Structured Programs, String Processing and File I/O

YE Yuxiao

Tsinghua University

yeyuxiao@outlook.com

2017/10/13

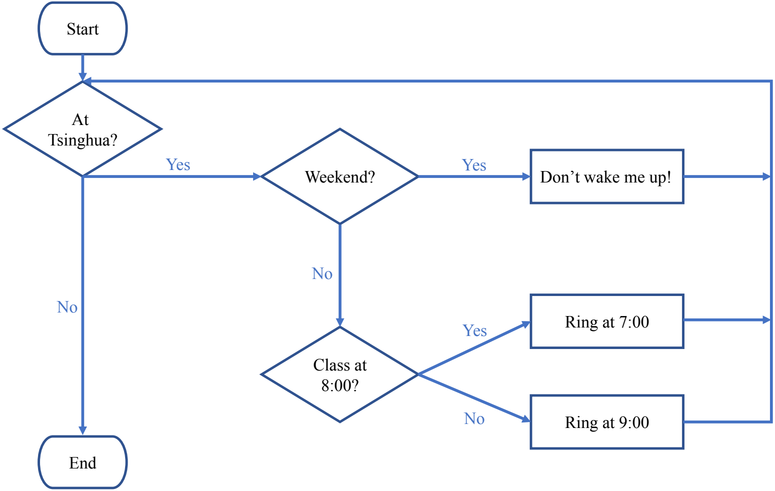
Overview

1. Structured Programs
    - Loops
    - Conditions
2. Basic String Operations
    - Indexing and Slicing
    - Stripping, Splitting and Joining
    - Upper/Lower Case
3. String Operations with NLTK
    - Tokenization
    - Stemming
4. File I/O
    - Read from Files

Why Structured?

- A single command(statement) usually cannot accomplish complex tasks
- Multiple statements are logically organized
- Sometimes such logic (algorithm) is shared by both humans and computers
- Task: set up an alarm to wake you up everyday

Algorithm of Setting up the Alarm



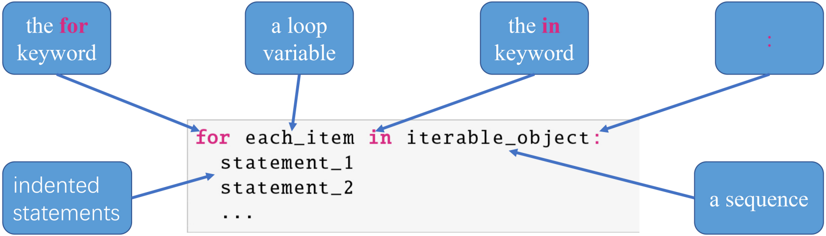
- Two fundamental constructs: loops and conditions

Loops

- The for loop: iterating over a sequence of items to perform an action repeatedly

```
for i in [1, 2, 3, 4, 5]:
  i += 1
  print (i)
```

- The for loop in detail:



```
for each_item in iterable_object:
  statement_1
  statement_2
  ...
```

Conditions

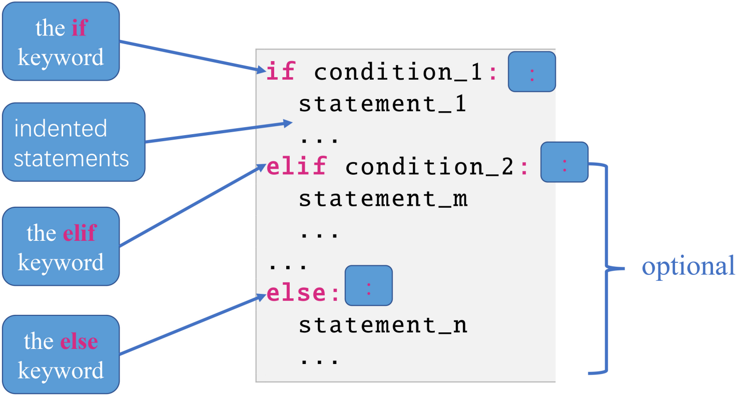
- The if statement: performing an action only when a certain condition is met

```
a = [1, 2, 3, 4, 5]

if a[0] < 1:
  print ('the first item in a is less than 1')
elif a[0] == 1:
  print ('the first item in a is equal to 1')
else:
  print ('the first item in a is more than 1')
```

Conditions

- The if statement in detail:



```
if condition_1:
    statement_1
    ...
elif condition_2:
    statement_m
    ...
...
else:
    statement_n
    ...
```

Conditions

- A condition in the if statement is presented as an expression which evaluates to either True or False
  - e.g. a == 1, 1 < 0
- These expressions evaluate to True:
  - True
  - 1 != 0
  - 1
  - "Hello world"
  - [0, 1]
- These expressions evaluate to False:
  - False
  - 1 == 0
  - 0
  - ""
  - [ ]
  - None

Loops and Conditions Combined

- Loops and conditions are often combined to construct more complex structures

```
students = ['Carlos', 'Alison', 'Catherine', 'Zebulon', 'Carol',
                                  'Simon', 'Camille']

for name in students:
  if name[0] == 'C':
    print (name)
```

Indexing and Slicing

- Covered in last week's class
  - `"Tsinghua"[0]`
  - `"Tsinghua"[5:]`

Stripping

- Strip: removing certain characters on both sides of a string
- Usage: str.strip([characters you want to remove])
  - `"abbbaaabbba".strip('a')`
  - `"abbbaaabbba".strip('ab')`
  - `" abbbaaabbba ".strip()` (default to blank characters: ” ”, ”\t”, ”\n”, ”\r”, ”\f”)

Splitting

- Split: slicing a string with certain separators
- Usage: str.split([separators])
  - `"info.tsinghua.edu.cn".split('.')`
  - `"I don't like Python.".split()` (default to blank characters)

Upper/Lower Case

- Upper/Lower Case: changing the case of all characters in a string
- Usage: str.upper()/lower()

  -
    ```
    "Tsinghua University".upper()
    ```

  -
    ```
    "Tsinghua University".lower()
    ```

String Operations with NLTK

- NLTK has many handy tools to process strings
- You can either use existing tools, or customize your own tools

Tokenization

- Decomposing a string into tokens
  - e.g. ”That U.S.A. poster-print costs $12.40” --> [’That’, ’U.S.A.’, ’poster-print’, ’costs’, ’$12.40’]
- An existing tokenizer: word_tokenize
  - `from nltk.tokenize import word_tokenize`
  - `sentence ="That U.S.A. poster-print costs $12.40"`
  - `words = word_tokenize(sentence)`
  - `print (words)`

Stemming

- Reducing inflected (or sometimes derived) words to their word stem, base or root form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.
    - e.g. ”stemming” --> ”stem”, ”arguing” --> ”argu”
    - stemming vs. lemmatization

- An existing stemmer: PorterStemmer
    - `from nltk.stem.porter import PorterStemmer`
    - `ps = PorterStemmer()`
    - `ps.stem("stemming")`
    - `ps.stem("took")`

File I/O

- Read from files: Python reads text in a file line by line

```
with open (fullpath+filename, 'r') as f:
  for line in f:
    print (line)
```

- Write to files

```
with open (fullpath+filename, 'w/a') as f:
  for i in ['a', 'b', 'c']:
    f.write(i + '\n')
```

Practice

Assume we have a sentence,

```
S = "Carlos and I met in 2014, in a bar called Helen's."
```

.

Write some code to:

1. count the number of words in S
2. print every word which:
    1. either begins with a capital letter
    2. or contains less than 3 letters
    3. or is a (numeric) number

Tip: `len('Hello')= 5`

Solution

```
S = "Carlos and I met in 2014, in a bar called Helen's."

print (len(S.split()))

import re
for word in S.split():
  if re.match(r'[A-Z\d]', word[0]):
    print (word)
  elif len(word) < 3:
    print (word)
```

Q & A

- Use Piazza to ask questions and discuss with instructors/classmates!
- Weekly homework
- Practice RE online:
  - https://regex101.com/#python
  - http://www.regexr.com