Assignment

Python Class 1:

- What is JPython & CPython

Answer- The "C" part in "**CPython**" refers to the language that was used to write **Python** interpreter itself. **JPython is** the same language (**Python**), but implemented using Java. Implementation means what language was used to implement **Python** and not how **python** Code **would** be implemented

- Basic difference between Python2 & python3

Answer-

1. Integer Division:-

In python 2 if we perform division on two integers then the output will be an integer too. But in python 3, output will be accurate, so the result can be in float too. Still want the result in integer, then you can use *print(9//2)* it return an integer result.

2. Print Function:-

In python 2 parenthesis aren't compulsory to use we can print anything without using parenthesis but in Python 3 it is compulsory to use parenthesis otherwise it will raise error.

3. Unicode:-

Python 2 has ASCII str() Types, separate Unicode but it doesn't have byte type.

But in Python 3 we have Unicode (utf-8: 8 means it uses 8-bit block to represent a character) strings and also 2 byte classes that are bytearray and byte.

Python 2 treats string and bytes as same, so we can also concatenate them. But in Python 3 we can't concatenate a byte array with strings, because both are different for python 3**.**

4. xrange function:-

python 2 has two handy function for creating a range of integers that is used in **for** loop, these are **range** and **xrange**. They both provide a way to generate a list of integers. So for the most part, **xrange** and **range** are the exact same in terms of

functionality. The only difference is that **range** returns a Python **List** object and **xrange** returns an **xrange** object. **range** function creates an array of integers, so it will take much memory if we create a range of millions, which can result **MemoryError** and crash your program. So **xrange** is the function to use if you've a really memory sensitive system such as cell phone.

But in python 3 there is no **xrange** function, the **range** function will work as **xrange** in python 2.

5. Raising exception:-

as we've seen the difference between print function, raising an exception will follow the same syntax.

In python 2, its

raise IOError, "file not found"

In python 3, its

raise IOError("file not found")

6. Handling exception:-

there is a minor change in syntax, we've to use **as** keyword in python 3.

In python 2, its

except IOError, err:

In python 3, its

except IOError as err:

7. Leak of for-loop variables in global namespace:-

In python 2 there was a problem that value of a global variable had changed while using that variable in for-loop.

But in Python 3, for loop variables don't leak into the global namespace anymore!

8. .next() method:-

Python have .next() method and next() function to fetch the next element of an iterator.

But in python 3 .next() method is no more. We have to use only next() function to iterate the next element of iterator.

9.input() method:-

python 2 have input() and raw_input() methods for taking input. The difference between them raw_input() returns a string, and input() tries to run the input as a python expression.

Mostly all we want the string as input then we convert it into any datatype as we want.

In python 3 there is no raw_input() method. The raw_input() method is replaced by input() in python 3. If you still want to use the input() method like in python 2 then we can use eval() method.

eval(input("enter something:"))

it will work as same as input() in python 2.

- Difference between ASCII & unicode

Answer- ASCII defines 128 characters, which map to the numbers 0–127. Unicode defines (less than) 221characters, which, similarly, map to numbers 0–221 (though not all numbers are currently assigned, and some are reserved).

Unicode is a superset of ASCII, and the numbers 0–128 have the same meaning in ASCII as they have in Unicode. For example, the number 65 means "Latin capital 'A'".

Because Unicode characters don't generally fit into one 8-bit byte, there are numerous ways of storing Unicode characters in byte sequences, such as UTF-32 and UTF-8.

C follows ASCII and Java follows UNICODE.