Python in Web Development

Recap

- How to declare a class
- Inheritance
- Encapsulation
- Polymorphism

LAB 4

Exceptions, Iterators, Generators and Debugging

AGENDA

- Exception handling
- Iterators
- Generators
- Debugging

Exception Handling

```
>>> def divide(x, y):
...     try:
...         result = x / y
...     except ZeroDivisionError:
...         print("division by zero!")
...     else:
...         print("result is", result)
...     finally:
...         print("executing finally clause")
```

Catch multiple exceptions

```
... except (RuntimeError, TypeError, NameError):
...     pass
```

Raising an exception

```
>>> raise Exception()
```

Exception hierarchy

The Iterator Protocol

Iterable

- a class that implements __iter__()

Iterator

- a class that implements __next__()

More info at:
http://nvie.com/posts/iterators-vs-generators/

The Iterator Protocol

```
>>> lst = [1, 2, 3]
>>> it = iter(lst)  # get an iterator from an iterable
>>> next(it)
1
>>> next(it)
2
>>> next(it)
3
>>> next(it)
StopIteration
```

Exercise 1

Using the iterator protocol, implement a Fibonacci class and iterate over it:

```
>>> for num in Fibonacci(100):
...     print(num, end=’ ’)
1 1 2 3 5 8 13 21 34 55 89
```

Generators

- generator functions
- generator expressions

Generators

Generator functions:

- any function with a ***yield*** statement

```
def count_until(n):
    for i in range(n):
        yield i
```

```
>>> for a in count_until(100):
...     print(a)
0
1
...
100
```

Generators

Generator expressions:

- like comprehensions, but with *()* instead of *[]*

```
>>> for a in (i for i in range(100)):
...     print(a)
0
1
...
100
```

Files are iterable

```
>>> with open(‘pg100.txt’, ‘r’) as fp:
... for line in fp:
...    print(line)
```

More:
https://docs.python.org/3.6/tutorial/inputoutput.html#reading-and-writing-files

Exercise 2

Using generators(expressions or functions), write a program that prints word count of a given word:

```
>>> word_count(‘love’, path=’shakespeare.txt’)
2018
```

http://www.gutenberg.org/cache/epub/100/pg100.txt

Debugging

- Interactive, inside the interpreter
- Pdb builtin module; navigate in stacks.
- Step-in, Step-over, Continue, Jump, Break, Up, Down, List Frames

```
>>> import pdb; pdb.set_trace()
or
$ python3 -m pdb module.py
```

Exercise 3 (optional)

Using generators, implement a function that computes prime numbers upto a given number

```
>>> for prime in primes(n=100):
...print (prime, end='\n')
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97]
```

Resources

- David Beazley: Generators: The Final Frontier
- David Beazley - Python Concurrency From the Ground Up
- Beyond PEP 8 -- Best practices for beautiful intelligible code

Quiz time :)

https://goo.gl/forms/lFw6PvAUA2Rit9kA2

Homework

- Learn Python the Hard Way, ex. 45 - 48

Thank you!