Python for Engineers

Tassadaq Hussain
Associate Professor Riphah International University

Collaborations:
Microsoft Research and Barcelona Supercomputing Center
Barcelona, Spain
European Network on High Performance and Embedded Architecture and Compilation
UCERD Pvt Ltd Islamabad

Agenda

- Importance of Programming Languages

- **Python Language**

- Python for Engineers

  - Interfacing with external world (etc)

  - RasppberyPi

- Python for Scientists

  - Data Analytic

Scripting Languages

- Scripting languages focus <u>flexibility</u>, <u>rapid development</u> and <u>dynamic checking.</u>

- Their type systems embrace very high level concepts such as tables, patterns, lists and files.

- There a number of distinct groups that fall under the scripting language family.

- Languages such as Perl and Python are known as ``glue'' languages because they were designed to glue existing programs together.

- There are other extensible types of scripting languages used in the WWW also.

UCERD
Gathering
Intellectuals
www.ucerd.com

Python Language

Everything in Python is an object.

The objects can be either mutable or immutable.

A mutable object can be changed after it is created, and an immutable object can't.

Strengths

- Easy to learn:
- Supports multiple programming paradigms
- Extensible
- Active open source community
- Large and Active Community Support
- Powerful Set of Packages
- Easy and Rapid Prototyping
- Easy to Collaborate

Python for Different Technologies
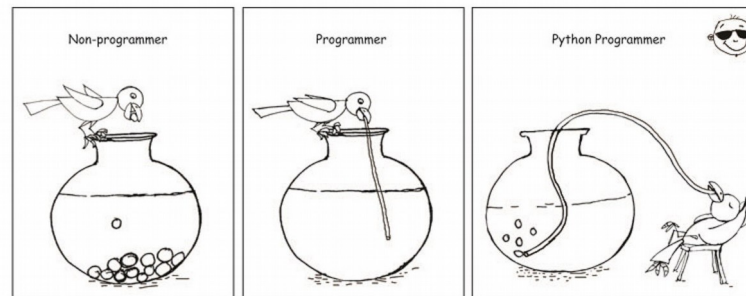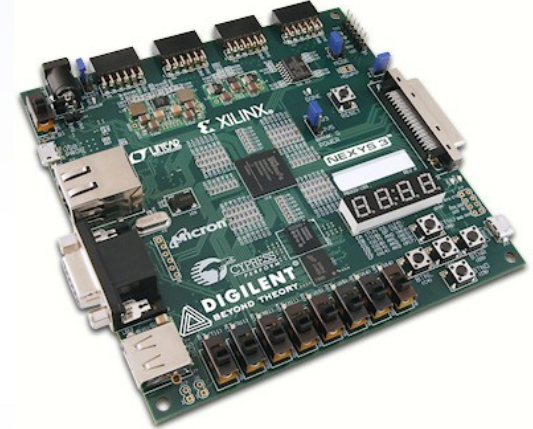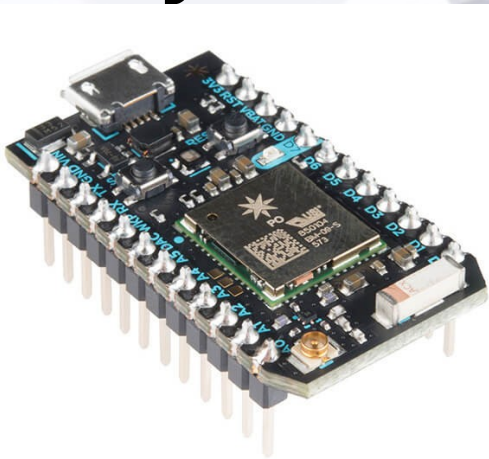
You have seen it all, Now, Choice is yours :-)

Non-programmer

Programmer

Python Programmer

Setting Up a Python Environment

- Set Up Anaconda Python Environment

- Installing Libraries
    - pip install required_package

-

Python for HPC

- Python is known for its very expressive language, easy to read syntax, large community, and impressive range of extension modules.

- Python has regularly come to be used as a program glue due to its ability to interface so easily with external applications.

- With an increase in the availability of Python tools for HPC comes a decrease in the performance barrier between Python and its complied foes.

- There are the more mature projects such as NumPy and SciPy, which offer performance without compilation, as well as recent attempts to bring parallel libraries and just-in-time compiling to Python.

- Numba is a python module that produces compiled code from python functions that can lead to significant speed-ups. Numba is also able to compile code for both Nvidia and AMD GPUs, which presents an exciting new HPC approach consisting of rapid development times and fast execution on desktops.

- **PyCOMPSs** is a framework which aims to ease the development and execution of Python parallel applications for distributed infrastructures, such as Clusters and Clouds.

Development Environments

1. PyDev with Eclipse
2. Komodo
3. Emacs
4. Vim
5. TextMate
6. Gedit
7. Idle
8. PIDA (Linux)(VIM Based)
9. NotePad++ (Windows)
10. BlueFish (Linux)
11. ipython

Python Keywords

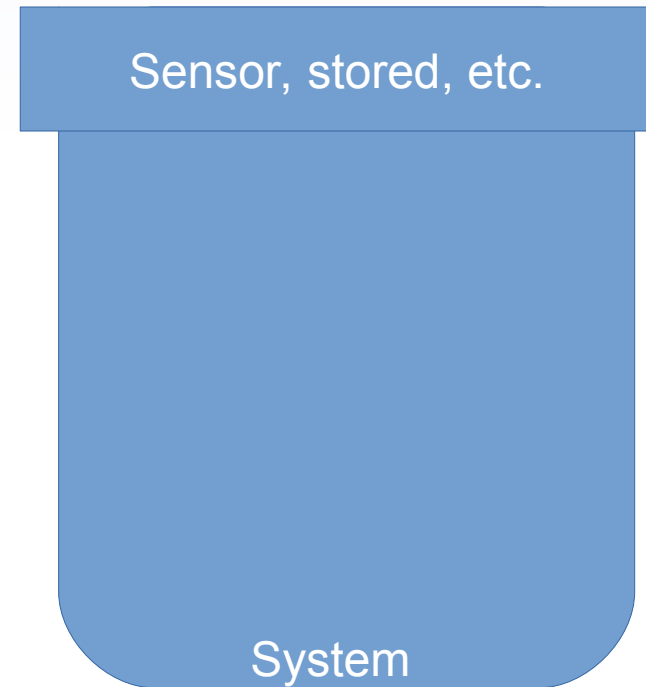|				
FALSE	Class	Finally	Is	return
None	Continue	For	Lambda	try
TRUE	Def	From	nonlocal	while
And	Del	Global	Not	with
As	Elif	If	Or	yield
Assert	Else	Import	Pass	
Break	Except	In	Raise	

Programming in Python

- Import Libraries
  - Libraries tool-kits frameworks
- Data I/O Sources
  - Source of Data, Disk, External Interface etc.
- Data Types
  - List, Tuple etc.
- Instructions
  - Conditional Statements
  - Repetition Statements
  - Functions
  - Training, Testing
- Results
  - Plots, Visualization,
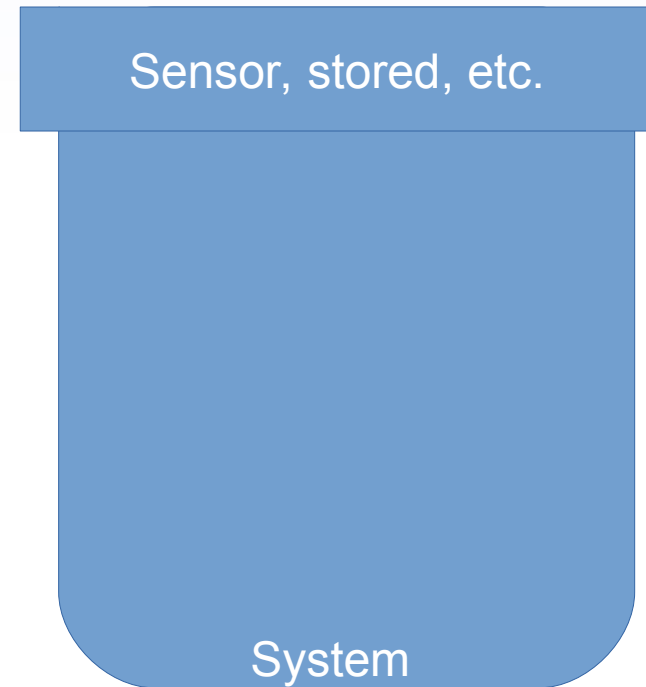
System
Method
Algorithm

.

.

Python Scripting Language

- Libraries

- Data Input Output

- Data Types

- Conditional Statements

- Repetition Statements
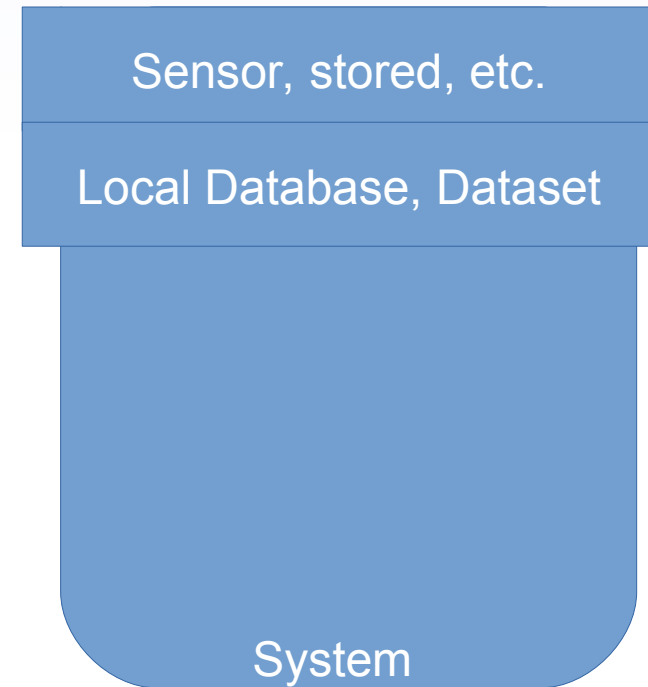
- Functions and Libraries

Sensor, stored, etc.

System

Python Scripting Language

- Data Input Output
- Data Types
- Conditional Statements
- Repetition Statements
- Functions and Libraries

Sensor, stored, etc.

System

UCERD
Gathering
Intellectuals
www.ucerd.com

Python Scripting Language

- Data Input Output

- Data Types

- Conditional Statements

- Repetition Statements

- Functions and Libraries

Sensor, stored, etc.

Local Database, Dataset
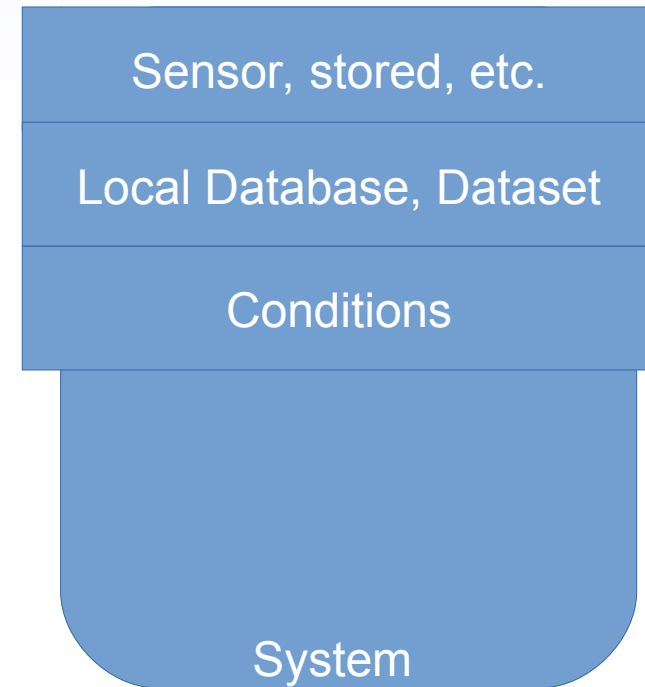
System

UCERD
Gathering
Intellectuals
www.ucerd.com

Python Scripting Language

- Data Input Output

- Data Types

- Conditional Statements

- Repetition Statements

- Functions and Libraries

Sensor, stored, etc.

Local Database, Dataset

Conditions

System

Python Scripting Language

- Data Input Output

- Data Types

- Conditional Statements

- Repetition Statements

- Functions and Libraries

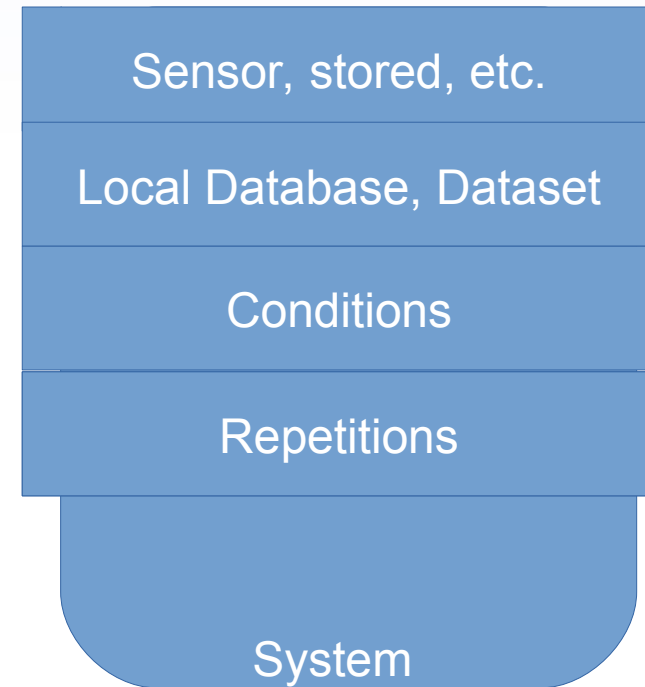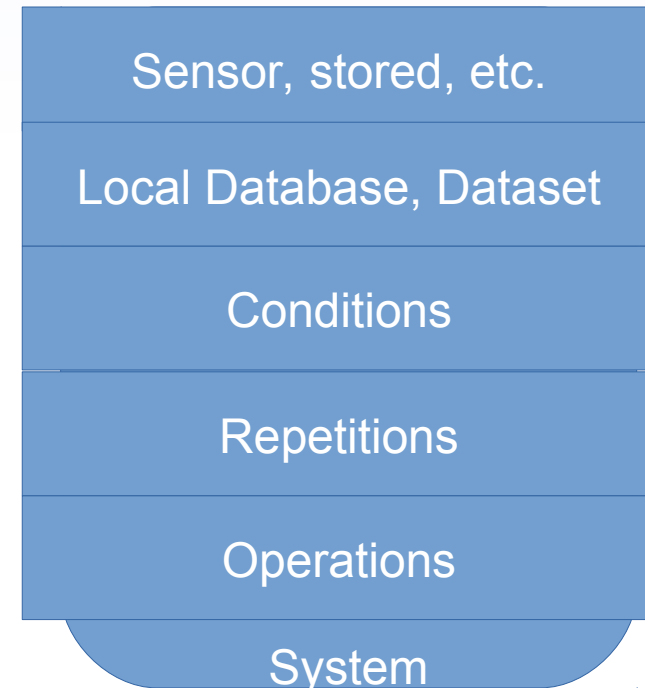| Sensor, stored, etc. |
| Local Database, Dataset |
| Conditions |
| Repetitions |
| System |

Python Scripting Language

- Data Input Output

- Data Types

- Conditional Statements

- Repetition Statements

- Functions and Libraries

| Sensor, stored, etc. |
| Local Database, Dataset |
| Conditions |
| Repetitions |
| Operations |
| System |

Python Environment

Libraries

Read data

Operations: Filtering, Processing, Classification, Control etc.

Visualizing

Write, Operate etc

UCERD
Gathering
Intellectuals
www.ucerd.com

Numpy

- NumPy, short for Numerical Python, is the foundational package for scientific computing in Python.

- Numpy is the backbone of Machine Learning in Python. It is one of the most important libraries in Python for numerical computations. It adds support to core Python for multi-dimensional arrays (and matrices) and fast vectorized operations on these arrays.

Numpy

- A fast and efficient **multidimensional array** object ndarray

- **Functions** for performing element-wise computations with arrays or mathematical operations between arrays

- Tools for **reading and writing array-based** data sets to disk

- **Linear algebra operations, Fourier transform, and random number generation**

- Tools for integrating **connecting** C, C++, and Fortran code to Python

Module: Numpy

NumPy is the fundamental package for scientific computing with Python.

It contains among other things:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

NumPy is a Python C extension library for array-oriented computing
- Efficient
- In-memory
- Contiguous (or Strided)
- Homogeneous (but types can be algebraic)

UCERD
Gathering
Intellectuals
www.ucerd.com

Source: http://www.numpy.org/

NumPy Functions

- comparison: `<, <=, ==, !=, >=, >`

- arithmetic: `+, -, *, /, reciprocal, square`

- exponential: `exp, expm1, exp2, log, log10, log1p, log2, power, sqrt`

- trigonometric: `sin, cos, tan, acsin, arccos, atctan`

- hyperbolic: `sinh, cosh, tanh, acsinh, arccosh, atctanh`

- bitwise operations: `&, |, ~, ^, left_shift, right_shift`

- logical operations: `and, logical_xor, not, or`

- predicates: `isfinite, isinf, isnan, signbit`

- other: `abs, ceil, floor, mod, modf, round, sinc, sign, trunc`

Pandas

Pandas provides rich data structures and functions designed to make working with structured data fast, easy, and expressive.

Pandas

Pandas is an important Python library for data manipulation, wrangling, and analysis.

Pandas allows you to work with both cross-sectional data and time series based data. So let's get started exploring pandas!

All the data representation in pandas is done using two primary data structures:

➢ Series
➢ Dataframes

Matplotlib

matplotlib is the most popular Python library for producing plots and other 2D data visualizations.

It is well-suited for creating plots suitable for publication.

Provides a comfortable interactive environment for plotting and exploring data.

UCERD

Gathering
Intellectuals
www.ucerd.com

SciPy

SciPy is a collection of packages addressing a number of different standard problem domains in scientific computing.

Importing libraries

import math as m

Created an alias to call functions of math library. User can call function like m.factorial()

from math import *

Imported all the functions user can directly call funciton like factorial()

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

Modules and Functions

import math as mt

mt.functions…

import math

math.cos

from math import cos, pi

cos

from math import *

UCERD
Gathering
Intellectuals
www.ucerd.com

Reading and Writing Data

- Understand the source and type of data

Reading Files

f = open("names.txt")

>>> f.readline()

Results

Uses libraries to deal with complex databases and datastructures.

Files

List of dictionaries

CSV files

Databases

data = pd.read_csv("train.csv")

```python
data = genfromtxt('./signals/ecg.csv',
  delimiter=',')
np.shape(data)


Signal = data2[:,1]


data.head()
```

Data Types/Structure

Lists

Tuples

Set

Dictionary

List, Tuple, Set and Dictionary

- List: Use when user need an ordered sequence of homogenous collections, whose values can be changed later in the program.

my_list = ['a','b','c','b', 'a']

- Tuple: User when you need an ordered sequence of heterogeneous collections whose values need not be changed later in the program.

l = (1, 2, 3)

- Set: It is ideal for user when user don't have to store duplicates and is not concerned about the order or the items. User just want to know whether a particular value already exists or not.

set([1, 2, 3, 4])

- Dictionary: It is ideal for use when user need to relate values with keys, in order to look them up efficiently using a key.

d = {'first':'string value', 'second':[1,2]}

d.keys()

Data Types

Lists – list can simply be defined by writing a list of comma separated values in square brackets. **Lists might contain items of different types, but usually the items all have the same type. Python lists are mutable and individual elements of a list can be changed.**

Lists

A list can be simply defined by writing comma separated values in square brackets.

```
In [1]: squares_list = [0,1,4,9,16,25]
```

```
In [2]: squares_list
Out[2]: [0, 1, 4, 9, 16, 25]
```

Individual elements of a list can be accessed by writing the index number in square bracket. Please note that the first index of a list is 0 and not 1

```
In [3]: squares_list[0] #Indexing returns the item
Out[3]: 0
```

A range of script can be accessed by having first index and last index

```
In [4]: squares_list[2:4] #Slicing returns a new list
Out[4]: [4, 9]
```

A Negative index accesses the list from end

```
In [5]: squares_list[-2] #It should return the second last element in the list
Out[5]: 16
```

A few common methods applicable to lists include: append() extend() insert() remove() pop() count() sort() reverse()

Strings – Strings can simply be defined by use of single ('), double (") or triple ("') inverted commas. Strings enclosed in tripe quotes ("') can span over multiple lines and are used frequently in docstrings (Python's way of documenting functions). \ is used as an escape character. Please note that Python strings are **immutable**, so you can not change part of strings.

Strings

A string can be simply defined by using single ('), double (") or triple ("') quotation

```
In [6]:  greeting = 'Hello'
         print greeting[1]            # Return character on the index 1
         print len(greeting)          # Prints length of string
         print greeting + 'World'     # String Concatenation

         e
         5
         HelloWorld
```

Raw strings can be used to pass on string as is. Python interpretter does not alter the string, if you specify a string to be raw. Raw strings can be defined by adding r to the string

```
In [8]:  stmt = r'\n is a newline character by default.'
         print stmt

         \n is a newline character by default.
```

Python strings are immutable and hence can be changed. Doing so will result in an error

```
In [9]:  greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-9-2da872e33998> in <module>()
         ----> 1 greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

         TypeError: 'str' object does not support item assignment
```

Common string methods include lower(), upper(), strip(), isdigit(), isspace(), find(), replace(), split() and join(). These are usually very helpful when you need to perform data manipulations or cleaning on text fields.

Tuples

A tuple is represented by a number of values separated by commas. Tuples are immutable and the output is surrounded by parentheses so that nested tuples are processed correctly. Additionally, even though tuples are immutable, they can hold mutable data if needed.

Since Tuples are immutable and can not change, they are faster in processing as compared to lists. Hence, if your list is unlikely to change, you should use tuples, instead of lists.

Tuples

A tuple is represented by a number of values separated by commas.

```
In [10]: tuple_example = 0, 1, 4, 9, 16, 25

In [11]: tuple_example #output would be enclosed in paranthesis
Out[11]: (0, 1, 4, 9, 16, 25)

In [12]: tuple_example[2] #Single elements can be accessed in similar fashion
Out[12]: 4

In [13]: tuple_example[2] = 6 #Tuples are immutable and hence this should result in error
         ----------------------------------------------------------------
         TypeError                                Traceback (most recent call last)
         <ipython-input-13-6c410e816018> in <module>()
         ----> 1 tuple_example[2] = 6 #Tuples are immutable and hence this should result in error

         TypeError: 'tuple' object does not support item assignment
```

Dictionary

Dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}.

Dictionary

A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}.

```
In [20]:  extensions = {'Kunal': 9073, 'Tavish' : 9128, 'Sunil' : 9223, 'Nitin' : 9330}
          extensions
```

```
Out[20]:  {'Kunal': 9073, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}
```

```
In [22]:  extensions['Mukesh'] = 9150
          extensions
```

```
Out[22]:  {'Kunal': 9073, 'Mukesh': 9150, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}
```
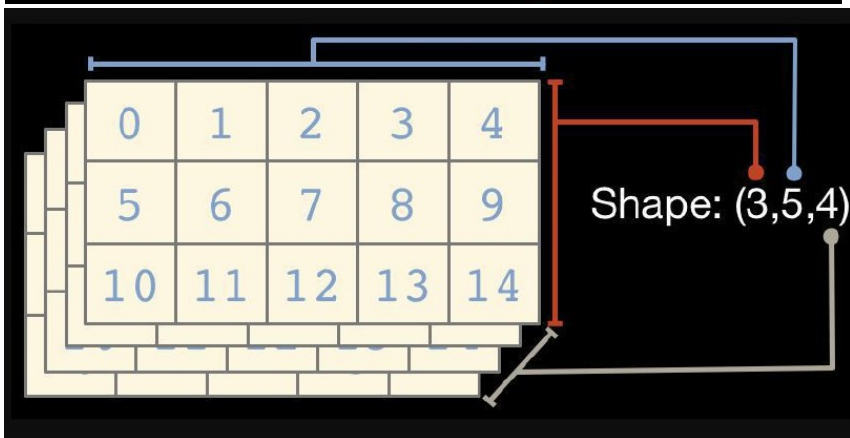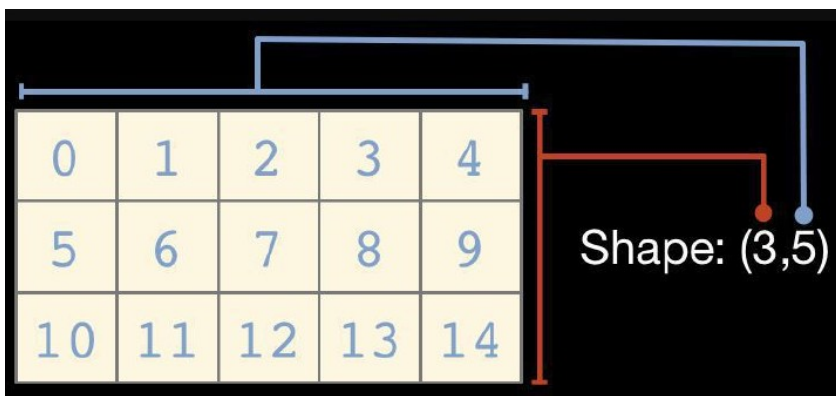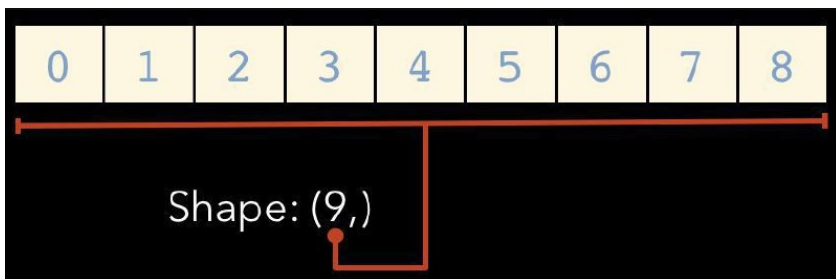
```
In [23]:  extensions.keys()
```

```
Out[23]:  ['Sunil', 'Tavish', 'Kunal', 'Mukesh', 'Nitin']
```

Understand Dataset

data.dtype

np.size(data)

np.shape(data)

Shape: (9,)

Shape: (3,5)

Shape: (3,5,4)

```
In [4]: import numpy as np
   ...: arr = np.array([1,3,4,5,6])
   ...: arr

Out[4]: array([1, 3, 4, 5, 6])

In [5]: arr.shape

Out[5]: (5,)

In [6]: arr.dtype

Out[6]: dtype('int32')

In [16]: arr = np.array([1,'st','er',3])
   ...: arr.dtype
Out[16]: dtype('<U11')

In [17]: np.sum(arr)
```

```
In [19]: arr = np.array([[1,2,3],[2,4,6],[8,8,8]])
   ...: arr.shape

Out[19]: (3, 3)

In [20]: arr

Out[20]:
array([[1, 2, 3],
       [2, 4, 6],
       [8, 8, 8]])

In [21]: arr = np.zeros((2,4))
   ...: arr
Out[21]:array([[ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.]])
```

Accessing Array Element

```
In [50]: arr[0]
Out[50]:
array([[0, 1, 2],
       [3, 4, 5]])

In [57]: arr = np.arange(10)
    ...: arr[5:]
Out[57]: array([5, 6, 7, 8, 9])

In [58]: arr[5:8]
Out[58]: array([5, 6, 7])

In [60]: arr[:-5]
Out[60]: array([0, 1, 2, 3, 4])
```

Linear Algebra Using numpy

```
In [39]: A = np.array([[1,2,3],[4,5,6],[7,8,9]])
    ...: B = np.array([[9,8,7],[6,5,4],[1,2,3]])

In [40]: A.dot(B)
Out[40]:
array([[ 24,  24,  24],
       [ 72,  69,  66],
       [120, 114, 108]])


In [48]: np.linalg.svd(A)
```

Condition

if continuation :

        print value

if gpa > 2 :

    print gpa

Repetition

```
for x in range(1, 6, +1): # range(start, stop, step)
    print x
```

Practice

Introduction to Barcelona Supercomputing Center

Google Colab

Google Colab

Do not know how to install and set up the Python running environment;

Do not know how to find the solutions effectively when facing the problems;

Do not know how to collaborate with others when trying to finish the group tasks；

Do not know how to handle version control, which may lead the code to chaotic.

Google Colab can help you with all of those things.

https://colab.research.google.com

Colab hardware support

By default, these cloud computing hardwares are not enabled. You need to choose "runtime" in the menu bar, and then "Change runtime type".

UCERD
Gathering
Intellectuals
www.ucerd.com