Python packages and Numpy

Contents

- Python package
- Numpy package fundamentals

Image source - https://pixabay.com/photos/chef-smell-cook-spice-eat-939437/

_VOIS

Image source - https://www.sharmispassions.com/sambar-powder-recipe-brahmin-style/

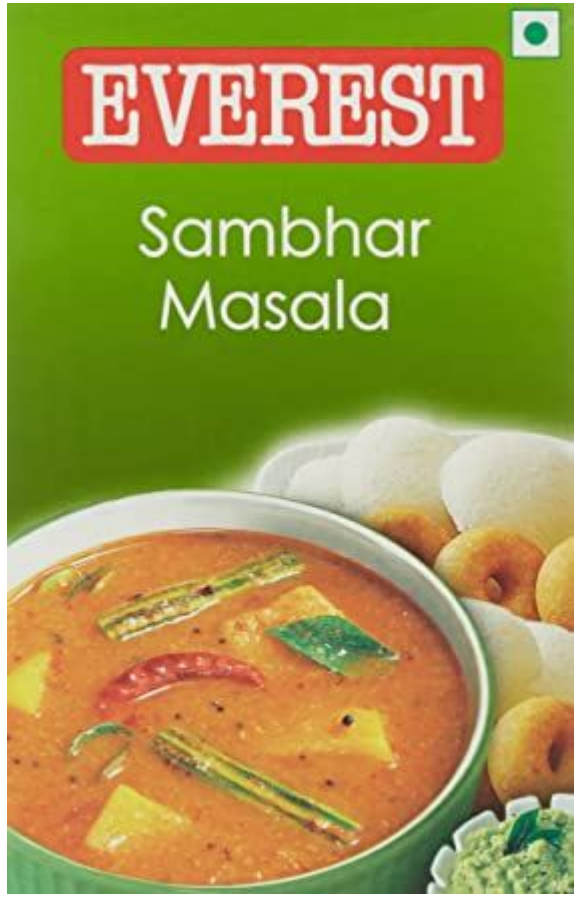


Image source - https://www.amazon.com/Everest-Sambhar-Masala-100-Grams/dp/B0152RP1WM

Python packages

- Functions and methods are very powerful

  Adding functions and methods in python codes will become mess, lot of codes will never used from methods and functions.

- Packages:
  - Directory of python scripts
  - Each script = module
  - Specify functions, methods and types
  - Thousand of packages available:
    - Numpy
    - Matplotlib
    - Pandas
    - Scikit learn (these packages need to be installed)

Package installation

- Download - http://pip.readthedocs.io/en/stable/installing/

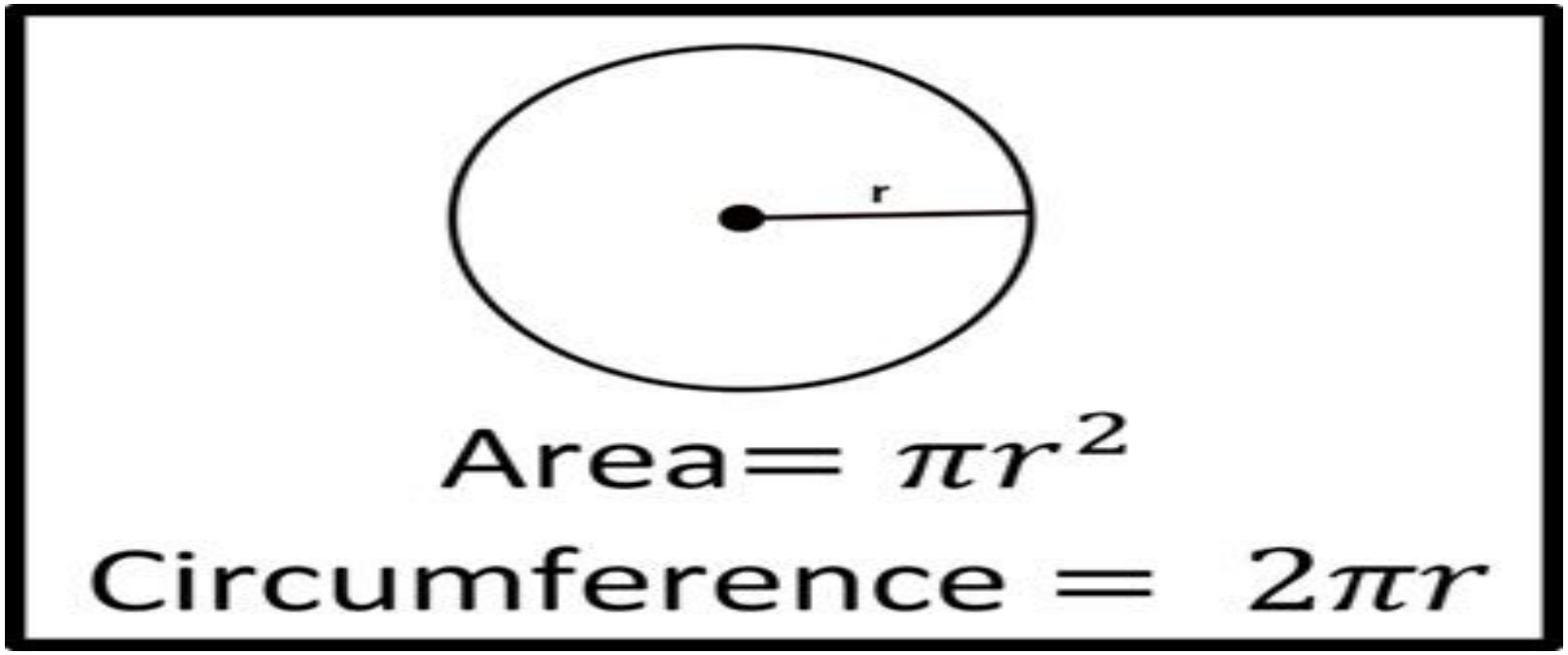


Image source - http://andymath.com/area-and-circumference-circle-2/

Import package

- For a fancy clustering algorithm, you want to find the circumference C and area A of a circle. When the radius of the circle is r, you can calculate C and A as:
- C=2πr
- A=πr2
- To use the constant pi, you'll need the math package. A variable r is already coded in the script. Fill in the code to calculate C and A and see how the print() functions create some nice printouts.
- Import the math package. Now you can access the constant pi with math.pi.
- Calculate the circumference of the circle and store it in C.
- Calculate the area of the circle and store it in A

Working math package

```
# Definition of radius
    r = 0.43
# Import the math package
    import math
# Calculate C
    C = 2*math.pi*r
# Calculate A
    A = math.pi*r**2
```

_VOIS

Selective import

- General imports, like import math, make all functionality from the math package available to you. However, if you decide to only use a specific part of a package, you can always make your import more selective:

```
from math import pi
```

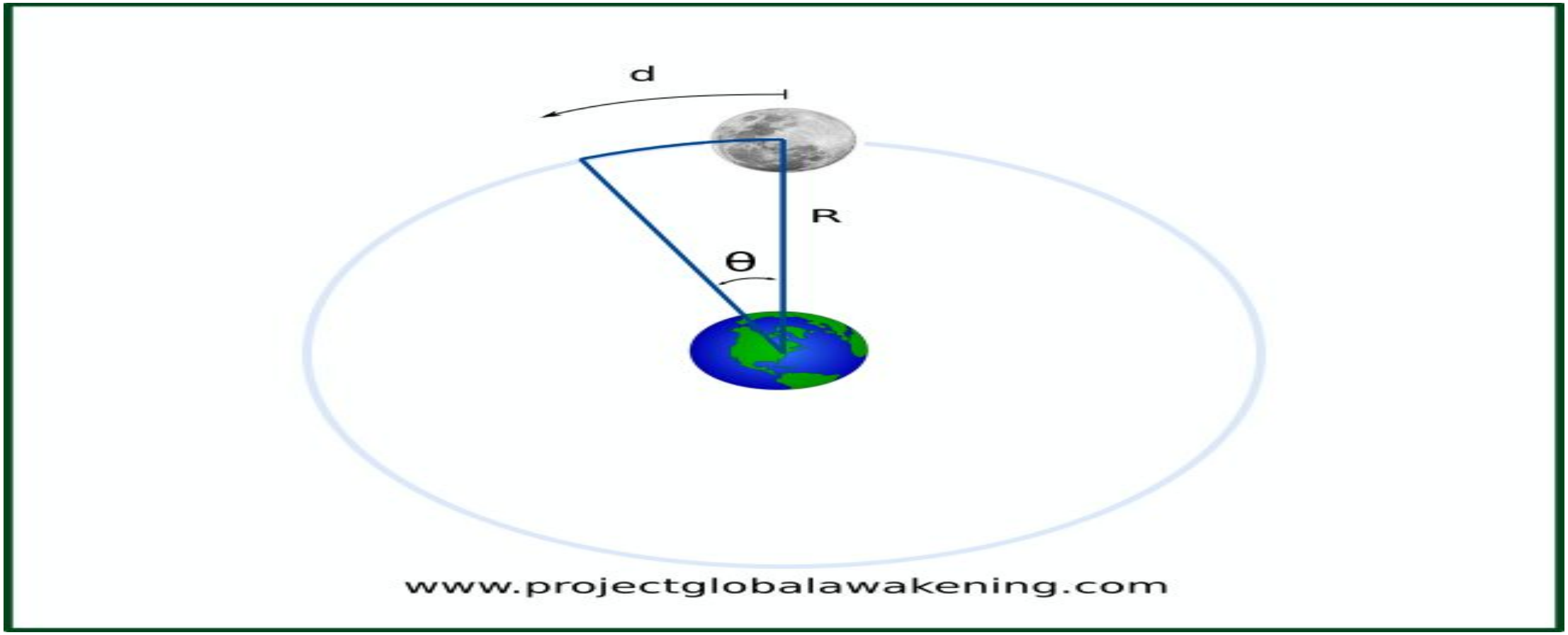


Image source - http://www.projectglobalawakening.com/daily-motion-moon/

Let's say the Moon's orbit around planet Earth is a perfect circle, with a radius r (in km) that is defined in the script.

```
#Perform a selective import from the math

#package where you only import the radians function.

#Calculate the distance travelled by the Moon over 12 degrees of its orbit.

Assign the result to dist. You can calculate this as r*ϕ, where r is the
radius and ϕ is the angle in radians.

Consider value of r as 3,84,400 kms.
```

Hint

To convert an angle in degrees to an angle in radians, use the radians() function. Print out dist.

Calculate distance travelled by moon

```
# Definition of radius
    r = 384400
# Import radians function of math package
    from math import radians
# Travel distance of Moon if 12 degrees. Store in dist.
    dist=r*radians(12)
# Print out dist
    print(dist)
```

Arrays

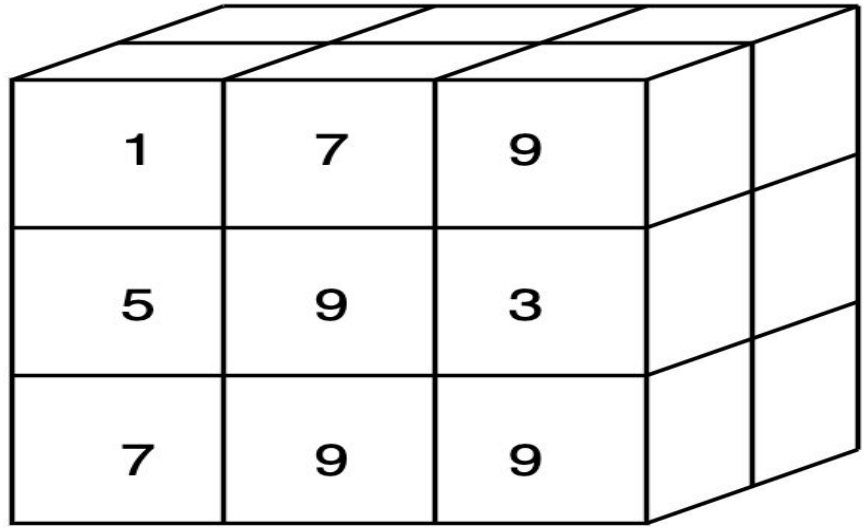
1D Array

3	2

2D Array

1	0	1
3	4	1

3D Array



Arrays Vs list

- 1.Why array ?
- 2.Why elements of arrays needs to be of same data type?
- 3.What is the way to include different data type elements in array? Is it possible in python? What alternative python holds?
- **4.How to work with large arrays???**

Source - https://phys.org/news/2015-07-kids-maths-figure.html

Source - https://www.oxfordlearning.com/what-is-math-anxiety/


python™


NumPy

NumPy array at structure level

- combination of a memory address, a data type, a shape, and strides
- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

e.g. **import numpy as np**

a = np.array([[1,2,3],[4,5,6]])

a.strides

(24,8)

Create numpy array

```
# Create list baseball
baseball = [180, 215, 210, 210, 188, 176, 209, 200]
# Import the numpy package as np
import numpy as np
# Create a Numpy array from baseball: np_baseball
np_baseball = np.array(baseball)
# Print out type of np_baseball
print(np_baseball)
```

Calculate bmi of players

- You are a huge baseball fan. You decide to call the MLB (Major League Baseball) and ask around for some more statistics on the height of the main players. They pass along data on more than a thousand players, which is stored as a regular Python list: height. The height is expressed in inches. Can you make a Numpy array out of it and convert the units to centimeters?

BMI

```
# height and weight are available as a regular lists # Import numpy
import numpy as np
# Create array from height with correct units: np_height_m
np_height_m = np.array(height) * 0.0254
# Create array from weight with correct units: np_weight_kg
np_weight_kg=np.array(weight)*0.453592
# Calculate the BMI: bmi
bmi=np_weight_kg/np_height_m**2
# Print out bmi
print(bmi)
```

Lightweight baseball players

```
#To subset both regular Python lists and Numpy arrays, you can use
square brackets:
    x = [4 , 9 , 6, 3, 1]; x[1]
    import numpy as np
    y = np.array(x); y[1]
#For Numpy specifically, you can also use boolean Numpy arrays:
    high = y > 5 ;y[high]
#The code that calculates the BMI of all baseball players is
already included.
#Follow the instructions and reveal interesting things from the
data!
```

```
# height and weight are available as a regular lists # Import numpy
import numpy as np
# Calculate the BMI: bmi
np_height_m = np.array(height) * 0.0254
np_weight_kg = np.array(weight) * 0.453592
bmi = np_weight_kg / np_height_m ** 2
# Create the light array
light=bmi < 21
# Print out light
print(light)
# Print out BMIs of all baseball players whose BMI is below 21
print(bmi[bmi<21])
```

Subsetting Numpy Arrays

subsetting (using the square bracket notation on lists or arrays) works exactly the same.

```
x = ["a", "b", "c"]
x[1]
np_x = np.array(x)
np_x[1]
```

Example

```
# height and weight are available as a regular lists# Import numpy
import numpy as np
# Store weight and height lists as numpy arrays
np_weight = np.array(weight)
np_height = np.array(height)
# Print out the weight at index 5
print(np_weight[5])
#Print out sub-array of np_height: index 2 up to and including
index 5
print(np_height[2:6])
```

2D Numpy Arrays

```
# Create baseball, a list of lists
baseball = [[180, 78.4], [215, 102.7], [210, 98.5], [188, 75.2]]
# Import numpy
import numpy as np
# Create a 2D Numpy array from baseball: np_baseball
np_baseball=np.array(baseball)
# Print out the type of np_baseball
print(type(np_baseball))
# Print out the shape of np_baseball
print(np_baseball.shape)
```

Baseball data in 2D form

```
# baseball is available as a regular list of lists

# Import numpy package

import numpy as np

# Create a 2D Numpy array from baseball: np_baseball
np_baseball=np.array(baseball)

# Print out the shape of np_baseball

print(np_baseball.shape)
```

Subsetting 2D Numpy Arrays

If your 2D Numpy array has a regular structure, i.e. each row and column has a fixed number of values, complicated ways of subsetting become very easy. Have a look at the code below where the elements "a" and "c" are extracted from a list of lists.

```
# regular list of lists
x = [["a", "b"], ["c", "d"]]
[x[0][0], x[1][0]]
# numpy
import numpy as np
np_x = np.array(x)
np_x[:,0]
```

```
array([[ 1.73, 1.68, 1.71],   0
       [ 65.4 , 59.2, 63.6]])  1
          0      1     2
```

Sub-setting 2D arrays

```
# baseball is available as a regular list of lists # Import numpy package
import numpy as np
# Create np_baseball (2 cols)
np_baseball = np.array(baseball)
# Print out the 5th row of np_baseball
print(np_baseball[5,:])
# Select the entire second column of np_baseball: np_weight
np_weight = np_baseball[:,1]
# Print out height of 5th player
print(np_baseball[5,0])
```

2D Arithmetic

Numpy was able to perform all calculations element-wise. For 2D Numpy arrays this isn't any different! You can combine matrices with single numbers, with vectors, and with other matrices. Execute the code below:

```
import numpy as np
np_mat = np.array([[1, 2], [3, 4], [5, 6]])
np_mat * 2
np_mat + np.array([10, 10])
np_mat + np_mat
```

References -

1. Zed A Shaw , *Learn Python 3 the hard way* , Addison Wesley
2. Erric Matthes , *Python Crash Course* , No starch press
3. Wes McKinney , *Python for data analysis* , O'Reilly Media, Inc.


thank you