**Introduction**

Data too big (big data) to be processed on a single machine needs distributed processing. Big data has three Vs, **Volume**, **Variety**, and **Velocity**.

**Data types: builtins**

**zip([iterable, it2, it3) 🡪** list of tuples sequencing from each iterable

**Data types: sets**

**set.add(‘s’) 🡪** adds ‘s’ to the set

**set.update(‘s’) 🡪** incorporates new components to set

**a.update(‘b’) 🡪** update a set of components in b

**set.discard(‘s’) 🡪** throws out a component from the set

**set.removes(‘s’) 🡪** throws out ‘s’, throws error not there

**set.pop() 🡪** throws out the last components

**a.difference(b) 🡪** components in “a” and not in “b”

**a.union(b) 🡪** a U b on set a and set b (their union)

**a.interesction(b) 🡪** only components that are common

**a.difference(b) 🡪** components in “a” and not in “b”

**a.symmetric\_difference(b) 🡪** union(a,b)- intersection(a,b)

**sorted(set\_s 🡪** sorts elements in set\_s

**Data types: lists**

**set.add(‘s’) 🡪** adds ‘s’ to the set

**String Manipulation**

**str.isalnum() 🡪** is string alphanumerical

**str.isalpha() 🡪** is string alphabetical

**str.isdigit() 🡪** is string digit

**str.islower() 🡪** is string lower case

**str.isupper()🡪** is string upper case

**str.count(substring)🡪** how many time subs occurs in str

**str[:3]+’char’+str[4**:**]🡪** to change strings, they are immutable

**str.split(“char”)🡪** list of char separated strings

**“char”.join(list) 🡪** a string of char spaced from list

**Itertools**

**product** 🡪 cartesian product of input iterables

**product (A, B)**🡪 equals ((x,y) for x in A for y in B)

**permutations([1,2])** 🡪 [2,1],[1,2]

**groupby()** 🡪 key pairs of group and their iterations

**groupby(condition, list)** 🡪 keeps the list when cond==0

**Collections**

**counter** 🡪 generates a dict {element: its frequency}

**defaultdict()** 🡪 same as dict, only data types are defined

**namedtuple()** 🡪 generates tuple subclass

**deque()** 🡪 double ended quee

**Cmath**

**phase(x,y)** 🡪 atan(y,x)

**abs(x,y)** 🡪 sqrt(x\*\*2+y\*\*2)

**complex(string)** 🡪 generates a complex number

**Numpy**

**transpose(array)** 🡪 transposes an array

**array.flatten()** 🡪 vectorizer the whole matrix

**inner(A,B)** 🡪 inner product

**outer(A,B)** 🡪 outer product

**Datetime**

**datetime.timestrip(string, format)** 🡪 string to date

**Some handy functions**

**getattr(data, attributename)(param)** 🡪 generates an attribute

**filter(condition, list)** 🡪 keeps the list when cond==0

**reduce(lambda x,y:x+y, list)** 🡪 calculates sum from beginning to the end

**eval(string expression)** 🡪 evaluates the expression as it is not a string, e.g., eval(print(2))-- >> 2

**Regular Expressions**

**re.search(string, text)** 🡪 true if string in text

**re.match(string, text)** 🡪 true if string begins the text

**re.findall(string, text)** 🡪 returns all non-overlapping matches

**re.match(string, text)** 🡪 true if string begins the text

**re.finditer(string, text)** 🡪 returns mathobject where you can use .group() to unpack

**re.start(group)** 🡪 returns start index of match by groupy

**re.end(group)** 🡪 returns end index of match by groupy

**visualize your regex** 🡪 <https://www.debuggex.com/>