# Python Programming

## Lecture 5 Strings

# 5.1 Strings

- A string is a sequence of characters. The elements of a string are characters. Empty string ''. (not' ') You can access the characters one at a time with the bracket operator.

```
>>> fruit = 'banana'
>>> fruit[1]
'a'
>>> len(fruit)
6
```

```
>>> fruit = 'banana'
>>> fruit[1:3]
'an'
>>> fruit[3:]
'ana'
```

- in operator

```
>>> print('a' in 'banana')
True
>>> print('seed' in 'banana')
False
>>> print('ana' not in 'banana')
False
```

- Iteration

```
fruit = 'banana'
for char in fruit:
    print(char)
```

- Strings are immutable (similar to Tuples)

```
>>> greeting = 'Hello, world!'
>>> greeting[0] = 'J'

TypeError: 'str' object does not
support item assignment
```

```
>>> greeting = 'Hello, world!'
>>> new_greeting = 'J' + greeting[1:]
>>> print(new_greeting)

Jello, world!
```

- Comparison operations are useful for putting words in alphabetical order.

```
>>> print('apple'>'banana')
False
>>> print('ba' > 'banana')
False
>>> a_list = ["orange", "apple", "banana"]
>>> sorted(a_list)
['apple', 'banana', 'orange']
```

# Methods of Strings

## Converting Characters: .title(), .lower(), .upper()

- String's methods do not change the original variable but return values.

```
>>> name = "ada lovelace"
>>> print(name.title())
Ada Lovelace

>>> print(name)
ada lovelace
```

```
>>> name = "Ada Lovelace"
>>> print(name.upper())
ADA LOVELACE

>>> print(name.lower())
ada lovelace
```

## Stripping Whitespace: .lstrip(), .rstrip(), .strip()

- To programmers 'python' and 'python ' look pretty much the same. But to a program, they are two different strings. To ensure that no whitespace exists at the right end of a string, use the rstrip() method.
- You can also strip whitespace from the left side of a string using the lstrip() method or strip whitespace from both sides at once using strip().

```
>>> favorite_language = ' python '
>>> favorite_language.rstrip()
' python'
>>> favorite_language.lstrip()
'python '
>>> favorite_language.strip()
'python'
```

- However, it is only removed temporarily. To remove the whitespace from the string permanently, you have to store the stripped value back into the variable.

```
>>> favorite_language = 'python '
>>> favorite_language = favorite_language.rstrip()
>>> favorite_language
'python'
```

## Parsing Strings: .find(), split(), .isalpha(), isspace()

- .find() searches for the position of a string in another string

```
>>> word = 'banana'
>>> index = word.find('a')
>>> print(index)
1
>>> word.find('na')
2
>>> word.find('na', 3)
4
```

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ',atpos)
>>> print(sppos)
31
>>> host = data[atpos+1 : sppos]
>>> print(host)
uct.ac.za
```

- .split() breaks a sentence into words and make a list

```
>>> s = 'break a sentence into words'
>>> t = s.split()
>>> print(t)
['break', 'a', 'sentence', 'into', 'words']
```

- .isalpha() returns True if all characters in the string are alphabetic (A–Z, a–z), otherwise False.

  .isspace() returns True if the string only contains whitespace characters (spaces, tabs, newlines), otherwise False.

```
>>> print("Hello".isalpha())   # True（全是字母）
>>> print("Hello123".isalpha()) # False（包含数字）
>>> print("你好".isalpha())     # True（支持中文等非拉丁字符）

>>> print("   ".isspace())  # True（全是空格）
>>> print("\t\n".isspace()) # True（制表符和换行符也算空白）
>>> print("Hello".isspace()) # False（包含字母）
```

# Formatted String Literals

```
>>> number = 42
>>> print('I have spotted number camels.') #error.
>>> print('I have spotted '+str(number)+' camels.') #not simple
```

```
>>> number = 42
>>> print(f'I have spotted {number} camels.')
I have spotted 42 camels.
```

```
>>> animal = 'camels'
>>> number = 42.12345678
>>> print(f'''I spotted {number:.2f} {animal}.''')
I spotted 42.12 camels.
>>> print(f'''I spotted {number:.0f} {animal}.''')
I spotted 42 camels.

>>> print(f'''I spotted {number:.2} {animal}.''')
I spotted 4.2e+01 camels.
>>> print(f'''I spotted {number:.5} {animal}.''')
I spotted 42.123 camels.

>>> print(f'''I spotted {number:.2%} {animal}.''')
I spotted 4212.35% camels.
```

# String: Summary

- The elements of a string are characters. Empty string ''
- Features: Ordered, Repeatable, Immutable
- Index and slice are the same with that of tuples.
- in operator shows the boolean value for whether a string contains a given string.
- You can compare two strings in Alphabetical order.
- String Methods
  - .upper(), lower(), .title()
  - rstrip(), .lstrip(), .strip()
  - .find(), .split(), .isalpha(), isspace()
- Formatted String Literals

# 5.2 Sentiment Analysis (情感分析)

# 情感分析(简化版)

情感分析（Sentiment Analysis），也称为情绪分析或意见挖掘，广泛用于各种行业和场景，帮助企业和个人理解文本中的情感倾向（正面、负面或中性）。以下是几个重要的应用场景：

- 社交媒体分析：检测品牌口碑，舆情监测
- 客户反馈分析：产品评论分析，客服聊天分析
- 电影音乐书籍评鉴分析：推荐算法改进，自动总结评论
- 竞争对手分析：市场调研，行业趋势分析
- 股票市场预测：金融新闻，社交媒体

- 1. Text Cleaning: Remove punctuation, convert to lowercase, split words, remove stopwords
- 2. Sentiment Analysis: Count positive/negative words to determine sentiment

```
text = "This movie was absolutely amazing! The story was engaging\
 and the characters were great. However, some scenes felt unnecessary\
 and a bit boring. Overall, I loved it!"
```

```
text = text.lower()
# 去除标点（只保留字母和空格）
cleaned_text = ""
for char in text:
    if char.isalpha() or char.isspace():
        cleaned_text += char

words = cleaned_text.split()

# 去除简单的停用词（自己定义）
stop_words = ["the", "was", "and", "a", "it", "i", "some"]
filtered_words = []  # 创建一个空列表来存放保留的单词
for word in words:
    if word not in stop_words:  # 仅当单词不在停用词中时才添加
        filtered_words.append(word)

print("关键词：", filtered_words)
```

```python
# 定义正面 & 负面词
positive_words = ["amazing","great","engaging","loved"]
negative_words = ["boring","unnecessary","bad","terrible"]

# 计算正面词的数量
pos_count = 0
for word in filtered_words:
    if word in positive_words:
        pos_count += 1

# 计算负面词的数量
neg_count = 0
for word in filtered_words:
    if word in negative_words:
        neg_count += 1
```

```python
# 判断情感倾向
if pos_count > neg_count:
    sentiment = "正面"
elif neg_count > pos_count:
    sentiment = "负面"
else:
    sentiment = "中性"

print("情感分析结果：", sentiment)
print(f"（正面词：{pos_count} 个，负面词：{neg_count} 个）")
```

- 早在2010年，就有学者指出，可以依靠Twitter公开信息的情感分析来预测股市的涨落，准确率高达87.6%！
- 这个简化版只用到了几个单词的集合，而专业情感分析库（如，TextBlob、VADER）通常包含上千个情感词，还能识别不同的语境。
- 在 VADER 词典中，每个单词都有情感强度评分（如 “amazing” 可能是 4.0，“good” 可能是 1.9）。本例中没有这种权重计算。
- 真实情感分析库可以处理否定词、语法结构等，如 "not bad" 可能被解析为正面，而不是简单地统计 “not” 和 “bad” 的出现次数。

# 5.3 Chinese Idiom Relay (成语接龙)

- 1. 加载成语词典。
- 2. 给定一个成语，找到可以接上的所有成语。
- 3. 从一个给定的成语开始，一直接下去，到不能接下去为止。

```
# 1. 加载成语词典
filename = 'idiom_dictionary.txt'
with open(filename, encoding="utf-8") as file_object:
    lines = file_object.readlines() #List
```

```
d_game={}
for line in lines:
    if line!="\n":
        endpoint=line.find("拼音")
        idiom = line[:endpoint].strip()
        pinyin_start = line.find(": ", endpoint)
        pinyin_end =line.find("释义")
        each= line[pinyin_start+1: pinyin_end]
        pinyin_list = each.split()
        d_game[idiom] = pinyin_list
print(len(d_game))
```

```python
# 2. 给定一个成语，找到可以接上的所有成语
idiom = input("请输入第一个成语\n")
char_4th = d_game[idiom][-1]
for x, y in d_game.items():
    if char_4th == y[0]:
        print(x)
```

```python
# 3. 从一个给定的成语开始，一直接下去，到不能接下去为止。
idiom = input("请输入第一个成语\n")
enter=""
while enter!="q":
    char_4th = d_game[idiom][-1]
    for x, y in d_game.items():
        if char_4th == y[0]:
            idiom = x
            print(idiom)
            break
    enter=input("continue?")
```

- 有可能找不到能够接下去的成语了，但是程序不会结束，怎么解决呢?

```
idiom = input("请输入第一个成语\n")
enter=""
exist = True
while enter!="q" and exist:
    char_4th = d_game[idiom][-1]
    for x, y in d_game.items():
        if char_4th == y[0]:
            idiom = x
            print(idiom)
            exist = True
            break
        else:
            exist = False
    if exist:
        enter=input("continue?")
    else:
        print("对不起，没有成语了")
```

- 可进一步添加如下功能：
- 基本释义，谐音取词，人机对战，随机取词

```
# 基本释义功能
# 修改第一步
d_ex={}
for line in lines:
    if line!="\n":
        endpoint=line.find("拼音")
        idiom=line[:endpoint].strip()
        pinyin_end=line.find("释义")
        pinyin_start=line.find(": ", endpoint)
        explanation=line[pinyin_end:]
        d_ex[idiom]=explanation
words = input("请输入要查询的成语\n")
print(d_ex[words])
```

```
# 谐音取词
# 修改第一步
import unicodedata
d_game={}
for line in lines:
    if line!="\n":
        endpoint=line.find("拼音")
        idiom=line[:endpoint].strip()
        pinyin_start=line.find(": ", endpoint)
        pinyin_end=line.find("释义")
        each=line[pinyin_start+1: pinyin_end]
        each=unicodedata.normalize('NFKD',each).encode('ascii','ignore').decode()
        pinyin_list=each.split()
        d_game[idiom]=pinyin_list
```

# Summary

- Strings
- Reading: Python for Everybody
  - Strings Chapter 6