Python 3 SL Review LIJIA XU (2/2)

Regular expressions ⇒ A domain specific language (DSL) that is present as a library in most modern programming languages, not just Python

```
import re
pattern = r"spam"
if re.match(pattern, "spamsp":
    print ("Match")
else:
    print ("No match")
```

re.match ⇒ begining of a string
re.search ⇒ find a match anywhere
re.findall ⇒ return a list of all strings

re.sub(pattern, repl, string, max=0) replace pattern with repl

character class: inside [ae1ou] certain any one of it

. ⇒ matches any character
^ ⇒ start of a string
$ ⇒ end of a string

[^A-Z] ^ ⇒ invert the class
$ and . ⇒ no means in character class

specify number of repetitions

* zero or more rep of previous
+ one or more rep of previous
? zero or one rep of previous
{x,y} x to y rep of previous

special groups: named groups (?P<name>...) non-capturing groups (?:...)

group(0) or group() ⇒ whole match
group(n) # n > 0 nth group from left
groups() ⇒ all groups from 1

| ⇒ or
pattern = r"gr|a|e|y"

Special sequences:

re.finditer ⇒ same as re.findall but return a iterator

\1 ⇒ matches the expression of the group of that number
\d ⇒ digits
10, 15, \W ⇒ means opposite
\s ⇒ white spale
\w ⇒ word characters
\A, \Z, \b, \B
pattern = r"\bcat\b"

The zen of Python: import this PEP8:

```
def function(named, *args):
    print(named)
    print(args)
function(1,2,3,4,5)
```

lines <= 80 characters
avoid "from module import"
there should be only one statement per line

```
>>>
1
(2, 3, 4, 5)
>>>
```

default value must come after named parameters without default

```
def function(x, y, food="sp")
    print(food)
function(1, 2) ⇒ "sp"
function(1, 2, "good") ⇒ "good"
```

**kwargs keyword arguments ⇒ return a dictionary ⇒

```
def my_func(x,y=7,*args,**kwargs)
    print(kwargs) #{'a':7,'b':8}
my_func(2,3,4,5,6,a=7,b=8)
```

```
x, y = (1, 2) ⇒ list will also work
x, y = y, x
print(y) # 1
```

```
a, b, *c, d = [1, 2, 3, 4, 5, 6]
print(a) #1
print(c) #2, 3, 4, 5
print(d) #6
```

else can be used for: for or while loops

Ternary operator

```
a=7
b=1 if a>=5 else 42
print(b)
```

module or .py

```
def function()
    print("this is module")
if __name__=="__main__":
    print("a script")
```

```
for i in range(10):
    if i==999:
        break
else:
    print("finished loop")
>>> "finished loop"
```

```
try:
    print(1)
except ZeroDivisionError:
    print(2)
else:
    print(3)
>>> 1
    3
```

import module
module.function
>>> This is module

Major 3rd-party libraries: Django, CherryPy

scraping data from websites
Beautiful Soup

Flask ⇒ web frameworks