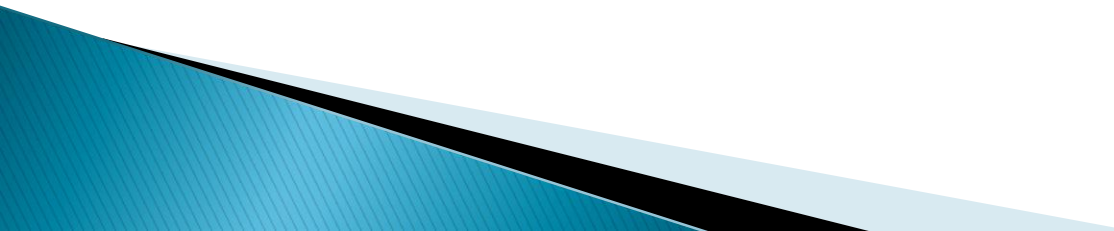


# Chap09–String

College of Computer Science  
Nankai University

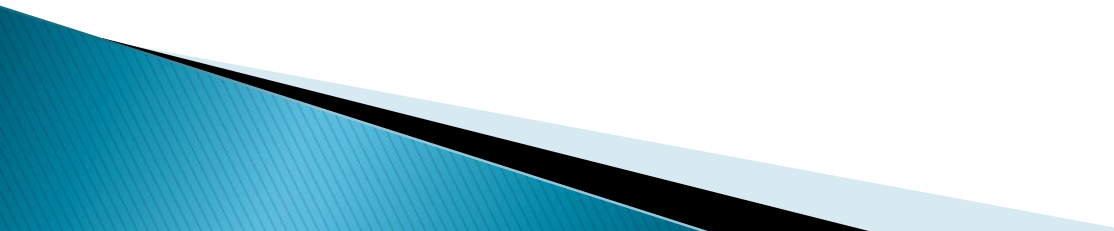
# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ String Methods, BIF
  - ▶ StringIO
  - ▶ Regular Expression
- 

# String-Introduction

- ▶ 在Python中，字符串也属于列表类型，字符串属于不可变序列类型，除了支持列表通用方法（包括切片操作）以外，还支持特有的字符串操作方法。
- ▶ Python字符串驻留机制：对于短字符串，将其赋值给多个不同的对象时，内存中只有一个副本，多个对象共享该副本。长字符串不遵守驻留机制。
- ▶ 判断一个变量s是否为字符串，应使用 `isinstance(s, str)`。
- ▶ 在Python3中，程序源文件默认为UTF-8编码，全面支持中文。

# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ String Methods, BIF
  - ▶ StringIO
  - ▶ Regular Expression
- 

# String-Encoding

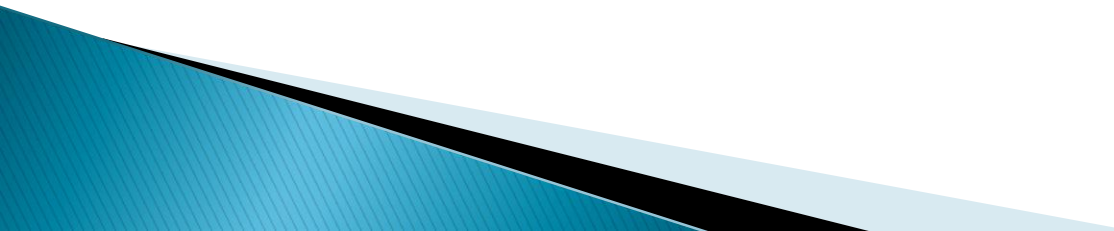
- ▶ 最早的字符串编码是美国标准信息交换码ASCII，仅对10个数字、26个大写字英文字母、26个小写字英文字母及一些其它符号进行了编码。
  - ASCII采用8位即1个字节，因此最多只能对256个字符进行编码。
- ▶ 随着信息技术的发展，各国的文字都需要进行编码，常见的编码有UTF-8，UTF-16，GB2312，GBK，CP936。
  - UTF-8编码是国际通用的编码，以8位，即1字节表示英语(兼容ASCII)，以24位即3字节表示中文及其它语言，UTF-8对全世界所有国家需要用到的字符进行了编码。

# String-Encoding

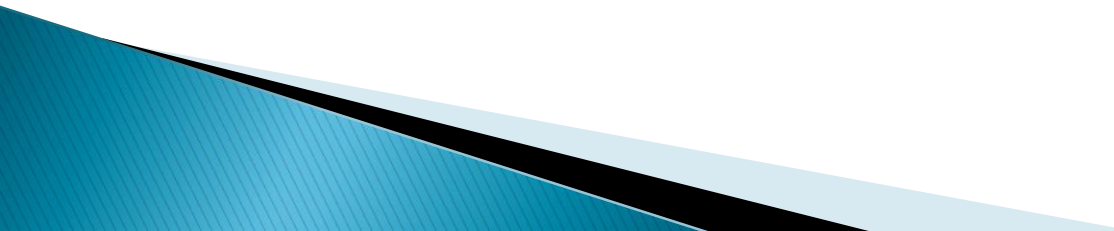
- ▶ GB2312是中国制定的中文编码，使用1个字节表示英语，2个字节表示中文；
- ▶ GBK是GB2312的扩充；
- ▶ CP936是微软在GBK基础上完成的编码；
- ▶ GB2312、GBK和CP936都是使用2个字节表示中文，UTF-8使用3个字节表示中文；
- ▶ Unicode是编码转换的基础。
- ▶ 在Windows平台上，input()函数从键盘输入的字符串默认为GBK编码，而Python程序的字符串编码使用#coding指定，如#coding=utf-8  
#coding:GBK

# String-Encoding

```
>>> import sys
>>> sys.getdefaultencoding()
'utf-8'
>>> sys.stdout.encoding
'cp936'
>>> s = '中国'
>>> s.encode('utf-8')
b'\xe4\xb8\xad\xe5\x9b\xbd'
>>> s.encode('gbk')
b'\xd6\xd0\xb9\xfa'
```

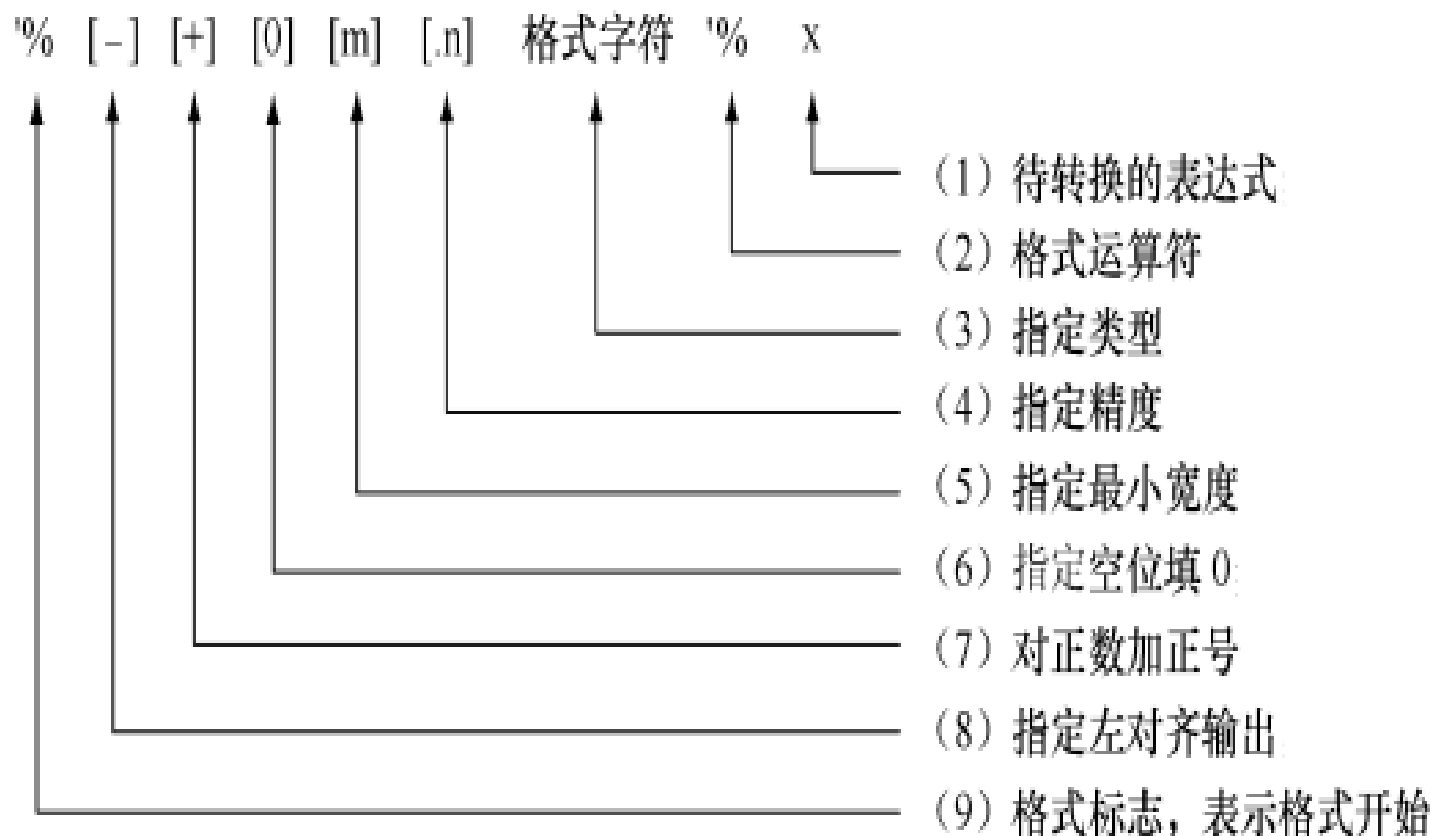


# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ **String Formatting**
  - ▶ String Methods, BIF
  - ▶ StringIO
  - ▶ Regular Expression
- 



# String-Formatting



# String-Formatting

## ▶ 格式字符

格式字符	说明
%s	字符串
%r	字符串
%c	单个字符
%b	二进制整数
%d	十进制整数
%i	十进制整数
%o	八进制整数
%x	十六进制整数
%e	指数（基底写为e）
%E	指数（基底写为E）
%f、%F、%F	浮点数
%g	指数(e)或浮点数（根据显示长度）
%G	指数(E)或浮点数（根据显示长度）
%%	字符"%"

# String-Formatting

- ▶ 使用字符串中的format方法也可以进行字符串的格式化操作，其语法格式为：`str.format(*args, **kwargs)`
  - `str`是用于格式化的字符串，可以包含由大括号`{}`括起来的替换字段。
  - 每个替换字段可以是位置参数的数字索引，也可以是关键字参数的名称。
  - `format`方法返回的是格式化的字符串副本（即通过`format`方法调用并不会改变`str`的值）。

# String-Formatting

#{}中的替换字段是位置参数的数字索引

```
str1='{0}的计算机成绩是{1}，{0}的数学成绩是{2}'
```

#替换字段是关键字参数的名称

```
str2='{name}的计算机成绩是{cs}，{name}的数学成绩是{ms}'
```

```
print(str1.format('李晓明',90,85))
```

```
print(str2.format(cs=90,ms=85,name='李晓明'))
```

# String-Formatting

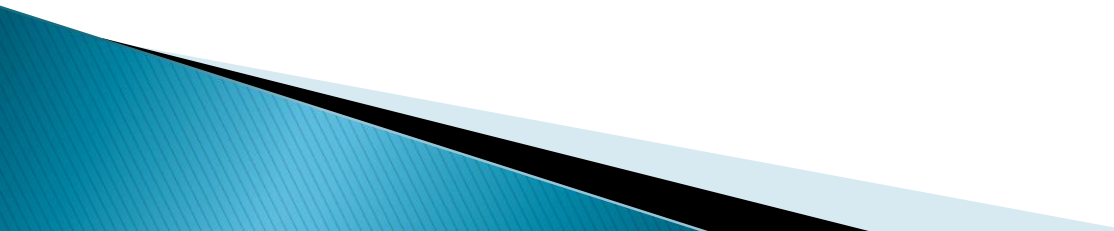
- ▶ 从python 3.6.x开始支持一种新的字符串格式化方式，在字符串前面加字母f，作用和format()方法类似。

```
>>> name = 'bob'
```

```
>>> gender = 'male'
```

```
>>> f'His name is {name}, and he is {gender}'  
'His name is bob, and he is male'
```

# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ **String Methods**, BIF
  - ▶ StringIO, Array
  - ▶ Regular Expression
- 

# String-Methods

- ▶ `find()`、`rfind()`、`index()`、`rindex()`、`count()`
- ▶ `find()`和`rfind()`方法分别用来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中首次和最后一次出现的位置，如果不存在则返回-1；
- ▶ `index()`和`rindex()`方法用来返回一个字符串在另一个字符串指定范围中首次和最后一次出现的位置，如果不存在则抛出异常；
- ▶ `count()`方法用来返回一个字符串在另一个字符串中出现的次数。

# String-Methods

```
>>> s="apple,peach,banana,peach,pear"
```

```
>>> s.find("peach")
```

```
6
```

```
>>> s.find("peach",7)
```

```
19
```

```
>>> s.find("peach",7,20)
```

```
-1
```

```
>>> s.rfind('p')
```

```
25
```

```
>>> s.index('p')
```

```
1
```

```
>>> s.index('pe')
```

```
6
```





# String-Methods

```
>>> s.index('pear')
```

```
25
```

```
>>> s.index('ppp')
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#11>", line 1, in <module>
```

```
    s.index('ppp')
```

```
ValueError: substring not found
```

```
>>> s.count('p')
```

```
5
```

```
>>> s.count('pp')
```

```
1
```

```
>>> s.count('ppp')
```

```
0
```

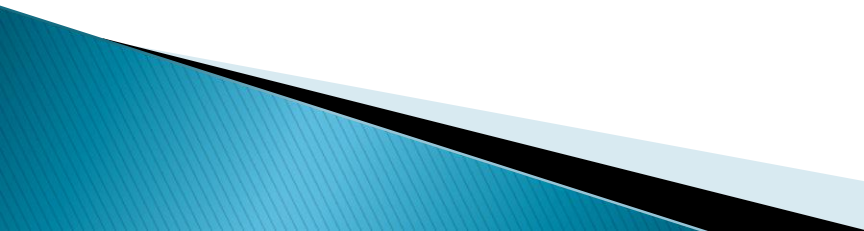


# String-Methods

- ▶ `split()`、`rsplit()`、`partition()`、`rpartition()`
- ▶ `split()`和`rsplit()`方法分别用来以指定字符为分隔符，将字符串左端和右端开始将其分割成多个字符串，并返回包含分割结果的列表；
- ▶ `partition()`和`rpartition()`用来以指定字符串为分隔符将原字符串分割为3部分，即分隔符前的字符串、分隔符字符串、分隔符后的字符串，如果指定的分隔符不在原字符串中，则返回原字符串和两个空字符串。

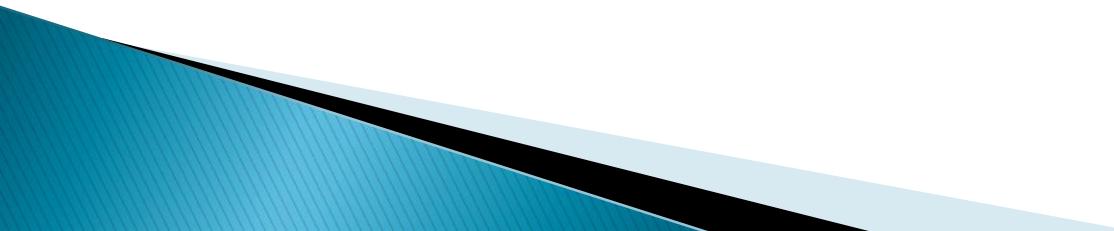
# String-Methods

```
>>> s="apple,peach,banana,pear"
>>> li=s.split(",")
>>> li
["apple", "peach", "banana", "pear"]
>>> s.partition(',')
('apple', ',', 'peach,banana,pear')
>>> s.rpartition(',')
('apple,peach,banana', ',', 'pear')
>>> s.rpartition('banana')
('apple,peach,', 'banana', ',pear')
```



# String-Methods

```
>>> s = "2014-10-31"  
>>> t=s.split("-")  
>>> print(t)  
['2014', '10', '31']  
>>> print(list(map(int, t)))  
[2014, 10, 31]
```



# String-Methods

## ▶ 字符串联接join( )

```
>>> li=["apple", "peach", "banana", "pear"]
```

```
>>> sep=","
```

```
>>> s=sep.join(li)
```

```
>>> s
```

```
"apple,peach,banana,pear"
```

不推荐使用+连接字符串

# String-Methods

- ▶ lower()、upper()、capitalize()、title()、swapcase()
- ▶ 方法分别用来将字符串转换为小写、大写字符串、将字符串首字母变为大写、将每个单词的首字母变为大写以及大小写互换。

# String-Methods

```
>>> s="What is Your Name?"
```

```
>>> s2=s.lower()
```

```
>>> s2
```

```
"what is your name?"
```

```
>>> s.upper()
```

```
"WHAT IS YOUR NAME?"
```

```
>>> s2.capitalize()
```

```
"What is your name?"
```

```
>>> s.title()
```

```
'What Is Your Name?'
```

```
>>> s.swapcase()
```

```
'wHAT IS yOUR nAME?'
```

# String-Methods

## ▶ 查找替换replace( )

```
>>> s="中国， 中国"
```

```
>>> s.replace("中国", "中华人民共和国")
```

```
中华人民共和国， 中华人民共和国
```

```
>>> s.replace("中国","中华人民共和国",1)
```

```
'中华人民共和国， 中国'
```



# String-Methods

- ▶ strip()、rstrip()、lstrip() 方法分别用来删除两端、右端或左端的空格或连续的指定字符

```
>>> s=" abc "
```

```
>>> s.strip( )
```

```
"abc"
```

```
>>> "aaaassddf".strip("a")
```

```
"ssddf"
```

```
>>> "aaaassddf".strip("af")
```

```
"ssdd"
```

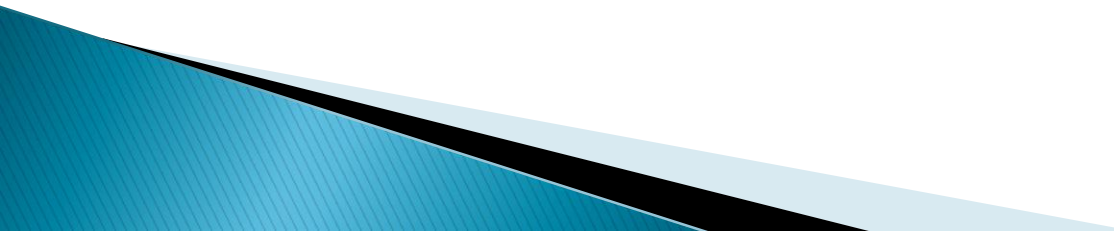
```
>>> "aaaassddfaaa".rstrip("a")
```

```
'aaaassddf'
```

```
>>> "aaaassddfaaa".lstrip("a")
```

```
'ssddfaaa'
```

# String-Methods

- ▶ `isalnum()`
  - ▶ `isalpha ()`
  - ▶ `isdigit ()`
  - ▶ `isdecimal ()`
  - ▶ `isnumeric ()`
  - ▶ `isspace ()`
  - ▶ `isupper ()`
  - ▶ `islower ()`
- 

# String-Methods

- ▶ startswith()、endswith()判断字符串是否以指定字符串开始或结束

```
>>> 'abcdedf'.startswith('abc')
```

```
True
```

```
>>> 'abcdedf'.startswith('edf')
```

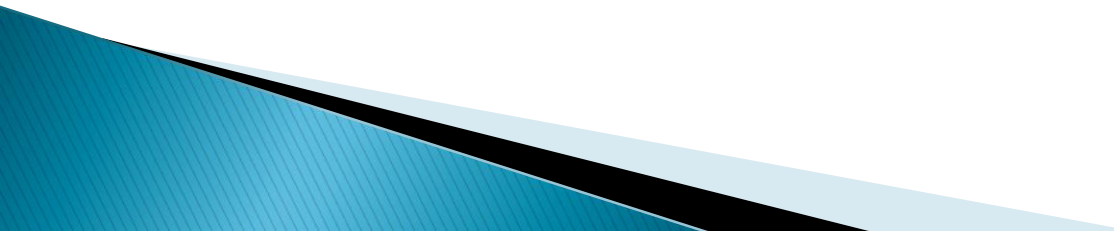
```
False
```

```
>>> 'abcdedf'.endswith('edf')
```

```
True
```



# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ String Methods, **BIF**
  - ▶ StringIO, Array
  - ▶ Regular Expression
- 

# BIF

- ▶ `len()`
- ▶ `max()`
- ▶ `min()`
- ▶ `zip()`
- ▶ `eval()`: 字符串表达式求值函数
- ▶ 切片也适合字符串，但是仅限于读取其中的部分，不支持字符串的修改。
- ▶ 成员判断符号: `in`  

```
>>> "a" in "abcde"
```

```
True
```

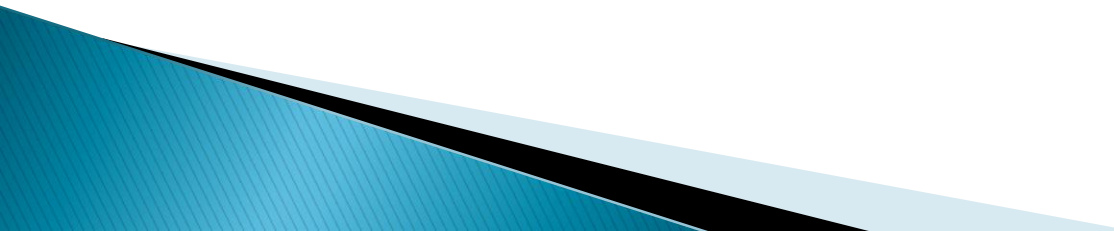
# String Processing Package

- ▶ Python的zlib库提供compress()和decompress()函数用于字符串的压缩和解压缩

# String Constant

```
>>> import string
>>> string.digits
'0123456789'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> string.letters
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
OPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~
\t\n\r\x0b\x0c'
>>> string.lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ String Methods, BIF
  - ▶ **StringIO**, Array
  - ▶ Regular Expression
- 

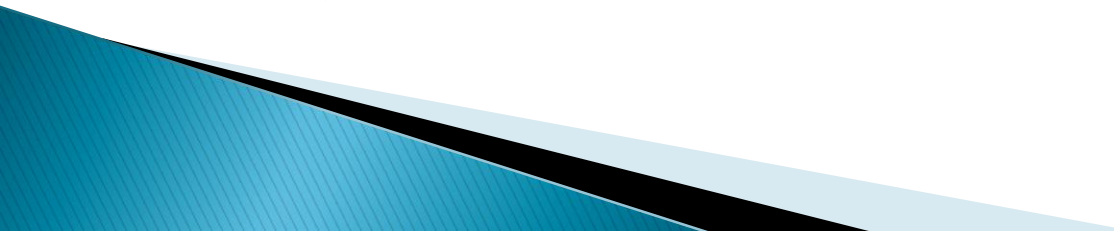


# StringIO and Array

- ▶ 在Python中，字符串属于不可变对象，不支持原地修改，如果需要修改其中的值，只能重新创建一个新的字符串对象。
- ▶ 如果确实需要一个支持原地修改的unicode数据对象，可以使用io.StringIO对象或array模块。

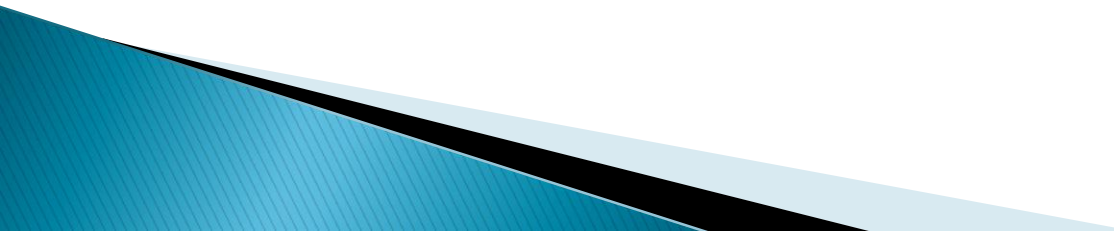
# StringIO

```
>>> import io
>>> s = "Hello, world"
>>> sio = io.StringIO(s)
>>> sio.getvalue()
'Hello, world'
>>> sio.seek(7)
7
>>> sio.write("there!")
6
>>> sio.getvalue()
'Hello, there!'
```

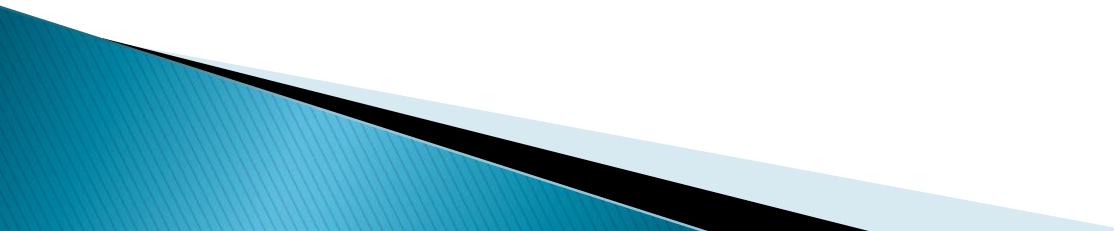


# Array

```
>>> import array
>>> a = array.array('u', s)
>>> print(a)
array('u', 'Hello, world')
>>> a[0] = 'y'
>>> print(a)
array('u', 'yello, world')
>>> a.tounicode()
'yello, world'
```



# Outline

- ▶ String Introduction
  - ▶ String Encoding
  - ▶ String Formatting
  - ▶ String Methods, BIF
  - ▶ StringIO, Array
  - ▶ Regular Expression
- 

# Regular Expression

