 为 False，那么结果为 x，否则 y
not	如果 x 为 False，那么结果为 True，否则 False

其他数据类型则可以通过比较运算得到新的逻辑值。

运算	含义
<	严格小于
<=	小于或等于
>	严格大于
>=	大于或等于
==	等于
!=	不等于
is	对象标识
is not	否定对象标识

```
>>> a=10
>>> b=20
>>> a<b and b-a>=a
True
```

```
>>> a=10
>>> b=20
>>> a==b
False
```

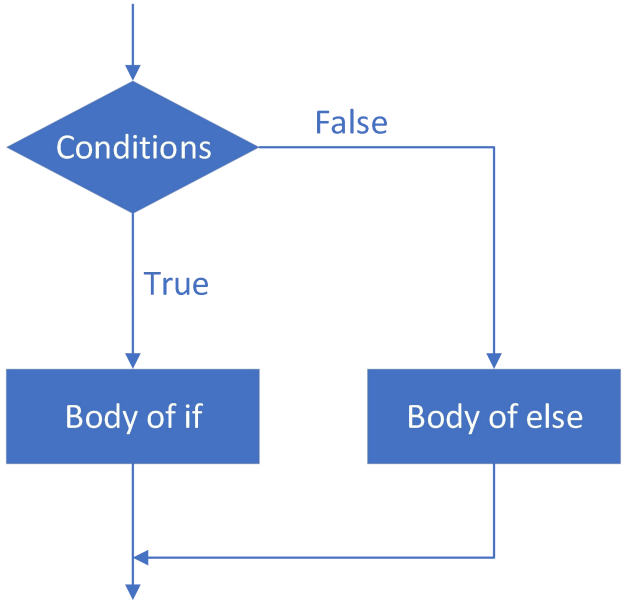
```
>>> a==b or a<b
True
```

2-2 语句与流程控制

分支语句

与其他语言一样，Python使用if/else/elif关键字根据判断条件实现程序的分支跳转。

```
if conditions:
    body of if
else:
    body of else
```




```
num = int(input())
if num >= 0:
    print(num, " is Positive or Zero")
else:
    print(num, " is Negative number")
```

仅当判断条件为 True 时才会执行if的代码块。如果条件为 False，则执行else的代码块。

此段代码输出是什么？
如果num改为-1呢？

```
num = 5
if num == 3:
    print('boss')
elif num == 2:
    print('user')
elif num == 1:
    print('worker')
elif num < 0:
    print('error')
else:
    print('roadman')
```

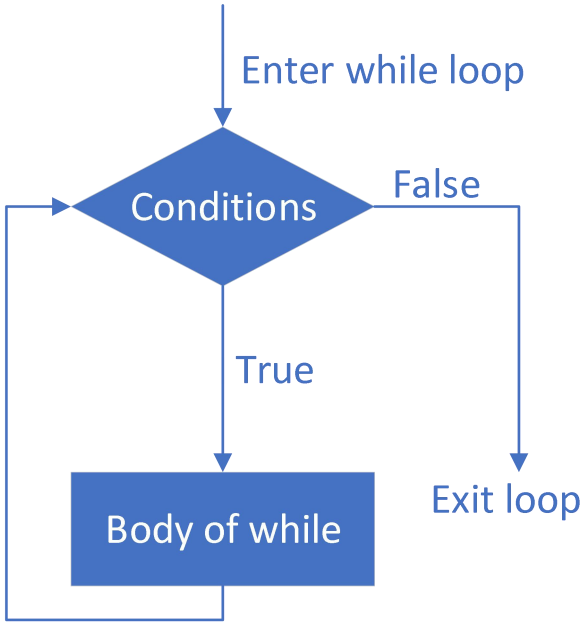
用elif实现多分支判断

2-2 语句与流程控制

循环语句

Python 中的 while 循环用于迭代代码块，只要判断条件为真，就会执行代码块。

```
while conditions:
    body of while
```




```
# sum = 1+2+3+...+n
# To take input from the user,
# n = int(input("Enter n: "))
n = 10
# initialize sum and counter
sum = 0
i = 1
while i <= n:
    sum = sum + i
    i = i+1    # update counter
# print the sum
print("The sum is", sum)
```

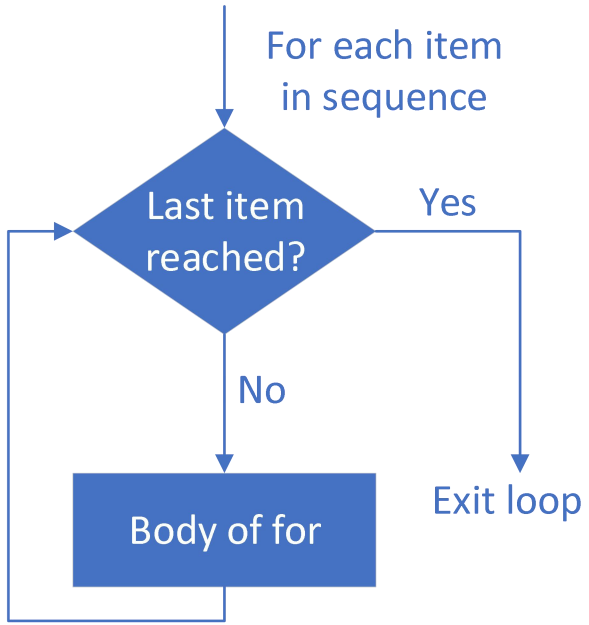
```
1  numbers = [12, 37, 5, 42, 8, 3]
2  even = []
3  odd = []
4  while len(numbers) > 0 :
5      number = numbers.pop()
6      if(number % 2 == 0):
7          even.append(number)
8      else:
9          odd.append(number)
```

www.penjee.com

计算机与网络空间安全学院
School of Computer and Cyber Sciences

循环语句

```
for item in sequence:
    loop body
```



Python 中的 for 循环可以遍历任何序列的元素，例如列表、元组、字符串等。for的用法非常灵活。

```
for char in "Python":
    print(char) # P,y,t,h,o,n

for i in [1,2,3]:
    print(i)    # 1,2,3

for i in (1,2,3):
    print(i)    # 1,2,3

for i in range(8):
    print(i)    # 0,1,2,3,4,5,6,7

for i in range(0,10):
    print(i)    # 0,1,2,3,4,5,6,7,8,9
```

我们可以使用 range() 函数生成一个数字序列。

2-2 语句与流程控制

循环语句

break / continue

Python 中的 break用于跳出循环，continue用于忽略当次循环剩下的语句直接进入下次循环。

```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

BREAK
VS
CONTINUE

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

2-2 语句与流程控制

循环语句

break / continue

Python 中的 break用于跳出循环，continue用于忽略当次循环剩下的语句直接进入下次循环。

```
# Use of break statement inside loops

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```

output

```
s
t
r
The end
```

BREAK VS CONTINUE

```
# Use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

output

```
s
t
r
n
g
The end
```

循环语句

pass

假设我们有一个尚未实现的循环或函数，但我们希望在将来实现它。在 Python 中，pass 语句是一个空语句，它什么也不做。

```
'''pass is just a placeholder for functionality to be added later.'''
sequence = {'p', 'a', 's', 's'}
for val in sequence:
    pass
```

```
def function(args):
    pass
```

```
class Example:
    pass
```

pass 经常用来占位，先留着一个空的位置，以后再实现具体功能。

本次课程结束，请继续加油！

Computational Thinking

计算机与网络空间安全学院
School of Computer and Cyber Sciences