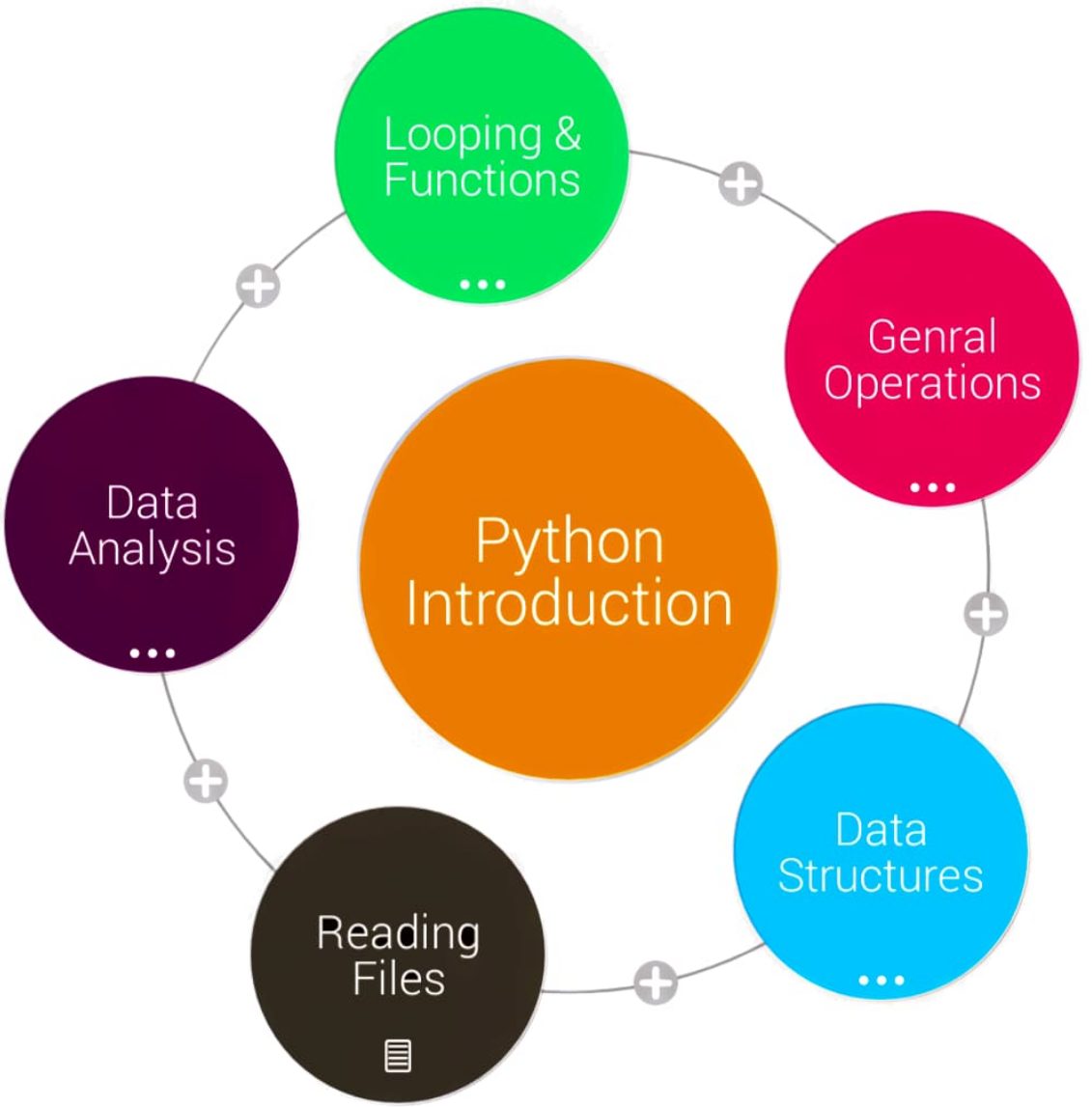
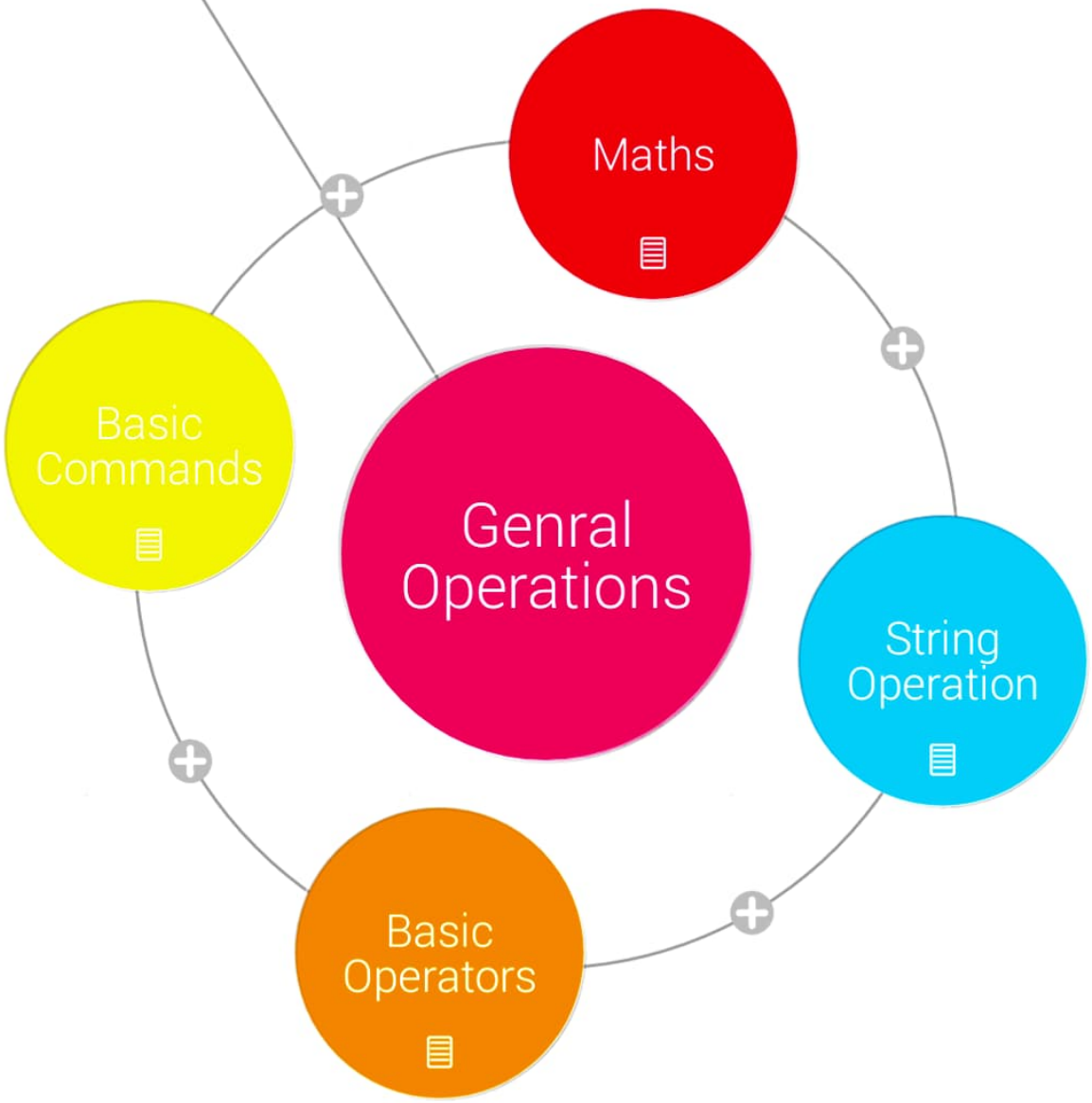
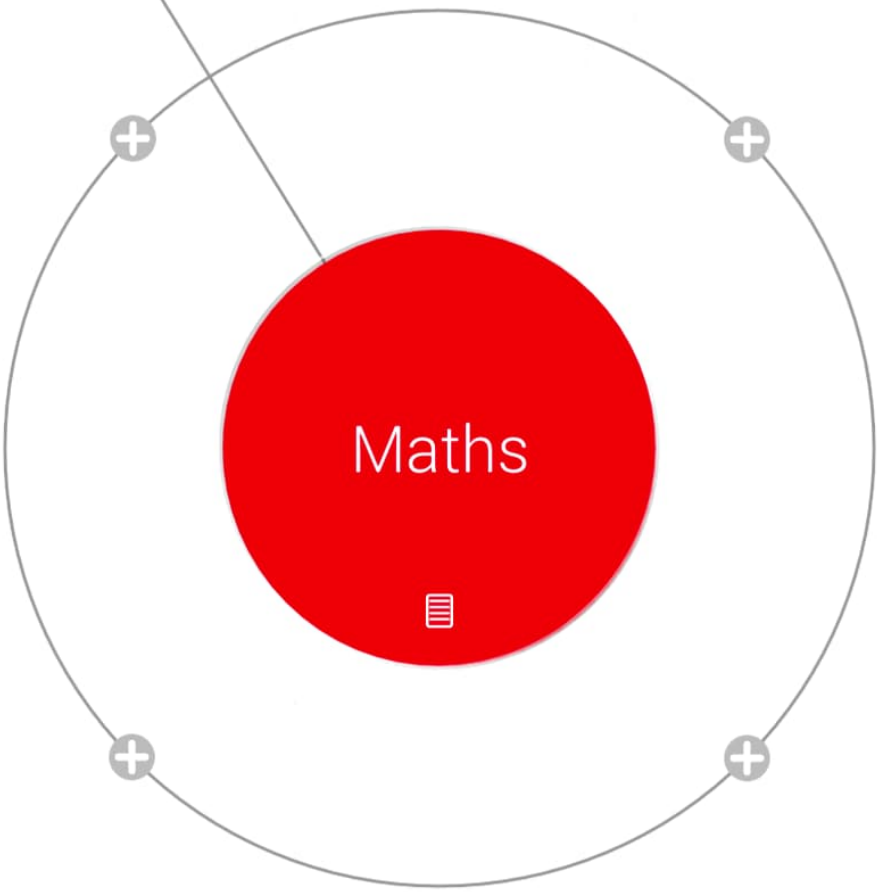

Looping &
Functions
Genral
Operations
Data
Analysis
Python
Introduction
Data
Structures
Reading
Files

Python Introduction



Genral Operations



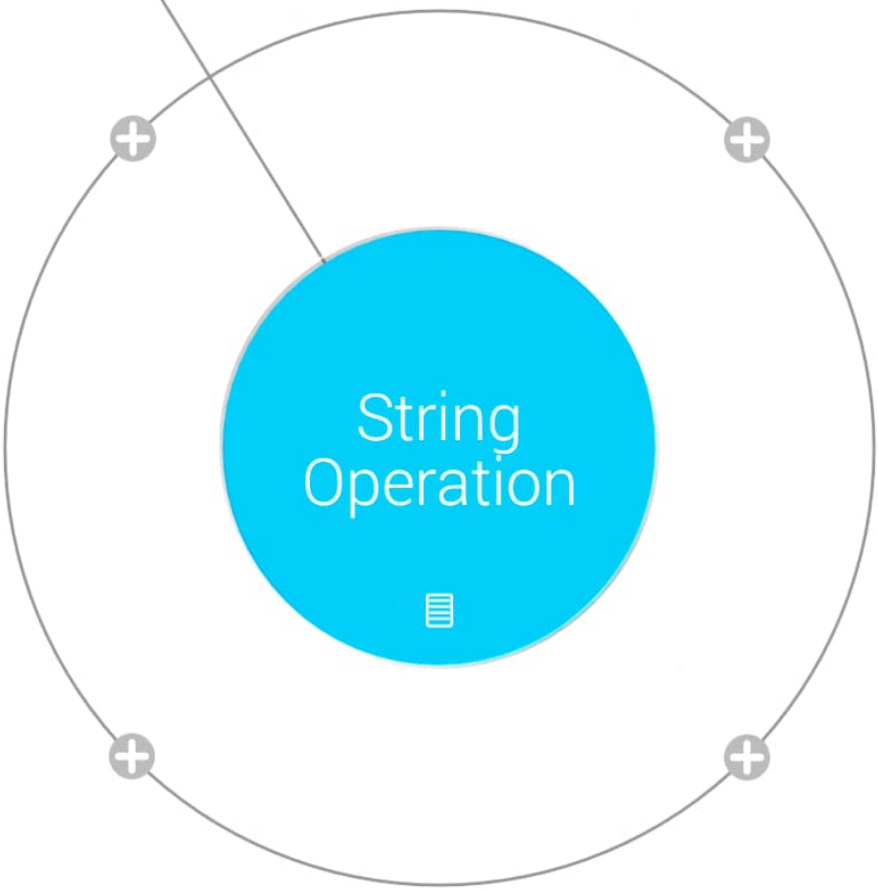
Maths

1. A//B - (Gives quotient)
2. A%B - (Gives remainder)

3. math.factorial(x)
   (Finds the factorial of x)

4. np.pi - (For pi operations)
5. round(x,y)
   (Round off a number x to y places)

Genral Operations



String Operation

1. String Declaration:
   A = " Michael Jackson"

2. A[0] - (Output : M)

3. A[0:4] - (Output : Mich)

4. len(string name) - (For string length)

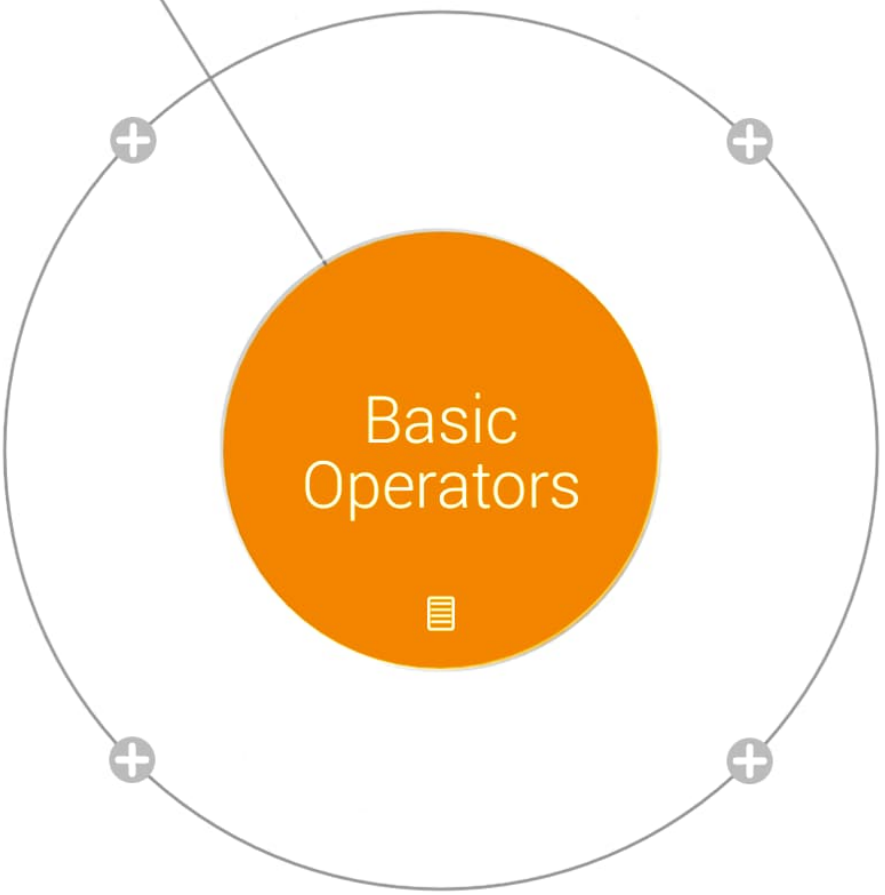
5. Concatenate Strings:
   B = A + "is best"
   Output : 'Michael Jackson is best"

6. stringname.upper()
   (Converts a string to uppercase)

7. stringname.replace('Hello' , 'Hi')
   (Here Hello is replaced with Hi in string)

8. stringname.find('Value to be searched')
   (Returns index of the searched value)

Genral Operations



Basic Operators

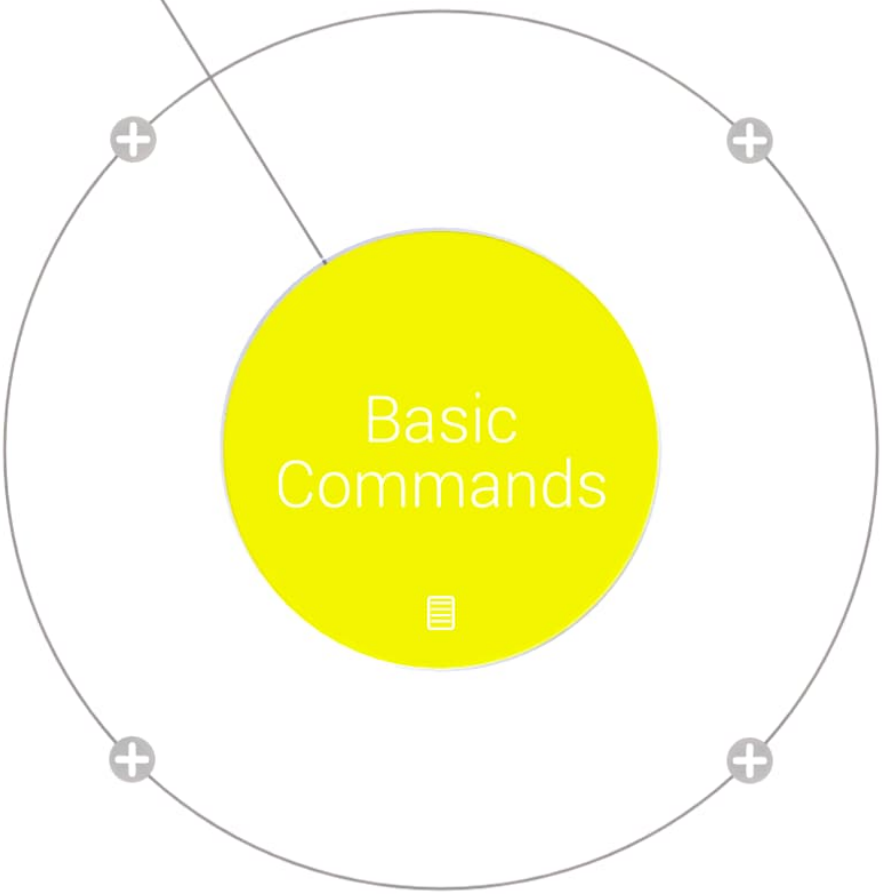
1. word = 'encyclopedia'
   Command: 't' in word (Output: False)

// in operator checks the particular value in a variable. It is applicable for all data structures.

2. + operator
   It concatenates two data structures/types

3. * operator
   It repeats the value to *x times.

Genral Operations



Basic Commands

1. type(x) : (Finds the data type of x)

2. datatypeA(x)
   (Changes x to Data type A (Typecasting))

3. len(x) : (Finds length of set/list/tuple)

4. S = sum(x) : (Adds all element in List/Set X)

5. S = sorted(x) : (Ascending order of List/Set)

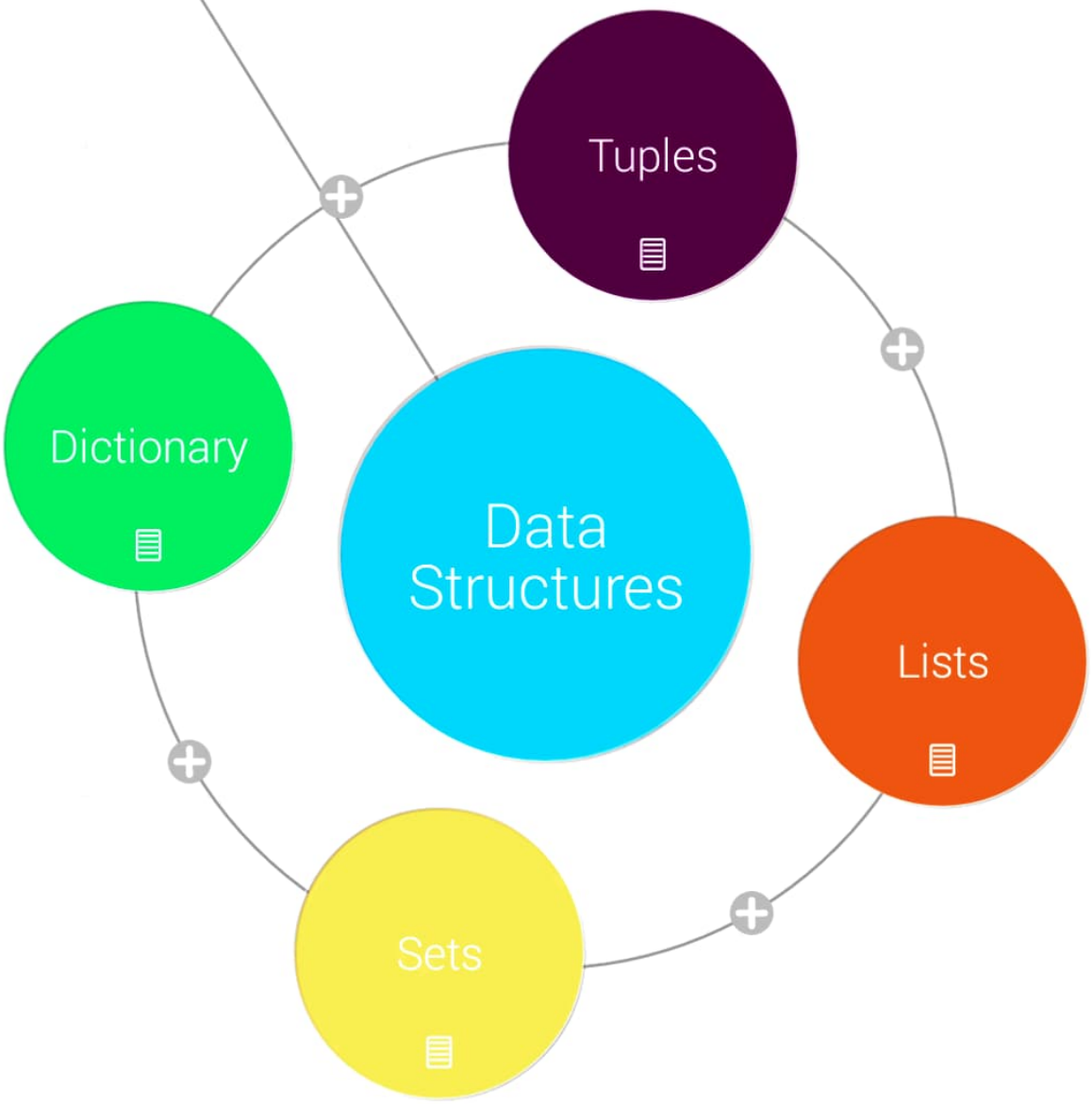
6. S = sorted(x, reverse = True) :- (Descending)

Printing a value:

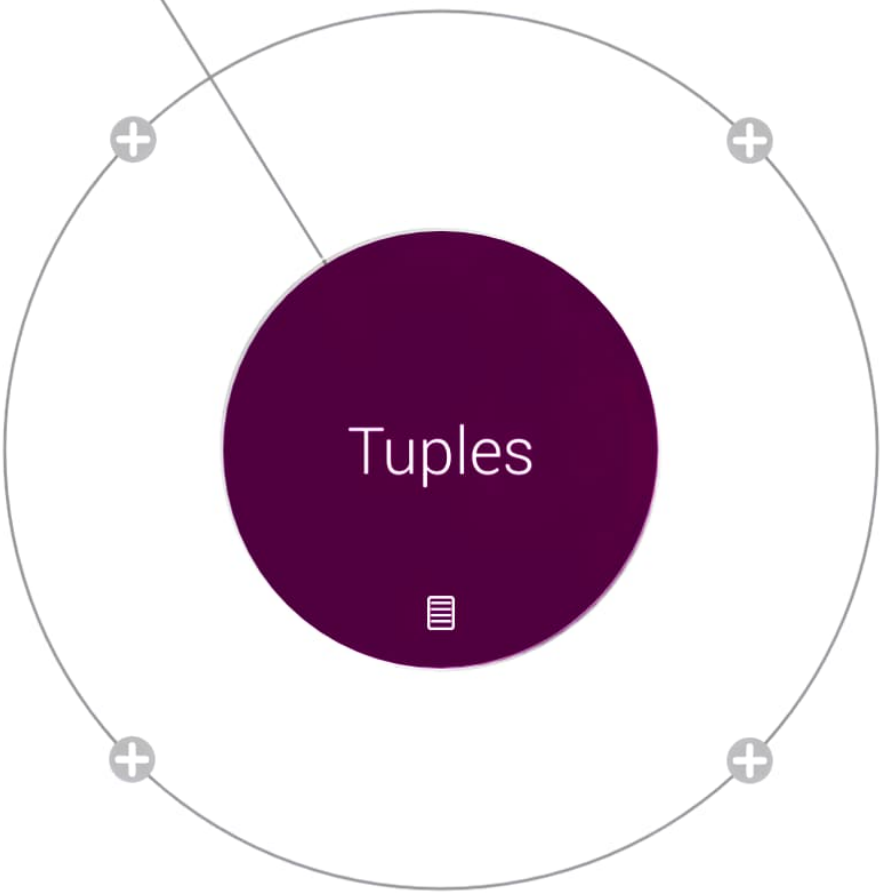
```
A = "Apple"
print("The name of Fruit is : " + A)
```
[Out] : The name of Fruit is Apple.

Alternative :
```
print("The name of Fruit is : {}".format(A))
```

Python Introduction



Data Structures



Tuples

1. Tuple Declaration, can be of any data type.
   A = (1, "abc" , 2.3)

2. Can be indexed, sliced and concatenated.

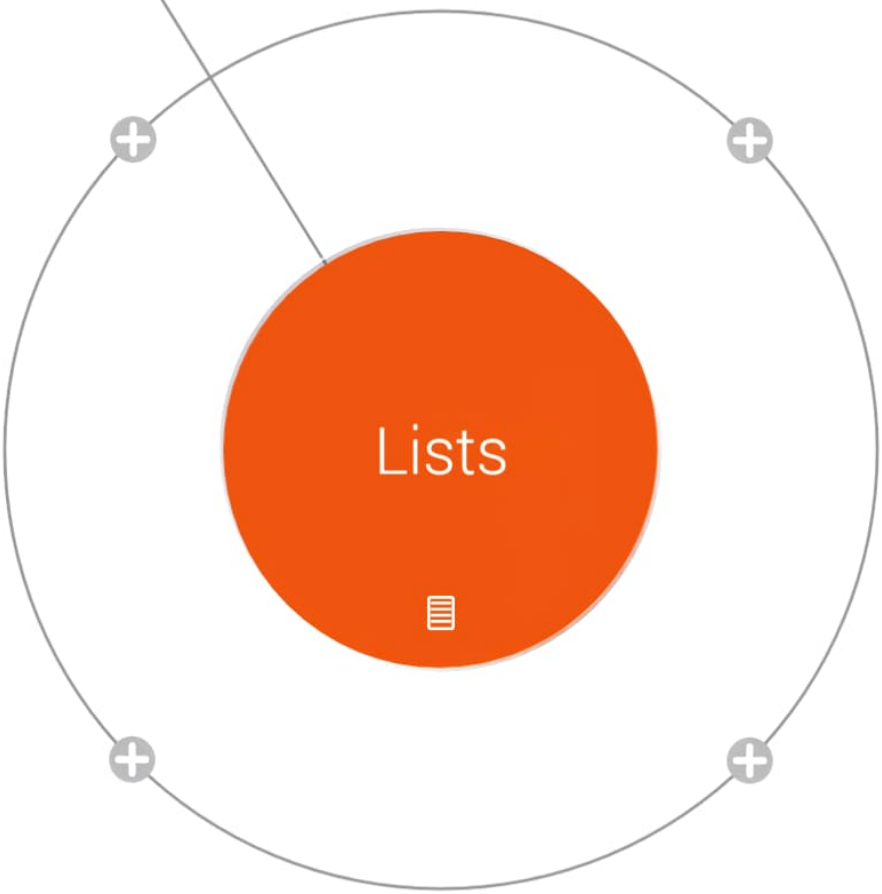
3. Tuple_B = sorted(Tuple_A)
   (sorts in ascending order)

4. Can be nested. Example :-
   A = (1, 2, ("POP", "ROCK"), 3)
   A[2][0][2] - (Output: P)

5. Typecasting: list(a)
   Converts tuple to list

// Immutable in nature. Values can't be changed here.

Data Structures



Lists

1. Declaration of Lists
   A = [1, 2, "ABC"] ; B = [3, 4]

2. A[0] = 23
   (Changes element no. 1 to 23)

3. Addition of Lists: (A + B)
   (Output: [1, 2, "ABC", 3, 4]

4. Listname.extend([List_1])
   (Adds List_1 at the end of Listname)

5. Listname.append([List_1])
   (Does the same, but indexed as 1 element)

6. del(listname[Element index])
   (Deletes an element from list for index no)

7. B = A[:]
   (Copies list A to B)

8. Conversion of String to List:

   A = "Apple is fruit"
   A.split()
   (Output : [" apple", "is", "fruit"])

7. B = A[:]
   (Copies list A to B)

8. Conversion of String to List:

   A = "Apple is fruit"
   A.split()
   (Output : [" apple", "is", "fruit"])

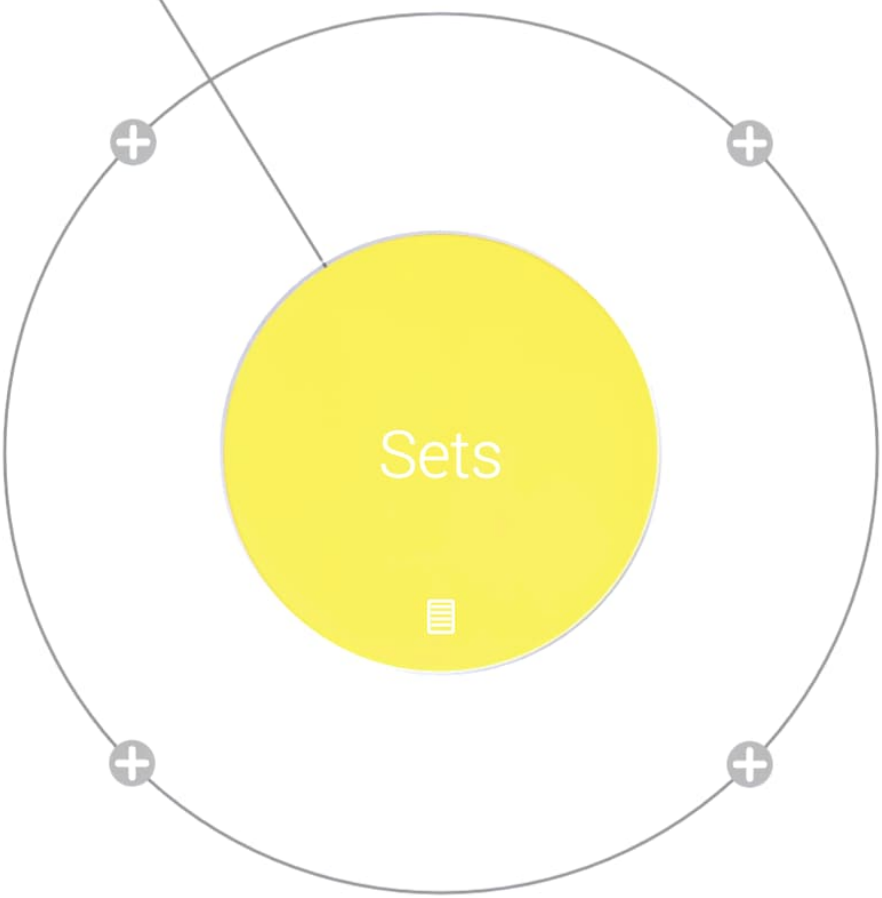
9. max(listname) and min(listname) returns max and min value from list

10. Listname.remove('Element')
    // Element gets removed from list

11. Listname.pop('Element')
Element get removed and viewed at same time

12. Listname.insert(1, 'ABC')
At index 1 ABC is inserted in the list name.

Data Structures



Sets

Features of sets:
No duplicate values are present.

1. Sets Declaration:
   Set1 = {"Apple", "Ball", "Cat"}

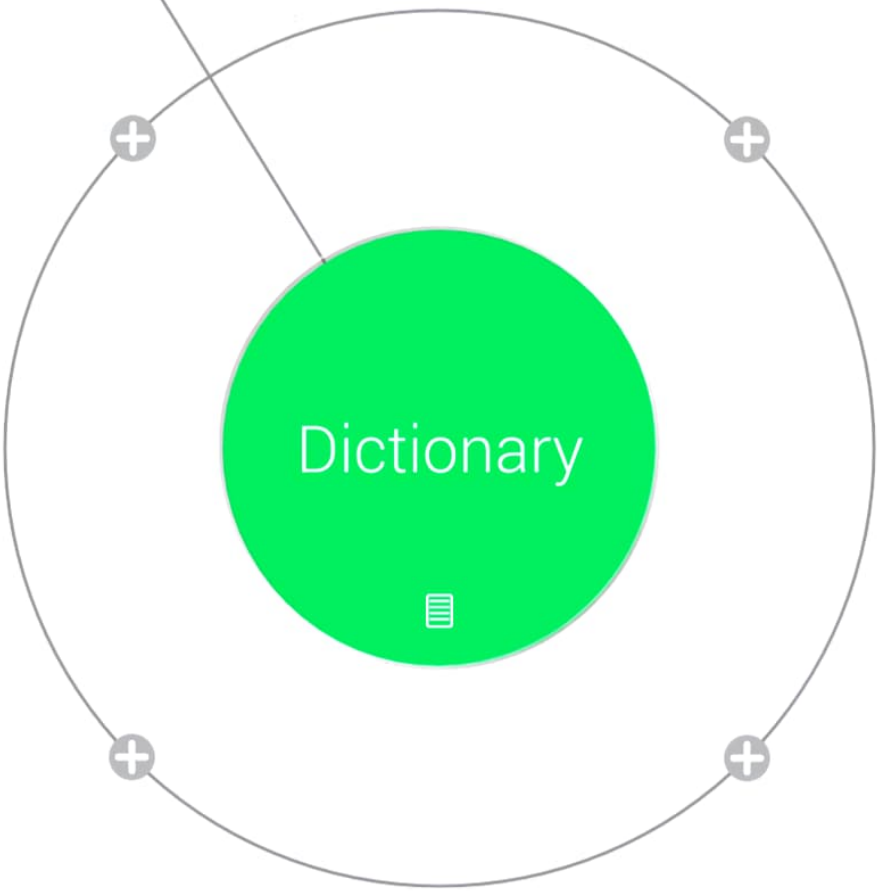
2. setname.add("...")
   (Adds a value to set)

3. SetC = set1 & set2
   (SetC holds value of intersection of 2 sets)

4. SetC = set1.union(set2)
   (SetC is the union of both the sets)

5. set(listname)
   (Typecasts a list to a set)

Data Structures



Dictionary

1. Declaration of Dictionary
   DICT = {"key 1": 1, "key 2": [3,3,3]}

   If we type, DICT["key 2"]
   (Output : [3,3,3])

2. Adding new keys to dictionary DICT:
   DICT[' Key name '] = Value of key

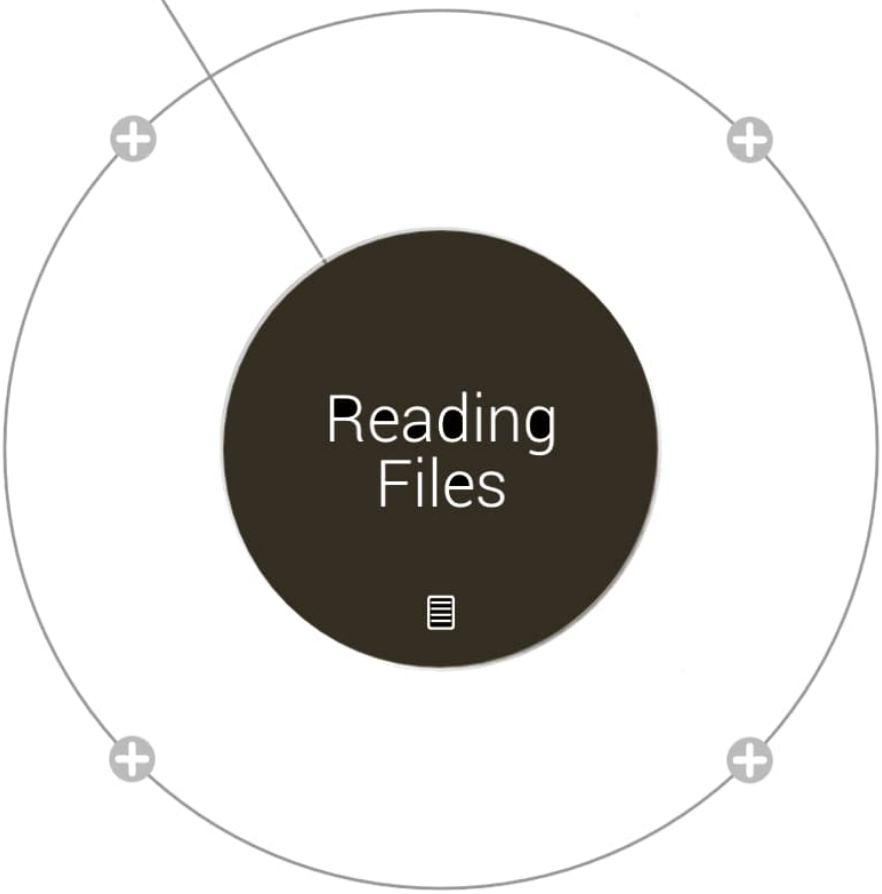
3. Deleting a key from dictionary DICT:
   del(DICT['Key name'])

4. Printing out Keys and Values:

   Dictionaryname.keys()
   (Prints all the keys used in dictionary)

   Dictionaryname.values()
   (Prints all the values in dictionary)

5. Dictionaryname.update({'key 1': 56})
   Updates value of key 1 to 56 from 1


Reading
Files

Reading Files

Downloading a file from a url in Python:

```
import urllib
from urllib import request
urllib.request.urlretrieve('url', filename='...')
```

Filename = name of the file (include extension)

Alternative:

```
!wget -q -O 'filename.extension' URL
```

This filename can be later used at various places.. ex: with open command or in pandas dataframe as..

```
df = pd.read_csv(filename)
```

Reading a file:

```
File = open('file_name', r).read()
```

(Reads and stores file in variable File)

```
File.readlines(4)
```

(Reads the first 4 lines in file)

Reading Files

dataframe as..

df = pd.read_csv(filename)

Reading a file:

File = open('file_name', r).read()
(Reads and stores file in variable File)

File.readlines(4)
(Reads the first 4 lines in file)

Reading a json file:

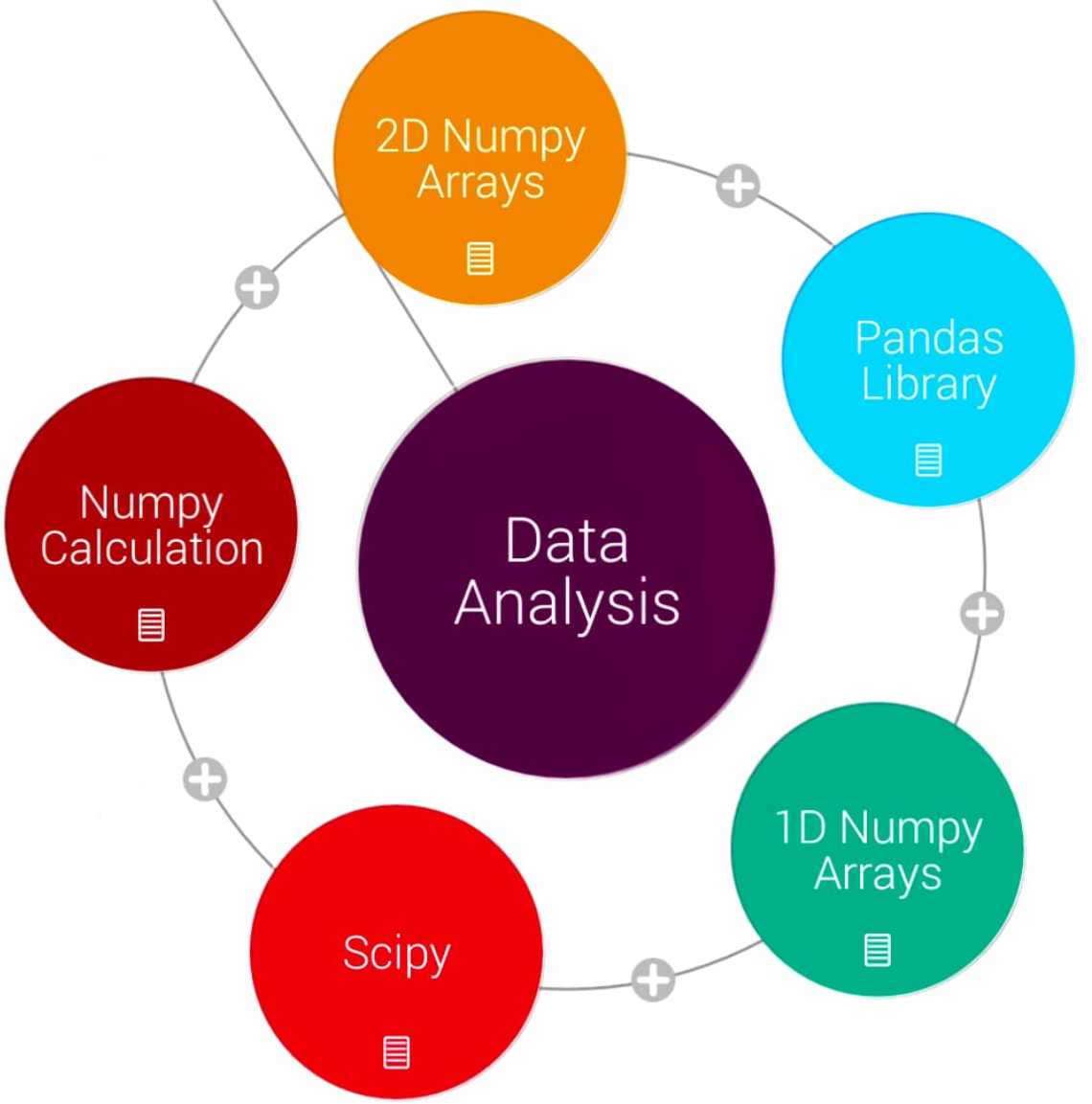
```
with open('file.json') as json_data:
    jsonfile = json.load(json_data)
```

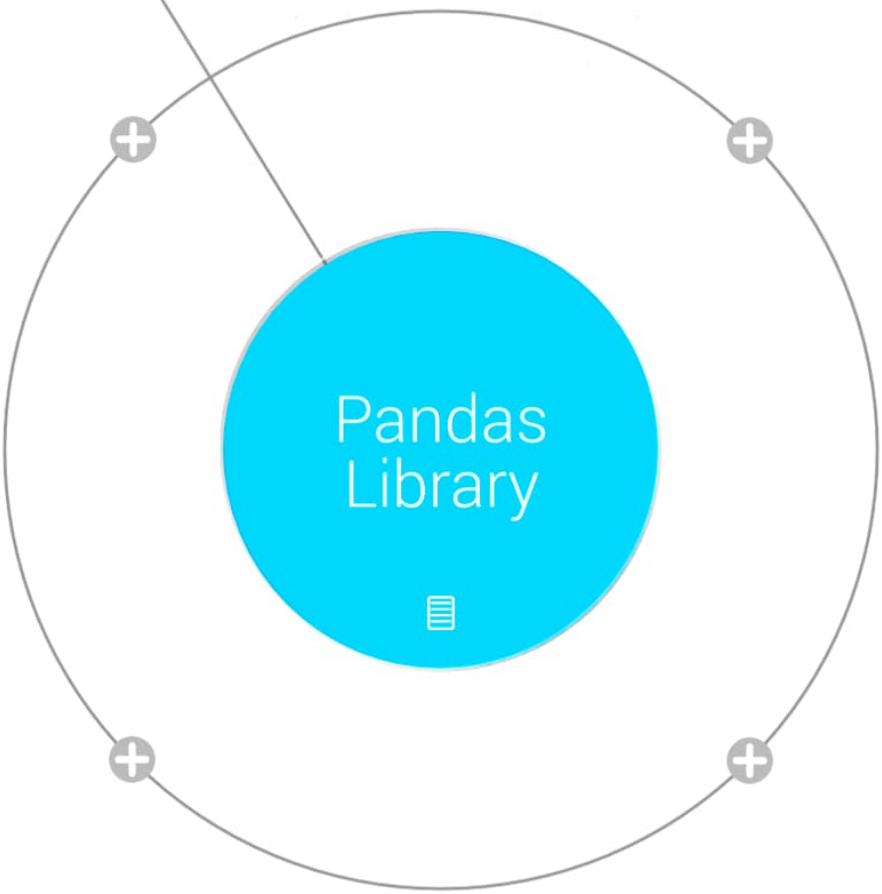
Accordingly slice the json file to convert it to csv version.

Unzipping a zipped file:

```
!wget -O 'filename.zip' url_of_file
!unzip -o -j filename.zip
```

Python Introduction




Pandas
Library

Pandas Library

1. df.head(x) : Know 1st x rows of dataframe.

2. df = pd.DataFrame(dictionary name)
   (Converts a dictionary to dataframe)

3. df.ix[Row name, Column name]
   (To print out particular row and column)

4. To slice out data:

   z = df.ix[Row range , Column range]
Ex: z = df.ix[0:2 , 0:3]

5. df['Column name']. unique()
   (Prints out unique values in dataframe)

6. df1 = df[df ['Released'] >= 1980]

(Stores value in df1 for values greater than/ equal to 1980 for column 'released')

7. df.to_csv(' path of file.csv ')
   (Saves the dataset to csv file)

8. df.shape -- (Size of dataframe)

Data Analysis

1D Numpy Arrays

Let's create a Numpy Array named a.
a = np.array([0, 1, 2, 3, 4, 5])
// Data type has to be same

1. a.dtype - (Data Type of elements in array)
2. a.size - (Size of Numpy array)
3. a.shape - (Dimensions of array)

4. a[2:4] = "200","450"
   (Changes value of index 2 and 3)

5. b = np.dot (a,c)
   (Dot product of array a and c)

6. b = a.mean() - Average of array elements.
7. b = a.max() - Max value of elements.

8. Linespacing:

   np.linspace (-2, 2, 5)
   (It inserts equal set of 5 no's btw.. -2 and 2)

9. Broadcasting:

   a = np.array([[1,2,3,4,5]])
   a+10

1D Numpy Arrays

4. a[2:4] = "200","450"
   (Changes value of index 2 and 3)

5. b = np.dot (a,c)
   (Dot product of array a and c)

6. b = a.mean() - Average of array elements.
7. b = a.max() - Max value of elements.

8. Linespacing:

   np.linspace (-2, 2, 5)
   (It inserts equal set of 5 no's btw.. -2 and 2)

9. Broadcasting:

   a = np.array([[1,2,3,4,5]])
   a+10

   Output: a = [11,12,13,14,15]

10. np.reshape(x,y) //where x are rows and y are columns.

Data Analysis

2D Numpy Arrays

Creating a 2D Numpy Array a :
a = np.array([[1,0,1],[2,3,2],[2,4,9]])

Creates a 3*3 matrix.

1. Indexing in 2D Arrays:
   a[0,0] : (Output: 1)

2. Slicing in 2D Arrays:
   a[0:2, 2] : (Output: vector 1,2)

3. Matrix Multiplication (Dot product):
   np.dot(a,b)

4. np.sqrt(array_name)
   np.log(array_name)

Finds square root and logarithms for each element in the Numpy array

5. np.arange(0,16)
   // Creats an array for continuing 0-16

6. arrayname.sum()
   arrayname.mean()
   arrayname.max()

2D Numpy Arrays

5. np.arange(0,16)
   // Creats an array for continuing 0-16

6. arrayname.sum()
   arrayname.mean()
   arrayname.max()
   arrayname.transpose()

arrayname.argmax() // returns the index of max value

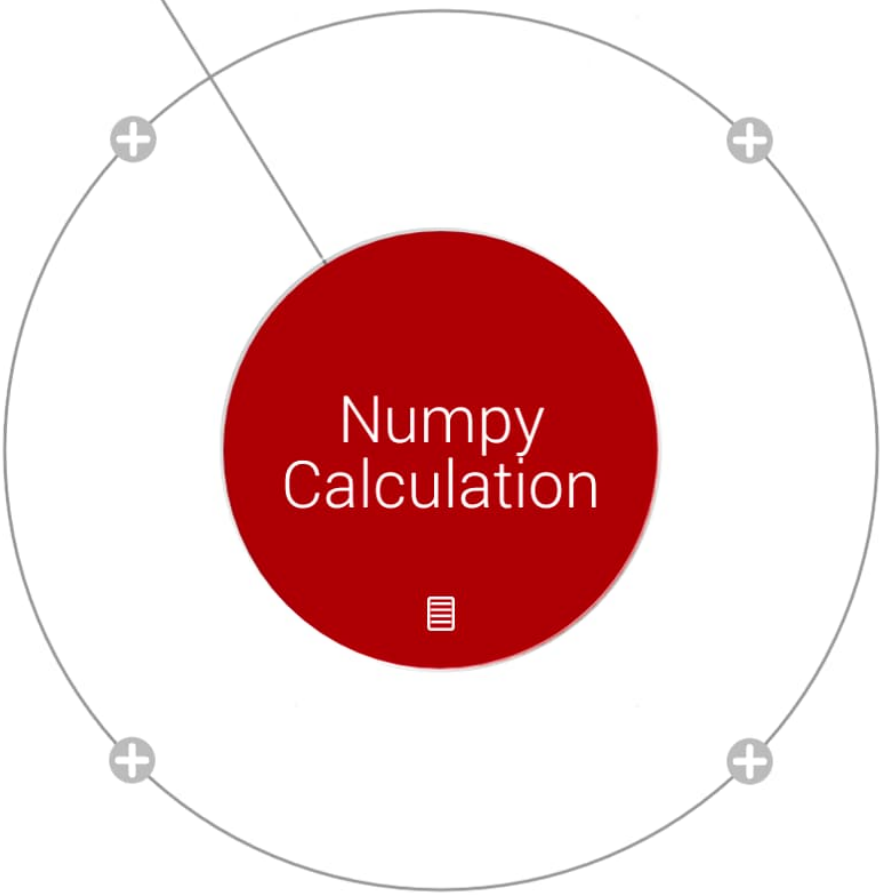
---

7. np.zeros((x,y))
// Creates a x * y shape array with just zeros

8. np.ones((x,y))
// Creates a x * y shape array with just ones

9. a = b.copy()
// Copies array b into a such that if values in b changes a remains unchanged.

Data Analysis



Numpy Calculation

1. a = np.array([7,8,9,9,7])*2
   [Output] : [14,16,18,18,14]

Can be used to square, divide etc..

2. sum(a) //Adds all the elements of a
3. a[a>16] // Prints all values > 16

Similar to this not equal, equal etc. can used.
Here if a[a>16] wasn't used and just [a>16] was used true/false would be printed.

4. np.logical_and(a>10, a<16)
   // Prints value more than 10 and less than 16

Similar to this or and not can be used.

---

5. np.linalg.inv(arrayname)
   // Inverse of given array (for square matrix)

6. np.trace(arrayname)
   // Prints addition of diagonal elements (L-R)

---

7. arrayname.ravel()

Similar to this not equal, equal etc. can used.
Here if a[a>16] wasn't used and just [a>16] was used true/false would be printed.

4. np.logical_and(a>10, a<16)
// Prints value more than 10 and less than 16

Similar to this or and not can be used.

---

5. np.linalg.inv(arrayname)
// Inverse of given array (for square matrix)

6. np.trace(arrayname)
// Prints addition of diagonal elements (L-R)

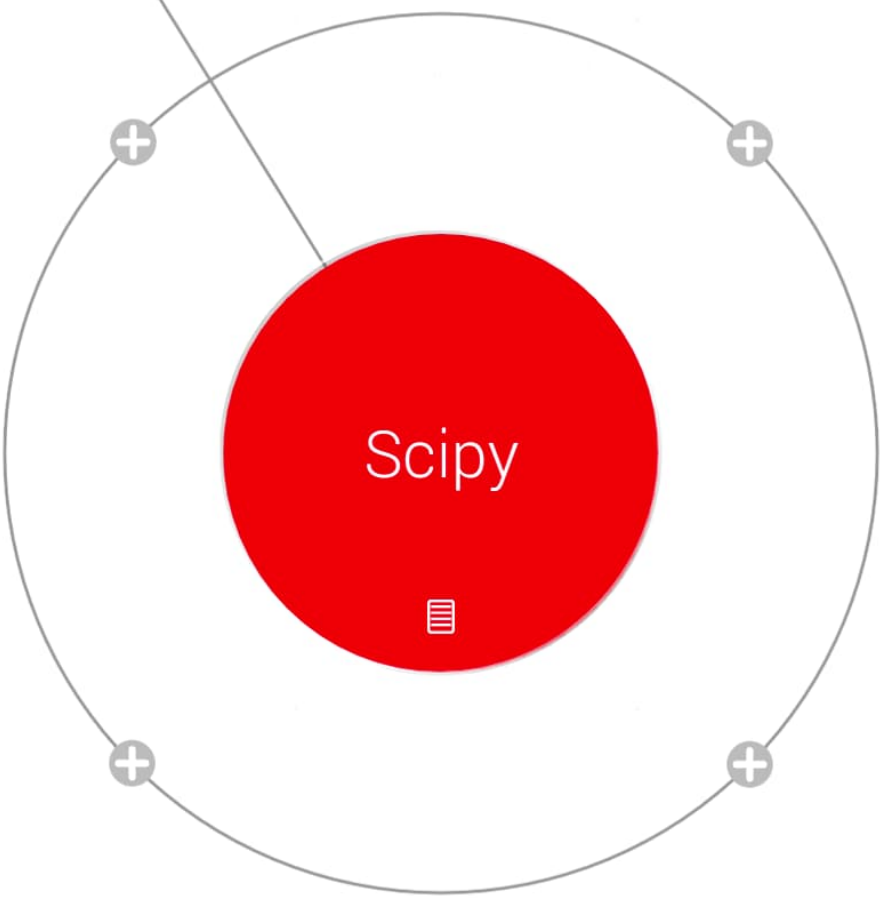
---

7. arrayname.ravel()
// Flattens any shape array to 1-D

8. np.hstack(array1, array2)
// Merges one array to end of other

9. np.hsplit(arrayname, 2)
// Splits an array to 2 equal parts

Data Analysis



Scipy

For optimization (Finding Min/Max of f(x)):

```
from scipy import optimize
def f(x):
   return x**2 + 5*np.sin(x)
minvalue = optimize.minimize(f, x0=2,
method='bfgs')
```

// This finds the value of x for which $x^2 + 5\sin x$ gives a minimum value.

For Integration:

```
from scipy.integrate import quad
def integrateFunction(x,a,b):
    return a*x + b
a, b = 5,6
quad(integrateFunction, 0, 1, args=(a,b))
```

// Here 0 is lower and 1 is upper limit. This integrates the function 5x + 6.

Linear Algebra operations:

Scipy

Linear Algebra operations:

1. from scipy import linalg
linalg.det(arrayname)
// Finds determinant of an array.

2. eigenvalues, eigenfunction =
linalg.eig(arrayname)

// To find Eigenvalue, Eigenfunction of array

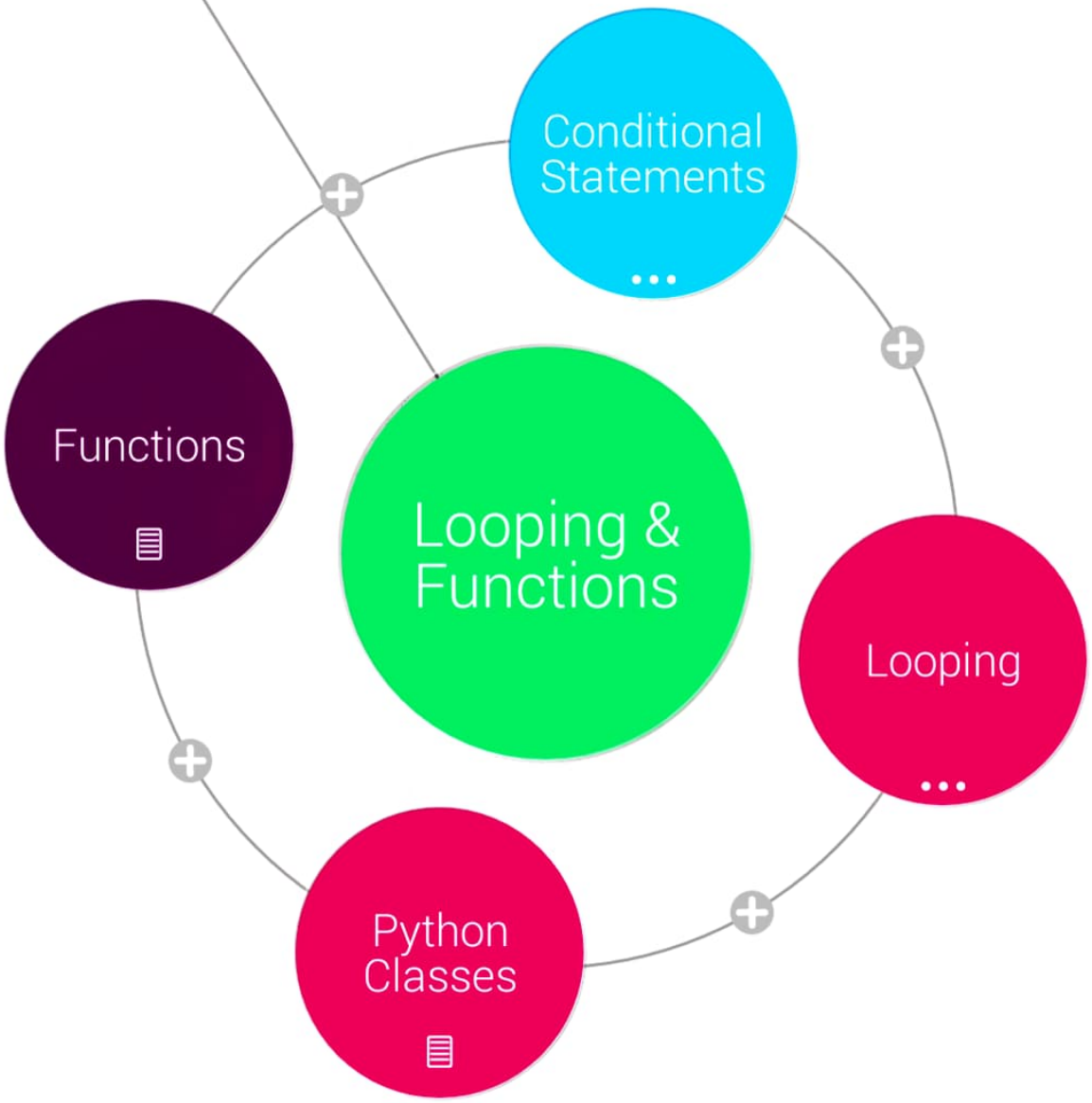
3. Solving Linear Equation
coef = np.array([[2,3,1],[1,5,7],[3,2,9]])
value = np.array([21,9,6])

// We write a coeffecient and value matrix

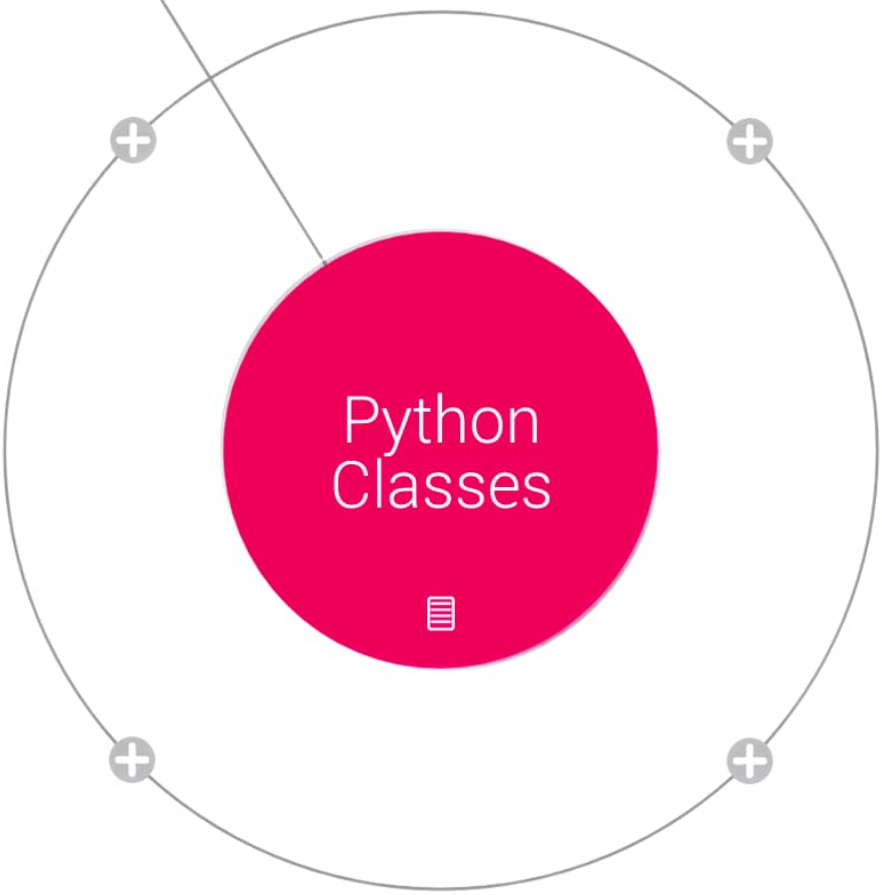
linalg.solve(coef, value)
// It outputs value for x,y,z

Equations are:
$2x + 3y + z = 21$, $x + 5y + 4z = 9$, $3x + 2y + 9z = 6$

Python Introduction



Looping &
Functions



Python Classes

Let's draw a circle using class in Python:

```
class circle (object):
  def __init__(self, radius, color):
      self.radius = radius
      self.color = color

def drawcircle (self):
      plt.gca().add_patch(plt.circle((0,0),
self.radius, fc=self.color))
      plt.show()
```

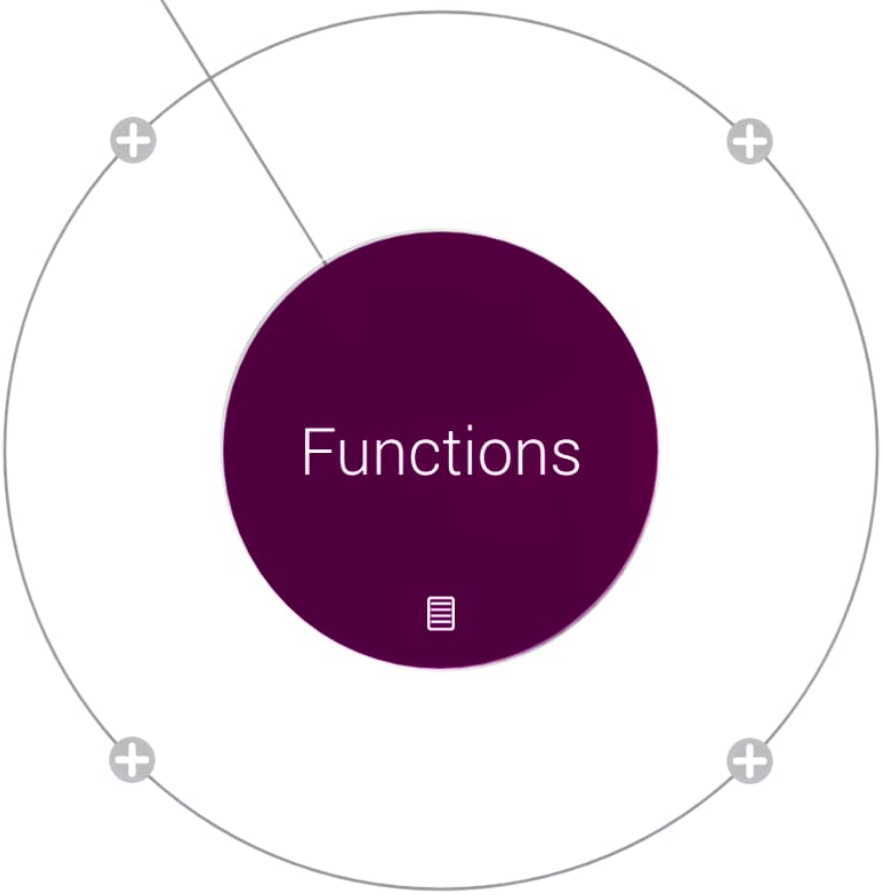
When we type:

```
C1 = circle ('10', 'red')
C1.drawcircle()
```

Changing color or radius of Circle C1:

```
C1.radius = '20'
```

Looping & Functions



Functions

1. Function Declaration: (Adds two numbers)

```
def add (a, b):
    c = a + b
    print (a, "plus", b, "equals", c)
```

add (3,4) -- (Function Calling)
Output (3 plus 4 equals 7)

2. Using Loops inside Functions:

```
def album(x):
     for i,s in enumerate (x):
        print("Album", "i", "Rating is", "s")
```

Now suppose we define a list:
Rate = [10,20,30]

album(Rate)
Output:

Album 0 Rating is 10
Album 1 Rating is 20
Album 2 Rating is 30

// Function can contain a return statement.

Functions

add (3,4) -- (Function Calling)
Output (3 plus 4 equals 7)

2. Using Loops inside Functions:

```
def album(x):
    for i,s in enumerate (x):
        print("Album", "i", "Rating is", "s")
```

Now suppose we define a list:
Rate = [10,20,30]

album(Rate)
Output:

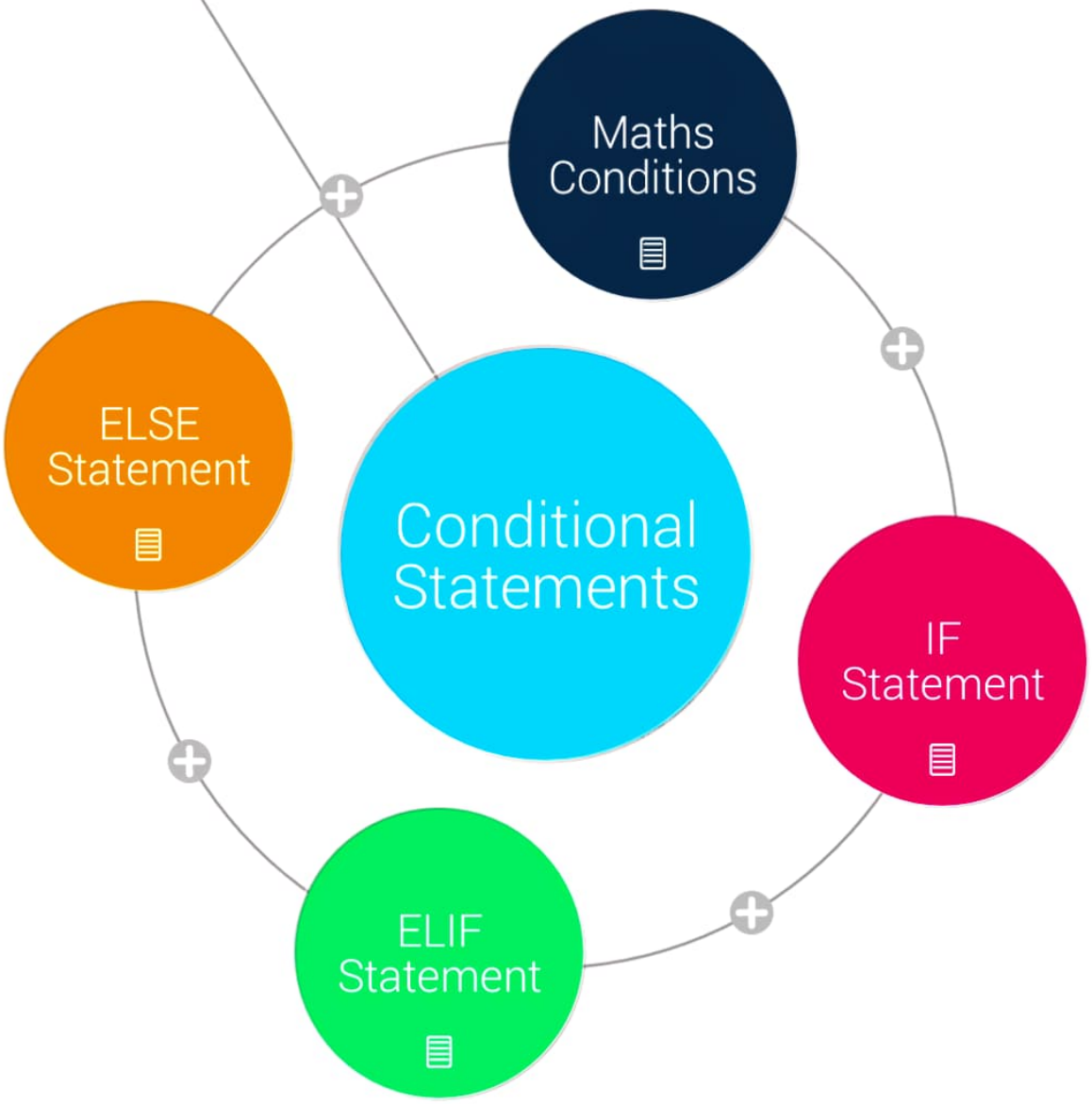
Album 0 Rating is 10
Album 1 Rating is 20
Album 2 Rating is 30

// Function can contain a return statement.

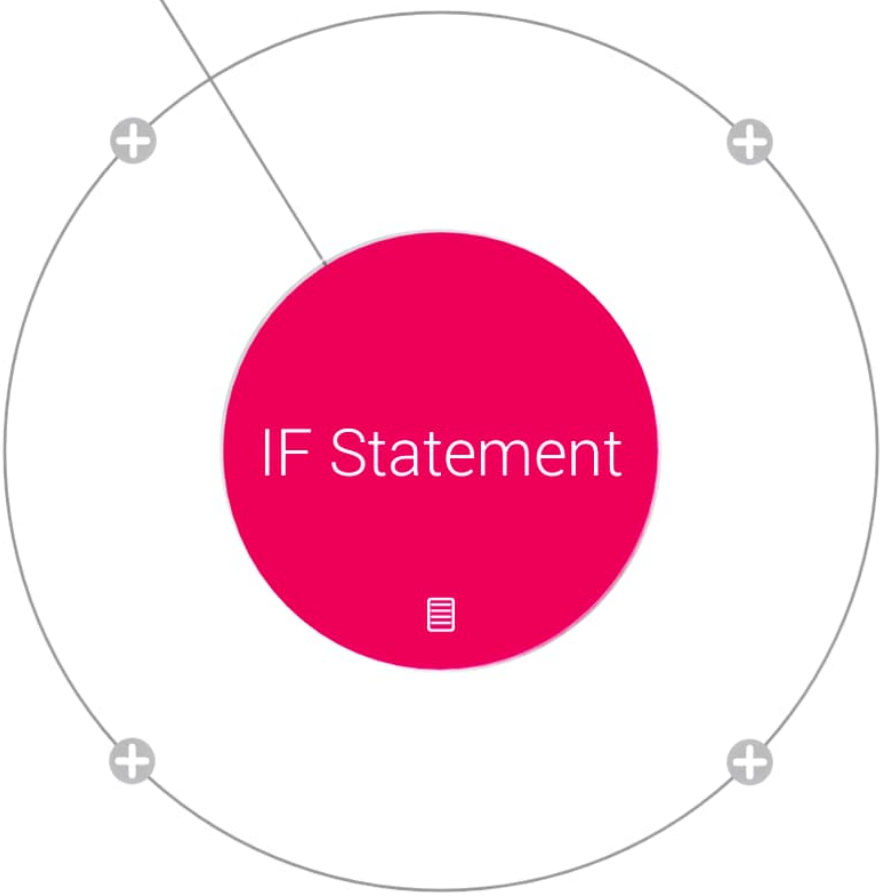
---

1. sorted() //arranges in ascending order
2. reverse() // arrange in descending order
3. zip() // merges two variables

Looping & Functions



Conditional Statements



IF Statement

If statement checks a condition and if it's true, proceeds towards the next statement. If the value of the condition is false it moves forward the loop. Example:

```
if (age > 18):
   print ("You can enter")
print ("move on")
```

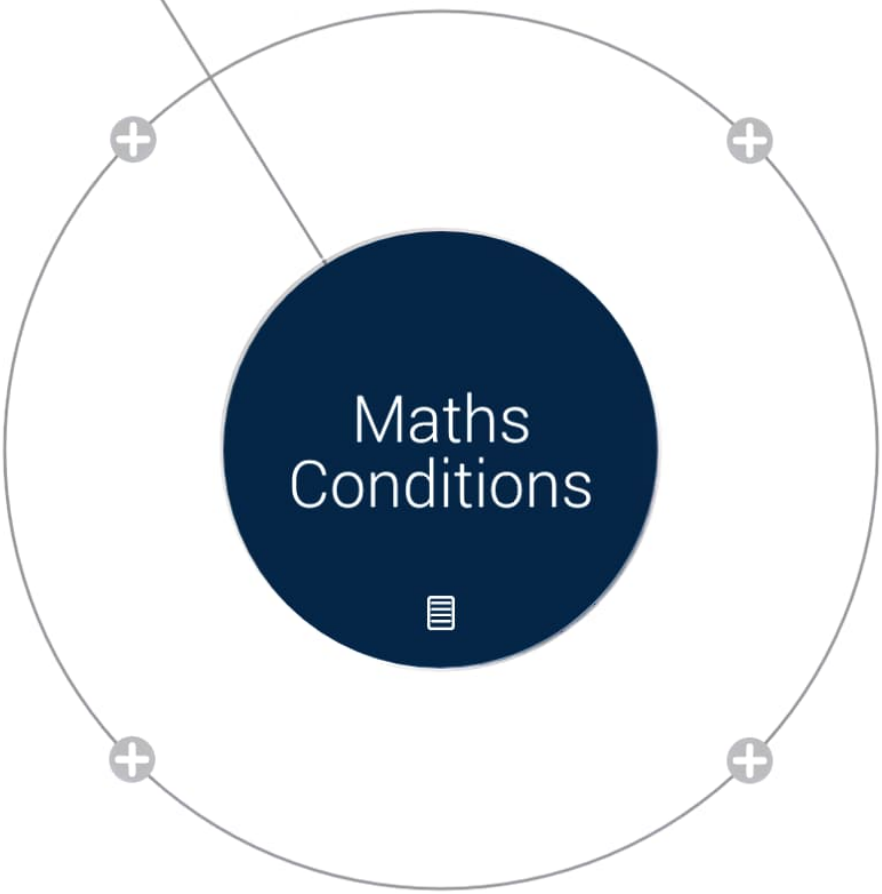
If the age is above 18, You can enter is printed. And if not, then "move on" is printed.

Logic Operations in if statement:

```
if (condition 1) or (condition 2):
    print ("x")
```

If either or both conditions are true, X gets printed. Similarly and operator works. (Logic Gate concept)

Conditional Statements



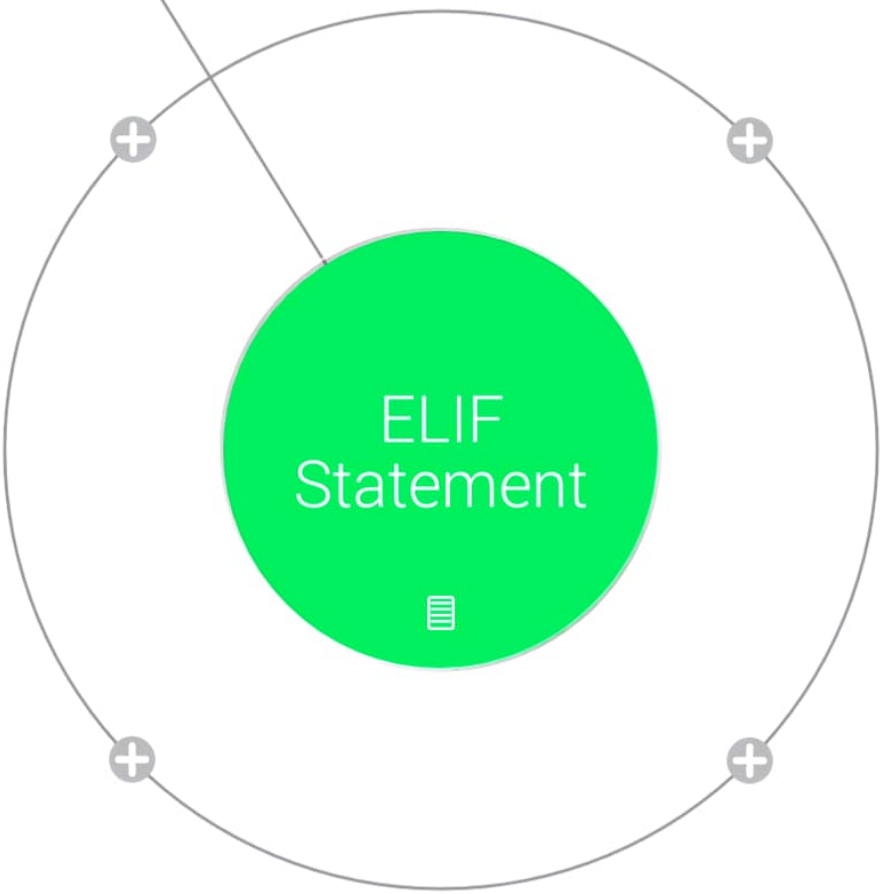
Maths Conditions

1. Equality : A == B
2. Greater : A > B
3. Smaller : A < B

4. Not Equal : A != B
5. And operator : &
6. Or operator : |

Apart the symbols, just by writing and, or, not would also solve the task.

Conditional Statements



ELIF Statement

Same as IF and Else combination but checks for two condition. Example:

```
if (age > 18):
    print ("You can enter")

elif (age == 18):
    print ("Wait")

else:
    print ("Go")
```

Conditional Statements

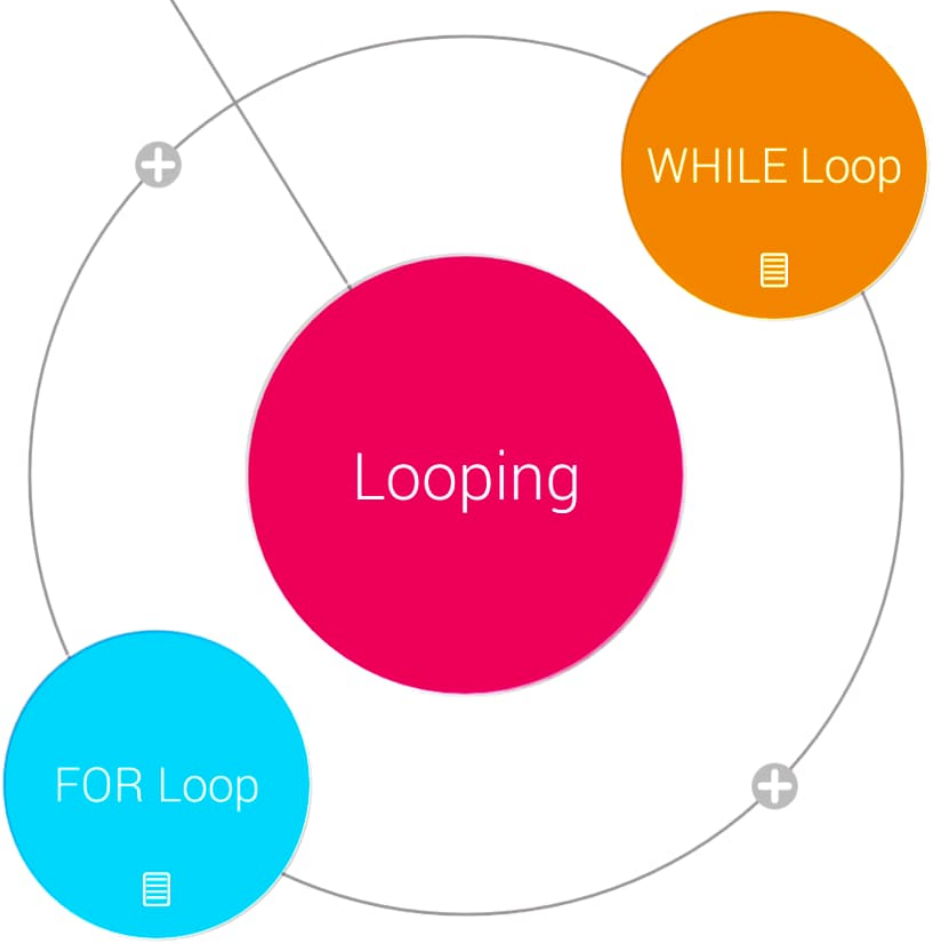
ELSE Statement

Similar to IF statement. If the condition of the IF statement is found false, this statement is printed. Example:

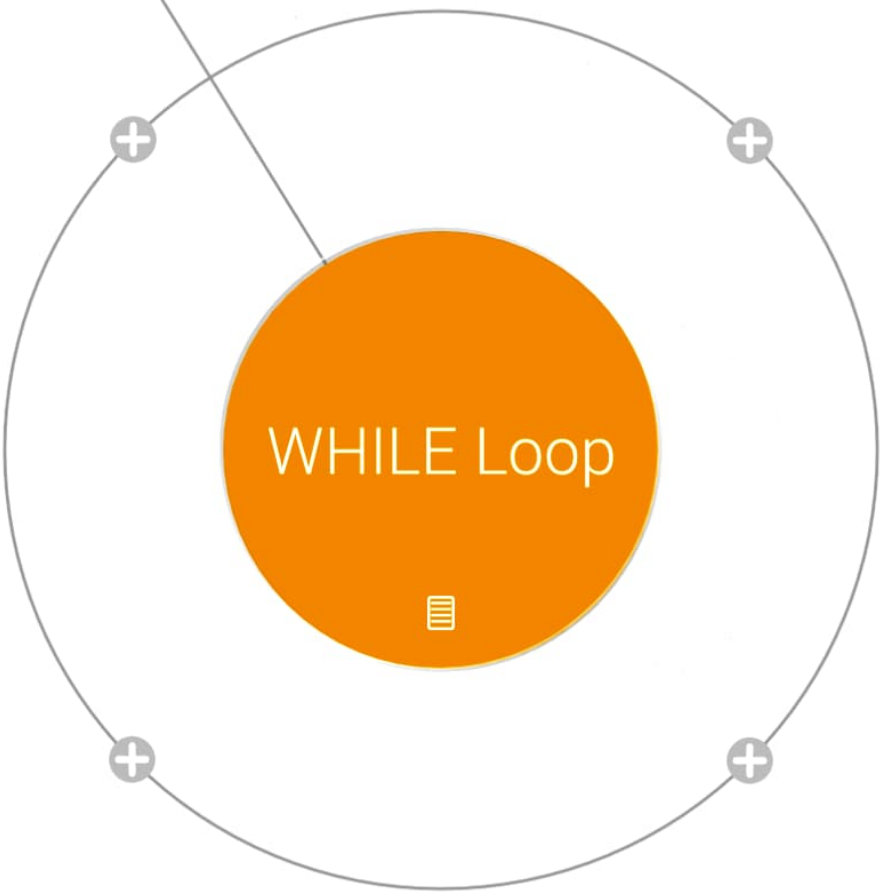
```
if (age > 18):
    print ("You can enter")

else:
    print ("Go")
```

If the age is above 18, " You can enter is printed and if not then "Go" is printed.

Looping & Functions



Looping



WHILE Loop

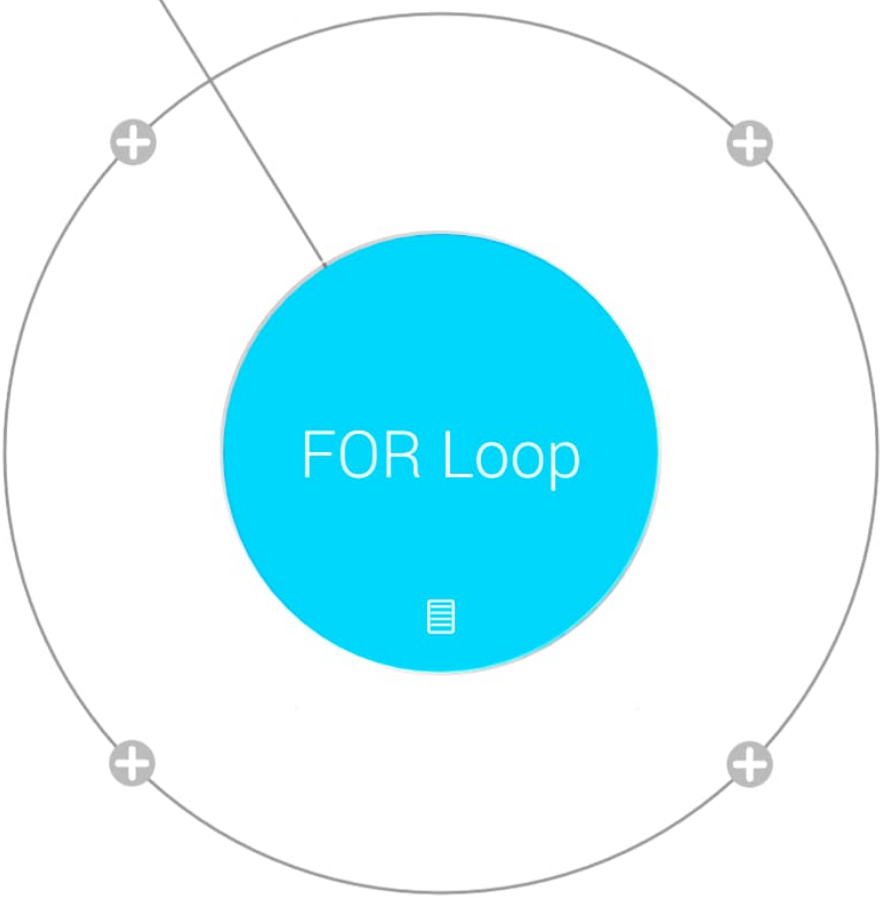
While Loop runs only till the condition is met.
Example:

```
squares1 = ['a', 'a', 'a', 'b', 'a']
squares2 = [ ]
i = 0

while (square1[i] == 'a'):
    square2.append(sqaure1[i])
    i = i + 1
```

The output for square2 is:

```
['a', 'a', 'a']
```

Looping



FOR Loop

Types of for statement:

1. for x in range (0,3):
        print(x)
   (Output: 0,1,2)

2. for x in ['A', 'B', 'C']:
        print (x + 'A')
   (Output: AA, BA, CA)

3. for i,x in enumerate (['A', 'B', 'C']):
        print (i,x)

   (Output) 0 A
            1 B
            2 C

The above command assigns index to values.