Regular expressions

- Complex searching and substitutions
- Regular expression is not specially quoted in Python
  - Be careful on \
  - Use raw string r”\.html$”
- Anchors ^,$
- Quantifiers *+? {}
- Character sets [] [^], interval a-z
- \d \w \z
- Grouping () \1..\99

import re

- Compilation re.compile(re,[modifiers])
- Methods of object representing RE
  - match
  - search
  - findall
  - finditer
- Or you can use match(re,string), search(re,string)…

Match object

- Methods
  - start()
  - end()
  - group()
  - span()
- Named group (?P<name>…)

- m=re.compile("a+")
- s="accaabaaavvv”
- print m.findall(s) #['a', 'aa', 'aaa']

Substitution with RE

- Methods of object representing RE
  - split(string[, maxsplit=0])
  - sub(replacement, string[, count=0])
  - subn(replacement, string[, count=0])

- m=re.compile("a+")
- s="accaabaaavvv”
- print m.sub('A',s) #AccAbAvvv

Iterable/Iterator

- Iterable is everything what can be used to iterate over:
  - for var in *iterable*:
  - for i in ‘cau’: print i
- Iterator is object which remembers state where is during and between iteration calls
- s=“Bye”
- i=iter(s)
- next(i) #‘B’
- next(i) #‘y’
- next(i) #’e’
- next(i) #exception StopIteration

Iterator

```
class firstn(object):
    def __init__(self, n):
        self.n = n
        self.num = 0

    def __iter__(self):
        return self

    # Python 3 compatibility
    def __next__(self):
        return self.next()

    def next(self):
        if self.num < self.n:
            cur, self.num = self.num, self.num+1
            return cur
        else:
            raise StopIteration()

sum_of_first_n = sum(firstn(100))
```

Generator

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1

sum_of_first_n = sum(firstn(100))
```