[6 marks] Read through this article on scripting by John Ouserhout (bit.ly/caseforscripting) and answer the following questions:
List Ousterhout's 3 most compelling arguments for scripting.
This article is many years old. Comment on Ousterhout's predictions for the future of scripting, specifically what did he anticipate correctly, and what was he incorrect about?

A:

In the article by John Ouserhout, a lot of comparison is made between the System programming languages like C, C++ and the Scripting languages. Scripting languages are used for the purpose of gluing application components together.

The 3 most compelling arguments provided by Ousterhout in favor of scripting are listed below:

i) Graphical User Interface: As mentioned in the article, GUI's are definitely widespread and a lot of effort is spend on the GUIs in most of the projects. A GUI is primarily intended to glue together applications and show the details in a user friendly way. There are no rapid development environments for GUI development based on system programming languages. Even the ones that exist are complicated when it comes to defining behavior of the components. Since the purpose of a Scripting language is to glue together component and a GUI is intended for a similar purpose it becomes a natural fit, and Scripting languages come in very handy for GUI development, by providing rapid-development environments.
ii) Internet: Internet, by essence is meant to glue together lots of varying individual components. It does not need to do any computations or create data. So the ideal language for Internet purposes is a language which helps in this process, by connecting various components and enabling them to work together. For this purpose, the scripting languages are the best fit since they are intended and designed for that purpose.
iii) Component Frameworks: The final example outlined in the article is the Component frameworks. The component frameworks like ActiveX and JavaBeans are created using System programming languages, but the same cannot be said about using them to assemble the components into applications. This task is left to the scripting languages. The scripting language helps in assembling/manipulating components. Without a scripting language, much of the power of component framework would be rendered useless.

I believe the Ouserhoust is correct about the following points, when it comes to the Scripting language efficacy:

i) Since scripting languages are typeless, it makes it easy to hook together components. As there are no restrictions on how things can be used, the components can be used for any situations even for the ones it wasn't initially designed for.
ii) Ouserhout correctly states that even though a scripting language is slower compared to higher level languages in terms of performance, this fact is made redundant because the machines of today are significantly faster than earlier. So as computers keep getting faster, scripting will become attractive for larger applications too, which is definitely true
iii) The other point which Ouserhout anticipated correctly is the fact that more casual programmers would become part of the programming community due to the arrival of PC. This did happen and has resulted in the increasing use of scripting, because as he mentions, casual programmers are not inclined to spend months learning a new program. If they need to develop something to help with their existing jobs, like macros etc., they want

to be able to do it in the shortest time frame possible and scripting languages definitely help in this process

iv) Ouserhout is also right about the fact that the benefits of scripting depends on the application. Scripting languages are more beneficial depending on the purpose for which it is used. As mentioned in the article, Scripting languages provide less benefits when used for the first implementation rather than for a reimplementation. These things hold true even today.

v) Also Ouserhout was spot on when he says that the scripting and system programming languages are symbiotic. They do produce programming environments of exceptional power than would have been possible, if used alone

In my opinion, the points that Ouserhout was incorrect about are the following:

i) Ouserhout mentions that the strong typing that is used in the OO Languages encourages packages that are difficult to use and the fact that if two packages need to work together, then they need to have a conversion code to translate between the types. This is not really true because since the OO languages are strong typed, it makes it more efficient when enforcing constraints on the kind of data that can be shared. This is an advantage because it helps validate the invalid casting of data

ii) It is mentioned by Oserhout in the article that inheritance is a problem with OO languages as they bind the implementation of clases together. This is not necessarily true. A very well designed application can utilize inheritance in efficient ways without causing too much brittleness.

iii) One of the things that Ouserhout expected to happen was that more applications would be written entirely in scripting languages. This is one thing which hasn't really happened. Scripting languages are definitely used a lot but they haven't taken over as the primary means of developing new applications. They are still used in conjunction with the system programming languages like earlier

[5 marks] Your boss got a hold of a case study in which a Python program has been developed ten times quicker than a program written in C++. He now demands that you immediately program entirely in Python. Tell him about five dangers of doing this.

A:

i) Python is a Dynamically typed language. As a result, we can avoid using types to declare variables but this ends up making it difficult in terms of ease of debugging. In C++, we can clearly identify which variable is being returned but it is not as easy in Python, especially in a long and complex method. The same point holds true for an argument too. In C++, we know the type of the argument being passed around but not in Python

ii) Because Python doesn't have primitives, everything including the attribute types are Objects. Therefore Python values have to store information about its types, which is an overhead

iii) A python program might be easier to develop but it is not really as fast as a C++ program can be. C++ programs allow a lot of optimizations to be made to achieve this, but this is not correspondingly possible in Python.

iv) Python is a Interpreted Language, in which the code is passed through a Python Interpreter and that performs the translation from high-level code to low-level operations on the fly, every time the program. In case of C++, the code is compiled using a Compiler and converted into an executable file. After this, the Compiler is not needed again unless there are code changes, unlike Python where the interpreter is needed for every execution of the code.

v) Since Python is executed through an interpreter, it is slower than a C++ program which is pre-compiled. Python hogs more memory and is generally more slower than a C++ code

vi) Python has less backwards compatibility and since majority of applications are developed using C++, it is a drawback to blindly give up the existing C++ code base

vii) During run time, Python must work harder than C++. When Python tries to evaluate an expression, it has to examine the objects and figure out their type as it is not known at compile time.

viii) The advantage of having an executable in C++ is that it can be easily distributed to customers as a free-standing software as long as it was designed for a particular machine and it can be executed by users without any further software installation requirements. In case of Python, there is no distinction between compile-time and run-time and the Interpreter has to always translate the source code as part of run-time. This means that the software can only be distributed to a user who has compatible interpreters.

[3] Briefly, a Java interface is a kind of type specification. Specifically, it specifies a group of related methods with empty bodies. (All methods are nonstatic and public.) Interfaces have many benefits over, for example, abstract classes - in particular, they are a good way to implement polymorphic functions in Java.

Describe the advantages and disadvantages of Python and Java for polymorphism. In particular, how would you implement a polymorphic function in Python?

A:

Python is a dynamically typed language and as such has polymorphism everywhere. Every operation in Python is polymorphic.

i) Data types are not specified for function arguments in Python. As long all operations on a function can be carried out with the given values, the program in Python executes correctly. This is a huge flexibility offered by Python. It basically allows a single function to be executed for different objects without having to define multiple version. In Java, we would have define the class and methods using generics in order to achieve such a behavior.

ii) Python is polymorphic in nature. It is easy to write polymorphic functions in python unlike Java where it can tend to be more complicated.

iii) In case of Java, it is essential that the polymorphic classes have the same interface or be sub-interfaces of a common parent interface or be subclasses of a common super class. This is referred to as “strong, hierarchical typing”. Python does not need to enforce such a rigid type rule and follow the subclass/subinterface hierarchy

iv) One of the indicators for poor use of polymorphism in python is using the types, isinstance and issubclass functions to determine the class of an object. These should be used rarely or not all. The processing should be focused on what is different about the objects rather than the class to which an object belongs

v) The type of Polymorphism that is implemented in Python is called “Duck typing”. This phrase is from a quote attributed to James Whitcomb Riley: "When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck." It basically means that two objects are effectively of the class Duck if they have a common collection of methods (walk, swim and quack, for example). In case of Java the desired polymorphic functionality can be achieved by Upcasting or Downcasting. However even though Upcasting has the advantage of providing a uniform interface to the clients and that the changes in the class do not affect clients, there is a disadvantage in that, the object must be explicitly downcast if type details are needed in clients after object has been handled already.

vi) The advantage of treating all the subclasses of a class in a uniform way is that we effectively delegate any kind of special processing involved to the relevant subclass.

vii) Polymorphism in high level languages like Java tend to be esoteric and it makes it difficult for beginners to understand without digging deeper. In case of Python, since it is already designed for that purpose, it is very easy for any person with a basic knowledge of programming to understand the functionality

viii) In Java, polymorphism is constrained by the inheritance hierarchy which is not the case in case of Python, so that's a distinct advantage.

Polymorphic function Implementation:

```
class Car:
        def accelerate(self):
                print "Press the Accelerator to Increase Speed"
        def brake(self):
                print "Press the Brake to Reduce Speed"

class Motorcycle:
        def accelerate(self):
                print "Roll the throttle to Increase Speed"
        def brake(self):
                print "Release the throttle while pressing the Brake to Reduce speed"


def roamAround(vehicle):
        vehicle.accelerate()
        vehicle.brake()

roamAround(Car())
roamAround(Motorcycle())
```

Output:

```
        Press the Accelerator to Increase Speed
        Press the Brake to Reduce Speed

        Roll the throttle to Increase Speed
        Release the throttle while pressing the Brake to Reduce speed
```

[4] Define a student class in Python. The class should include name, GPA and age fields. Implement __str__(), __lt__(), __eq__(), and __hash__() methods. Write test code that exercises these methods using sorted() and dict().

A:

The student class has been created with the required fields and the sorted() and dict() methods are implemented which make use of the __str__(), __lt__(), __eq__(), and the __hash__() methods.

Implementation is uploaded to GIT at the locations:

HW1_Student.py

Personal GIT:

https://github.com/asimsaleem/APT_Fall2014/blob/master/hw1/HW1_Student.py

Official GIT:

https://github.com/apt-fall13/student-Asim-Saleem/blob/master/hw1/HW1_Student.py

[2] Create a test array of 5 students and sort in order of increasing GPA using a lambda expression. Break ties using name then age.

A:

The code to sort an array of 5 Students using lambda in increasing order of GPA, breaking ties using Name and Age is given below:

```
        sorted_list = sorted(testArray, key=lambda x:(x[2], x[0], x[1]))
```

Sample Data Used:

```
#Name, Age, GPA - GPA gets repeated. Sorting on Name and Age
testArray = [["Adam", 25, 3.0],
             ["Bart", 24, 4.0],
             ["Carl", 23, 4.0],
             ["Deng", 22, 2.0],
             ["Deng", 21, 2.0]]
```

Output:

```
Test Array is:  [['Adam', 25, 3.0], ['Bart', 24, 4.0], ['Carl', 23,
4.0], ['Deng', 22, 2.0], ['Deng', 21, 2.0]]
```

```
Sorted List is:  [['Deng', 21, 2.0], ['Deng', 22, 2.0], ['Adam', 25,
3.0], ['Bart', 24, 4.0], ['Carl', 23, 4.0]]
```

It can be observed that the list was first sorted based on GPA, and then if there were ties, using the Name and the Age.

Implementation is uploaded to GIT at the locations:

HW1_StudentSorting.py

Personal GIT:

https://github.com/asimsaleem/APT_Fall2014/blob/master/hw1/HW1_StudentSorting.py

Official GIT:

https://github.com/apt-fall13/student-Asim-Saleem/blob/master/hw1/HW1_StudentSorting.py

[1] Use the iPython %timeit command to find the largest array of random [0,1] floats that can be sorted in 10ms, 100ms and 1000ms on your computer.

A:

The array size required for the random[0, 1] array of floats to get sorted in 10ms, 100ms and 1000 ms are provided below.

Time	Array Size of Random Floats (Approx.)
10ms	Approx. 39000
100ms	Approx. around 277,000 to 278,000
1000ms	Approx. 1,720,000

Sample Implementation:

```
In [1]: import random

In [2]: floatList=[]

In [3]: rangeValue=39000

In [4]: for i in range(rangeValue):
   ...:     floatList.append(random.random())
   ...:
```

```
In [5]: print "Length of the Float List is:", len(floatList)
Length of the Float List is: 39000

In [6]: timeit(sorted(floatList))
100 loops, best of 3: 10 ms per loop
```

[5] Use iPython to debug this program. Specifically, read the specification, write tests, and use the debugger to root cause the failure.

A:

On testing it is noticed that the program does not lay down the cards even for a combination where for a particular rank, all the conditions are met.

We can also observe that the code lays down the cards for a condition where there are 4 items in the “hand” dictionary and one of the deck values matches any of them, even when the conditions expected are not met.

On further analysis, it is noticed that the problem occurs in the method:

def drawCard(name, deck, hand) at the point:

if len(hand) == 4:

The code checks to see if the length of the “hand” dictionary that is being passed to it is of size 4 or not. It doesn’t take into consideration whether the cards in the hand are of the same rank or not. As a result, even in the scenario where all the conditions outlined in the problem are met, the code does not lay down the cards.

To fix this, the change required is to make sure that when the code checks the length of the “hand” dictionary, it does so on the basis of the type of the rank.

Thus by changing the code error line from

if (len(hand) == 4)

to

if(len(hand[cardRank]) == 4)

we can fix the issue encountered. The code will now specifically check for a specific rank length in the hand to determine if it should be laid down or not.

Since Python is not strongly typed and allows same operator to be used for different kinds of data, the program did not fail during execution initially, even though the output was intrinsically wrong.

Implementation is uploaded to GIT at the locations:

Personal GIT:

https://github.com/asimsaleem/APT_Fall2014/tree/master/hw1/debug

Official GIT:

https://github.com/apt-fall13/student-Asim-Saleem/tree/master/hw1/debug