

FSE598 前沿计算技术

模块 2 数据与数据处理 单元 2 数据结构与类型 第 1 讲 基本数据类型

讲座的英文版内容基于本书：

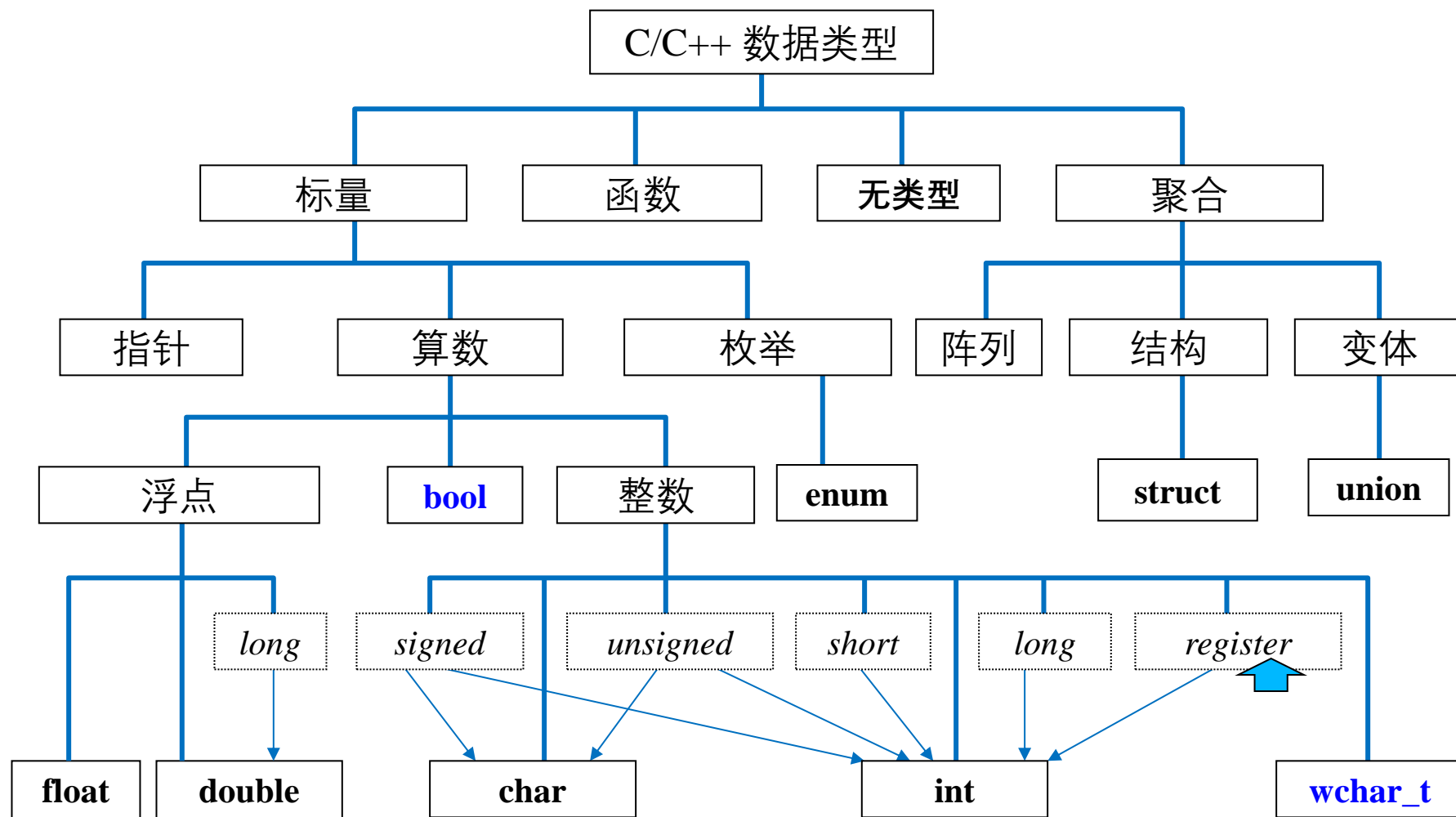
Y. Chen 《编程语言入门：C、C++、Scheme、Prolog、C# 和 Python 编程》(Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and Python), 第 6 版, Kendall Hunt Publishing Company, 2019 年。

<https://www.public.asu.edu/~ychen10/book/IntroPl.html>

学习

- ❑ 不同的数据类型
- ❑ Python 中的不同数据类型
- ❑ 字符串和字符类型
- ❑ 字符串操作

C/C++ 中的数据类型



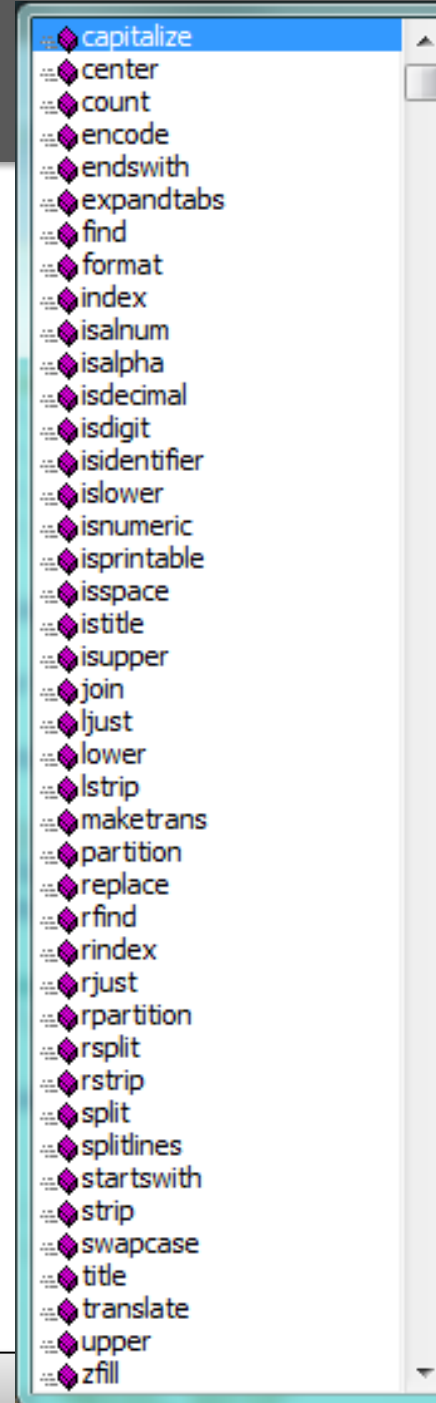
- ❑ C 和 C++ 类型都是静态的。
- ❑ 编译器在编译期间分配内存。
- ❑ 类型必须在使用前声明

在图中：
粗体：类型关键字
*斜体：*可选关键词。
其他名称是通用术语

- ❑ Python 数据类型是动态的。不需要预先说明。
- ❑ 将值分配给变量时分配内存
- ❑ Python 数据类型：
 - 文本类型：str (string)
 - 字符类型：没有字符类型。一个字符被认为是一个长度为 1 的字符串。
 - 数字类型：int, float, complex
 - 序列类型：list, tuple, range
 - 映射类型：dict (dictionary)
 - 集合类型：set, frozenset
 - 布尔类型：bool
 - 二进制类型：bytes, bytearray, memoryview
- ❑ 我们可以通过 type() 函数询问变量是什么类型
 - name = "bob"
 - print(type(name)) → <class 'str'>

字符串操作

- ❑ 字符串是表示文本的可变（可修改）字符集合
- ❑ 像 Java 和 C# 一样，它们属于对象
- ❑ 它们有很多方法(操作)
 - <http://www.python-ds.com/python-3-string-methods>



字符串运算符

- ❑ `[index]`
 - 获得一个特定的字符
- ❑ `[start : finish]`
 - 获得一个子字符串
- ❑ `[start :]`
 - 获取从start位开始到结尾的子字符串
- ❑ `+`
 - 是连接运算符
- ❑ `*`
 - 用于重复字符串
 - `print ("hello"*2)` 就会输出 “hello” 两遍

常用方法

- ❑ `count(value, start, end)`
 - 返回子字符串在字符串中出现的次数
- ❑ `find(value, start, end)`
 - 返回在字符串中找到的子字符串的索引
- ❑ `swapcase()`
 - 返回的字符串中字母大小写全部反转
- ❑ `split(sep, maxsplit)`
 - 返回由你指定的分隔符分隔的子字符串列表（分隔符默认是空格）
- ❑ `upper()/lower()/title()`
 - 改变字符串的大小写
- ❑ `strip(char)`
 - 从字符串中删除一组字符

- ❑ Python 字符串是编程语言中字符串操作最强大的对象之一
- ❑ 不仅是因为 Python 字符串拥有惊人的内置方法
- ❑ 还有一个用于格式化和模板制作的模块

<https://docs.python.org/3/library/string.html>

字符串和字符操作示例

`str1 = "Arizona State University"`

`str1[0] → A`, `str1[1] → r`, `str1[3:6] → zon`, 注意, 不包括`str1[6]` “a”

Python 字符串类型支持 `in` 和 `not in` 的成员操作。例如:

`"A" in str1 → true` and `"A" not in str1 → false`

`"B" in str1 → false` and `"B" not in str1 → true`

Python 字符串类型还支持连接操作 `+` 和重复操作 `*`。例如:

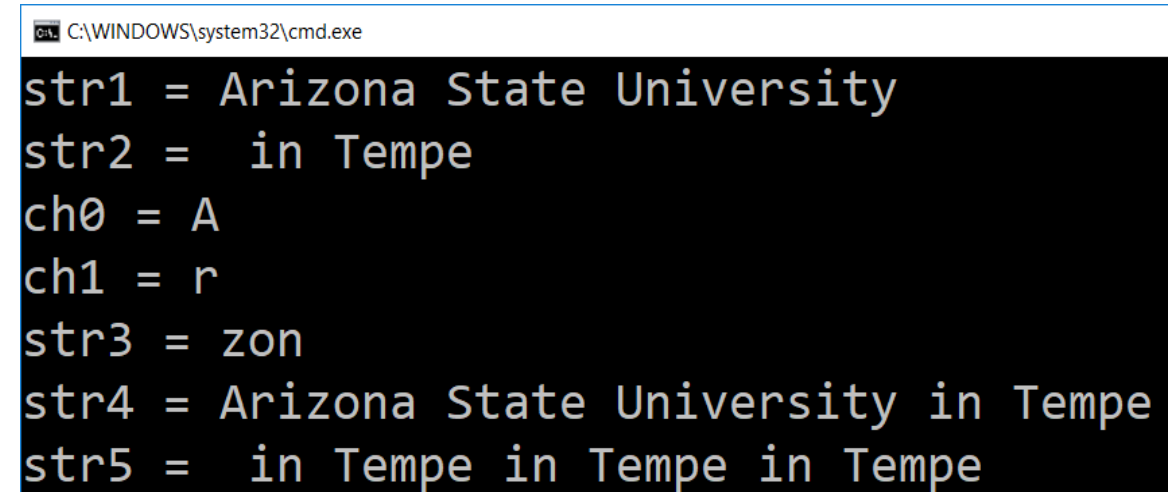
`str2 = " in Tempe"`

`str1 + str2 → Arizona State University in Tempe`

`str2*3 → in Tempe in Tempe in Tempe`

字符串操作示例

```
def main():
    str1 = "Arizona State University"
    str2 = " in Tempe"
    ch0 = str1[0]    # Length = 1 string is a char
    ch1 = str1[1]
    str3 = str1[3:6] # str1[6] will not be printed
    str4 = str1+str2
    str5 = str2*3
    print(F'str1 = {str1}')
    print(F'str2 = {str2}')
    print(F'ch0 = {ch0}')
    print(F'ch1 = {ch1}')
    print(F'str3 = {str3}')
    print(F'str4 = {str4}')
    print(F'str5 = {str5}')
if __name__ == "__main__": #Optional in Python 3
    main()
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\WINDOWS\system32\cmd.exe". The window contains the following output from the Python script:

```
str1 = Arizona State University
str2 =  in Tempe
ch0 = A
ch1 = r
str3 = zon
str4 = Arizona State University in Tempe
str5 =  in Tempe in Tempe in Tempe
```

更多字符串操作示例

```
def main():
    str1 = "Arizona State University"
    str2 = " in Tempe, Arizona"
    str3 = str2.replace("Tempe", "Phoenix Downtown")
    # substitute Phoenix Downtown for Tempe
    str4 = str3.upper() # Convert to upper cases
    str5 = "-".join(str2) # Add "-" before each character in str2
    str6 = "".join(reversed(str2)) # Reverse str2 and convert to string
    str7 = str1.split(' ') # split a string by " " into a list of words
    print(F"str1 = {str1}") #double quotes and single quotes are the same
    print(F'str2 = {str2}')
    print(F'str3 = {str3}')
    print(F'str4 = {str4}')
    print(F'str5 = {str5}')
    print(F'str6 = {str6}')
    print(F'str7 = {str7}')
    if __name__ == "__main__":
        main()
```

C:\WINDOWS\system32\cmd.exe

```
str1 = Arizona State University
str2 =  in Tempe, Arizona
str3 =  in Phoenix Downtown, Arizona
str4 =  IN PHOENIX DOWNTOWN, ARIZONA
str5 =  -i-n- -T-e-m-p-e-, -A-r-i-z-o-n-a
str6 =  anozirA ,epmeT ni
str7 =  ['Arizona', 'State', 'University']
```

在使用变量之前初始化变量

- ❑ 这不是必需的，但是在实际使用某个数据类型之前，创建一个拥有默认值的变量来代表这一数据类型是一个很好的做法。
- ❑ 在说明变量时，需要注意一些细微差别，以确保正确的数据类型在起作用

语句	类型
myVar = 100	myVar 是一个整数
myVar = 100.0	myVar 是一个浮点数
myVar = "100"	myVar 是一个字符串
myVar = 0	myVar 是一个整数
myVar = 0.0	myVar 是一个浮点数

- ❑ 为从用户获得一个简单的输入，你使用 `input()` 函数
- ❑ `input()` 是一个值返回函数，因此它将与赋值语句相结合
 - `name = input()`
- ❑ Python 中 `input()` 的一个很酷的功能是你可以将提示符直接构建到命令中
 - `name = input("Please enter your name: ")`
 - 这是一个直观的功能！

输入函数：将数据转换为字符串类型

- ❑ Python 3x 中的 `input()` 函数会将所有内容转换为字符串，即使所分配得变量已经拥有数据类型
 - 该代码：

```
myVar = 0.0
myVar = input("Enter a value: ")
print( type(myVar) )
```
 - 将输出 `<class 'str'>`
- ❑ 这里要注意，因为 `input()` 在 Python 2.x 中的运作方式不同
 - 在 2.x 中，输入将根据输入自动决定数据类型，而不是将所有内容都变成字符串
 - Python 2.x 有一条命令 `raw_input()`，与 Python 3.x `input()` 命令的功能相似（通常推荐使用）

类型转换

□ 使用 `input()` 时，我们可以将输入的结果转换为所需的数据类型

- 大多数数据类型都有用于转换的函数形式

- `int()` 用于整数
- `float()` 用于浮点数
- 等等

- 该代码:

```
myVar = 0
```

```
myVar = int(input("Enter a value: "))
```

```
print( type(myVar) )
```

- 将输出 `<class 'int'>`

□ 注意，在使用类型转换时，如果输入错误的类型会产生异常

- 输入 `12.3` 会产生异常（无自动类型转换，强类型）

```
def main(): #Perform operation 'ch' on 2 given numbers (n1 and n2)
```

```
    n1 = 10
```

```
    n2 = 15
```

```
    ch = "
```

```
    while (ch != 'q'):
```

```
        print('Menu: enter an operator: +, - , *, /')
```

```
        ch = input()
```

```
            #Print output using Python3 String Formatters
```

```
        print(F'ch = {ch}')
```

```
        if (ch == '+'): #Check operator
```

```
            print(F'Sum= {n1+n2}')
```

```
        elif (ch == '-'):

```

```
            print(F'Diff= {n1 - n2}')
```

```
        elif (ch == '/'):

```

```
            print(F'Division= {n1/n2}')
```

```
        elif (ch == '*'):

```

```
            print(F'Product= {n1*n2}')
```

```
        elif (ch=='q'):

```

```
            print('Terminated')
```

```
            break #exit the loop
```

```
    else:

```

```
        pass # a dummy place holder for doing nothing
```

```
            # continue
```

```
        print('invalid operator')
```

```
if __name__ == "__main__":
```

```
    main()
```

C:\Program Files (x86)\Microsoft Visual Studio\

```
Menu: enter an operator: +, - , *, /
```

```
*
```

```
ch = *
```

```
Product= 150
```

```
Menu: enter an operator: +, - , *, /
```

```
/
```

```
ch = /
```

```
Division= 0.6666666666666666
```

```
Menu: enter an operator: +, - , *, /
```

```
+
```

```
ch = +
```

```
Sum= 25
```

```
Menu: enter an operator: +, - , *, /
```

```
-
```

```
ch = -
```

```
Diff= -5
```

```
Menu: enter an operator: +, - , *, /
```

```
q
```

```
ch = q
```

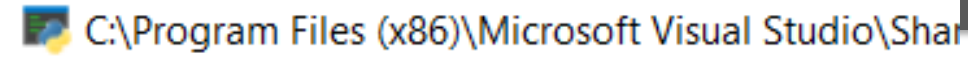
```
Terminated
```

```
Press any key to continue . . .
```



```
def main():
    #Use a for-loop, instead of a while loop
    n1 = 10
    n2 = 15
    ch = ''
    for i in range(10):
        print('Menu: enter an operator: +, - , *, /, q')
        ch = input()
        #Print output using Python 3 String Formatters
        print(F'ch = {ch}')
        if (ch == '+'):    #Check operator
            print(F'Sum= {n1+n2}')
        elif (ch == '-'):
            print(F'Diff= {n1 - n2}')
        elif (ch == '/'):
            print(F'Division= {n1/n2}')
        elif (ch == '*'):
            print(F'Product= {n1*n2}')
        elif (ch=='q'):
            print('Terminated')
            break    # exit the loop
        else:
            pass    # a dummy place holder for doing nothing
            print('invalid operator')

if __name__ == "__main__":
    main()
```



```
Menu: enter an operator: +, - , *, /, q
+
ch = +
Sum= 25
Menu: enter an operator: +, - , *, /, q
*
ch = *
Product= 150
Menu: enter an operator: +, - , *, /, q
-
ch = -
Diff= -5
Menu: enter an operator: +, - , *, /, q
/
ch = /
Division= 0.6666666666666666
Menu: enter an operator: +, - , *, /, q
q
ch = q
Terminated
Press any key to continue . . .
```