Interview Preparation – Python/JS/Application development

Python

[note: I have included some references as links ... but you should do a google search and find out more information yourself]

1) Read and work out the code samples / understand the concepts presented in this document: http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html (google search: code like a pythonista)

2) Review Classes / Objects: Exercises 40 to 44 of http://learnpythonthehardway.org/book/ (learn python the hard way)

3) Basic Object Oriented Programming terminology: Inheritance, multiple inheritance, constructor, base class/derived class, has-a relations and is-a relations, inheritance vs composition.

How does a derived class override the methods in a base class?

How can a base class method be called from an overriden derived class method?

What is meant by “super”?

What are static methods? How do you create them in Python?

What does __init__ do?

4) Read and work out the code samples given here: http://www.tldp.org/LDP/LG/issue54/pramode.html (it is an old article, please let me know if there are any problems in the code)

5) Read and work out the code samples: http://linuxgazette.net/109/pramode.html (functional programming: lambda, closures, higher order functions etc)

6) Write your own versions of the following python functions: map, filter, reduce

7) Learn about generators by reading and working out code in: http://linuxgazette.net/100/pramode.html (old article ... from __future__ import generators in not required in modern python)

8) What is an iterator in python? How do you create one? How does a python “for loop” actually work?

Read http://www.diveintopython3.net/iterators.html and learn how to create a class based iterator (example given in the document: a simple fibonacci iterator)

9) How does a Python decorator work? http://simeonfranklin.com/blog/2012/jul/1/python-decorators-in-12-steps/ (google search: decorators in python)

10) Learn to use regular expressions in python (refer google python class – you can try doing the “baby names” exercise once again)

11) Learn to use all the basic data structures (string, list, tuple, dict, set) properly. You should be familiar with all the basic methods (eg: “join” on strings, “append”, “insert” etc on list, slicing operations).

12) Learn about mutability/immutability. Find out which of the basic data structures are mutable and which all are immutable

13) Understand the concept of “references” in Python. That is, stuff like:

```
a = [1,2,3,4]
b = a
b.append(10)
print a # prints [1,2,3,4,10]
```

“a” is simply a pointer which points to a block of memory containing 1,2,3,4. b=a simply copies the address in “a” to “b”.

14) What is meant by “reference counting based garbage collection”?
http://en.wikipedia.org/wiki/Reference_counting (note: Python uses reference counting; it also makes use of some more advanced automatic memory management techniques)

15) Learn to use the “pickle module”: https://wiki.python.org/moin/UsingPickle

16) Learn about different ways to use the “import” statement

17) Learn about exception handling in Python. You should be able to write code using: try, except, else, finally, raise

18) Show how you can define your own exception types as classes. You should also demonstrate some code which “raises” such user defined exceptions.

19) Learn the basics of unit testing in Python: http://www.diveintopython.net/unit_testing/
http://en.wikipedia.org/wiki/Unit_testing

20) Why should we use “xrange” instead of “range” in a for loop?

21) Explain how the “else” clause works with a while loop:
http://stackoverflow.com/questions/3295938/else-clause-on-python-while-statement

---

22) Understand the various ways in which function parameters can be defined in Python.

For example:

```
def fun(a, b=10):
   print a, b

fun(1)
fun(1,2)
```

```
def fun(a, *b):
  print a, b

fun(1,2,3,4,5)

def fun(a, b, **c):
  print a, b, c

fun(1, 2, p=10, q=20, r=30)
```

Reference: https://docs.python.org/2/tutorial/controlflow.html#more-on-defining-functions

23) Learn to use a few functions from the “itertools” module:
https://docs.python.org/2/library/itertools.html (eg: cycle, repeat, chain, dropwhile, takewhile)

24) Quickly go through some of the modules available in the standard library:
https://docs.python.org/2/tutorial/stdlib.html

25) How do you access command line arguments in Python?

Hint: check sys.argv

http://www.tutorialspoint.com/python/python_command_line_arguments.htm

26) The Python language has many implementations; the standard one (which we are using) is called Cpython.

http://en.wikipedia.org/wiki/CPython

There are other Python implementations: examples being Jython for the JVM and IronPython for the Microsoft .NET platform.

27) Learn more about Python byte code:
http://stackoverflow.com/questions/19916729/how-exactly-is-python-bytecode-run-in-cpython

28) What is the purpose of this line in Python code:

```
if __name__ == '__main__':
    statements
```

Ref: http://stackoverflow.com/questions/419163/what-does-if-name-main-do

29) You have a Python program called “a.py”. You want to run it by just typing ./a.py. How do you do this?

Ans: First make the file executable: chmod u+x a.py (learn about the chmod command in detail)

Next, the very first line of the file should be:

```
#!/usr/bin/python
```

Reference: http://en.wikipedia.org/wiki/Shebang_%28Unix%29 (do a google search for “shebang”)

30) Read and understand as much as you can of this document:
http://sahandsaba.com/thirty-python-language-features-and-tricks-you-may-not-know.html

31) What does the “zip” function do?

32) What does “enumerate” do? In what way is it useful?

```
fruits=['apple', 'orange', 'mango']
for i, f in enumerate(fruits):
    print i, f
```

33) Mention one use of “with” in Python.

Eg:

```
with open('data.txt') as f:
     statements
```

Why do we prefer using “with” in the above context (ans: file gets closed automatically when we exit the with block)

34) What is meant by “tuple unpacking”

35) Mention 3 differences between Python 2.x and Python 3.x

36) Why is Python code slow compared to C?

37) Why do you prefer using Python over C? What are the major advantages of using Python?

38) Is it possible to call C functions from Python? (Ans: yes, possible)

Web Based Applications

1) What is HTTP?

2) What are the common HTTP methods (GET, POST etc):

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

3) What is the difference between GET and POST?

http://www.w3schools.com/tags/ref_httpmethods.asp

4) What is a URL ?

http://en.wikipedia.org/wiki/Uniform_resource_locator

5) What is a Query String?

http://en.wikipedia.org/wiki/Query_string

6) What happens when you submit a form?

http://stackoverflow.com/questions/21814294/what-happens-when-submit-button-is-clicked

7) How does the web really work?

http://www.garshol.priv.no/download/text/http-tut.html

http://101.lv/learn/CGIperl/ch2.htm

8) What is DNS?

http://en.wikipedia.org/wiki/Domain_Name_System

9) What are the layers present in TCP/IP?

http://www.omnisecu.com/tcpip/tcpip-model.php

10) What are the major protocols present in TCP/IP?

Ans: TCP, IP, UDP, ICMP, ARP, RARP, ...

http://en.wikipedia.org/wiki/Internet_protocol_suite

11) What is the role of each of these protocols: TCP, UDP, IP, ARP, RARP, ICMP?

11a) What is the meaning of a “port number” in the context of TCP/IP? On what port does an HTTP server run, by default?

12) What is HTTPS? How does the browser indicate that a URL is an HTTPS url?

http://en.wikipedia.org/wiki/HTTP_Secure

13) What is public key cryptography?

http://en.wikipedia.org/wiki/Public-key_cryptography

14) What are some commonly used open source web server programs?

Ans: Apache, Nginx

15) What is the difference between HTML and HTML5? (eg: HTML5 has support for new features like a graphics canvas ... do a google search or read “dive into html5”)

16) What is meant by DOM?

http://en.wikipedia.org/wiki/Document_Object_Model

Simple answer: your browser will read an HTML document and represent it in memory as a tree structure – this tree is called the DOM.

Javascript code basically interacts with this DOM.

17) What is the difference between Javascript (JS) and Python?

Major difference is that JS is mostly used for programming interactions within the web browser – all major browsers have a JS interpreter embedded in them (google chrome's embedded js interpreter is called v8). But these days, JS can also be used for writing standalone scripts using something called “nodejs”.

18) What is “nodejs”?

Usually, JS code runs within the browser and can't interact directly with the operating system.

“nodejs” is basically google chrome's JS engine removed from the browser and augmneted with libraries facilitating interaction with the operating system. “nodejs” can be used for writing server side applications, just like Python or Ruby.

19) What does a web framework do?

Ans: A framework automates most of the repetitive tasks involved in designing a web app – for example, handling login and authentication.

20) What is the difference between Flask and Django?

(ans: Flask is a “micro” framework whereas Django is more full fledged. Django comes integrated with its own templating engine, object relational mapper etc whereas in the case of Flask, you will have to use third party libraries for these purposes)

21) What is an ORM?

http://en.wikipedia.org/wiki/Object-relational_mapping

22) What is SQL? (make sure you can write basic SQL queries)

23) What is an RDBMS?

http://en.wikipedia.org/wiki/Relational_database_management_system

23a) What is SQL? Learn the basics of SQL by working out:

http://sql.learncodethehardway.org/book/

24) What is a NoSQL database system?

http://en.wikipedia.org/wiki/NoSQL

(examples of NoSQL databases: MongoDB, Riak, CouchDB etc)

25) What is meant by “ACID properties”

http://en.wikipedia.org/wiki/ACID

26) Why do we need database systems?

http://www.quora.com/Why-do-we-need-relational-database-management-systems

27) What are some security issues involved in writing web apps?

http://en.wikipedia.org/wiki/Cross-site_scripting

http://en.wikipedia.org/wiki/SQL_injection

28) What is meant by Database Normalization? What are the commonly used “normal forms”?

http://www.studytonight.com/dbms/database-normalization.php

http://en.wikipedia.org/wiki/Database_normalization

29) What is a “primary key” and a “foreign key”?

http://databases.about.com/cs/administration/g/primarykey.htm

http://en.wikipedia.org/wiki/Foreign_key

30) What are the two commonly used Open Source Relation Database Management Systems (RDBMS's)?

MySQL and PostgreSQL

31) What is meant by MVC?

http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

MVC is a software architectural pattern – modern frameworks like Django, Rails etc are MVC frameworks.

32) What is meant by CSS?

http://en.wikipedia.org/wiki/Cascading_Style_Sheets

33) What is XML?

http://en.wikipedia.org/wiki/XML

34) What is JSON?

http://en.wikipedia.org/wiki/JSON

Python Examples to work out

1) Use the “map” function (together with a lambda expression) to compute square of all elements of a list.

2) Use the “filter” function (together with a lambda expression) to filter out all strings whose length is less than 3 characters (from a list of strings)

3) Use the “reduce” function (together with a lambda expression) to compute sum of all elements of a list (assume we have a list of integers)

4) Use “reduce” to concatenate all the strings in a list of strings (ie, if the list is [“abc”, “def”, “ijk”], the output of “reduce” should be the string “abcdefijk”

5) Write two functions my_map and my_filter which works like the builting map and filter functions.

6) Write a list comprehension to square all the elements of a list of integers.

7) Write a list comprehension to square all even numbers from a list of integers.

8) Suppose you have a list of N integers (example: [1,2,3]). Use a double list comprehension to generate a list of ALL pairs like this: [(1,1),(1,2),(1,3), (2,1),(2,2),(2,3),(3,1),(3,2),(3,3)]

9) Modify the above comprehension so as to include only those pairs (x,y) where x-y = 0

10) Say you have a list of integers with duplicate values present. What is the easiest way to remove duplicates? (ans: convert to a set)

11) Use a “set comprehension” to square all the elements of a set.

12) What does the following comprehension do? (hint: it is a “dictionary comprehension”):

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

{k : k+1 for k in a}
```

13) Suppose you have a text file like this (called say “data.txt”):

```
this is a
simple text file
containing
two lines
```

Use a comprehension to generate a list which holds length of each line of the file. (hint: an open file handle is an “iterable” - you can use it like a list when writing a comprehension)

solution:

```
f = open('data.txt')
```

a = [len(s) for s in f]

14) Find out how to take union/intersection/symmetric difference of two sets

15) Solve the “FizzBuzz” problem using a list comprehension (http://c2.com/cgi/wiki?FizzBuzzTest). The output will be a list where each element will either be a number or one of the strings: “Fizz”, “Buzz”, “FizzBuzz” (hint: you can think of writing a small helper function to do the “mapping” operation)

16) Redo (15) using the “map” function.

17) Create an “infinite” generator (using “yield”) which will generate values 0, 1, 2, 3, 4, 5, ...

For example, if you do:

```
a = fun() # “fun” is the generator function
for x in a:
    print x
```

This will print the integers endlessly ....

18) Create a “fibonacci generator” (using “yield”) which will generate all elements of the fibonacci series.

For example:

```
a = fun()
for x in a:
     print x
```

This will print the inifinite series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

19) Create a fibonaaci iterator using a class and the __iter__ and __next__ methods (http://www.diveintopython3.net/iterators.html)

20) Create a simple decorator (without arguments) which will simply print the current time to the console when the decorated function is called.

Example:

```
@showtime
def fun():
    print 'hello'
```

Now, when you call “fun”, it will first print the current date/time (using time.asctime()) and then only, “hello”

21) Modify the decorator above to accept a file name as parameter. When the decorated function is called, the current date/time should be appended to the file (and not shown on the console).

22) Create a base class and a derived class and demonstrate the working of “super”.

23) Write code to demonstrate the working of: try, except, else, finally, raise

24) How do you write an “except” statement so that it handles multiple exceptions?

25) Create a user defined exception class (call it MyException) and demonstrate how you can “raise” this exception.

26) You can define a function like this:

```
def fun(a, *b, **c):
    ....
    ....
```

Explain the meaning of the notations: *b and **c

27) What is a “generator expression”? In which context would you use a generator expression instead of a list comprehension?

28) Write a Python function to demonstrate a “closure”:

Ans:

```
def add(a): return lambda b: a + b

f = add(10)
print f(20)
```

[check out: http://linuxgazette.net/109/pramode.html]

29) Does the following code work?

```
print True + 1

print False + 1
```