AUG
Faculty of Computer and Information Sciences
Ain Shams University

Session 4

Even More Python

-> Table of Contents <-

Python Packages

Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

- **The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot***
- **An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.**

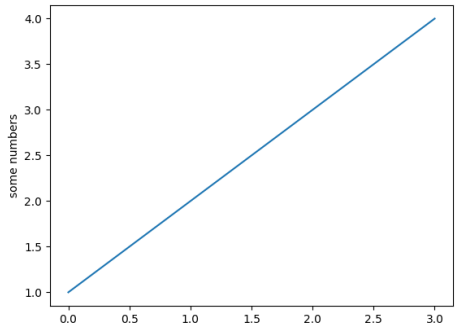
Matplotlib and Pyplot in Python

The pyplot API has a convenient MATLAB-style stateful interface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

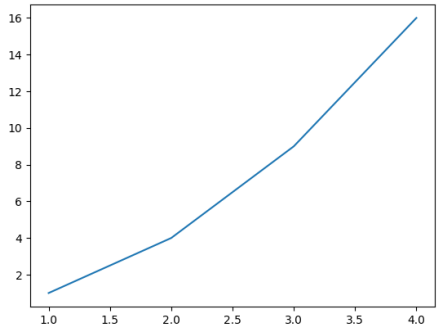
Understanding matplotlib's pyplot API is key to understanding how to work with plots:

- ***matplotlib.pyplot.figure: Figure* is the top-level container. It includes everything visualized in a plot including one or more *Axes*.**
- ***matplotlib.pyplot.axes: Axes* contain most of the elements in a plot: *Axis, Tick, Line2D, Text,* etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.**

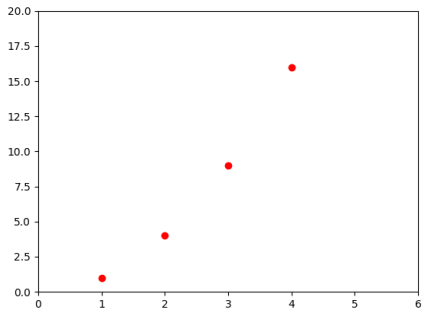
```
plt.plot([1, 2, 3, 4])
```



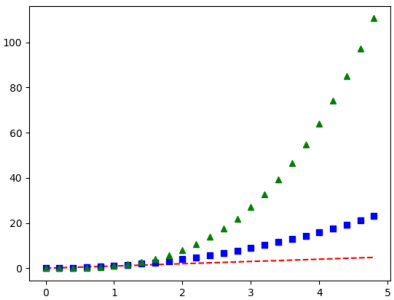

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```



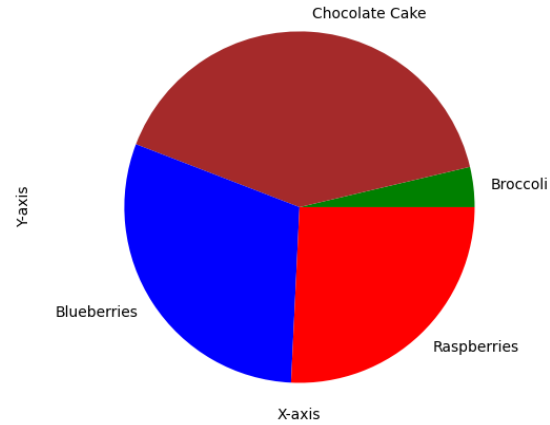
```
plt.plot([1, 2, 3], [1, 4, 9], 'ro')
plt.axis([0, 6, 0, 20])
```



```
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r--', t, t**2,
         'bs', t, t**3, 'g^')
```

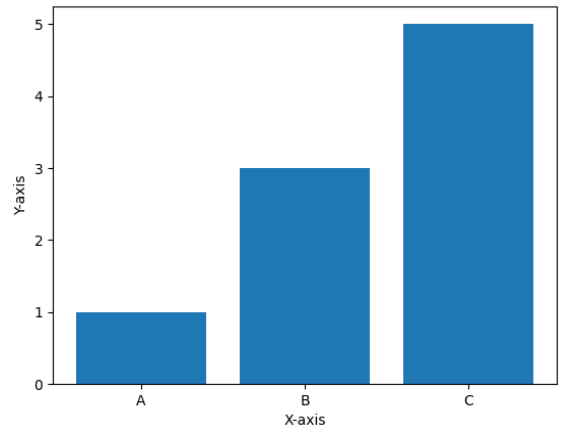


```
labels = 'Broccoli', 'Chocolate Cake',
'Blueberries', 'Raspberries'
sizes = [30, 330, 245, 210]
colors = ['green', 'brown', 'blue',
'red']
plt.pie(sizes, labels = labels, colors
= colors)
plt.axis('equal')
```




```
xdata=['A', 'B', 'C'] #X data
ydata=[1, 3, 5] #Y data

plt.bar(xdata, ydata)
```



`Line2D` properties.

Property	Value Type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance
color or c	any matplotlib color
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	(np.array xdata, np.array ydata)
figure	a matplotlib.figure.Figure instance
label	any string
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']
markeredgecolor or mec	any matplotlib color
markeredgewidth or mew	float value in points
markerfacecolor or mfc	any matplotlib color
markersize or ms	float
markevery	[None | integer | (startind, stride)]
picker	used in interactive line selection
pickradius	the line pick selection radius
solid_capstyle	['butt' | 'round' | 'projecting']
solid_joinstyle	['miter' | 'round' | 'bevel']
transform	a matplotlib.transforms.Transform instance
visible	[True | False]
xdata	np.array
ydata	np.array
zorder	any number

To get a list of settable line properties, call the `setp` function with a line or lines as argument

○ Numpy

NumPy is a fundamental package for scientific computing in Python.

It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ***ndarray* object**.

This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.

- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

- In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

- **ndarray.ndim**

  the number of axes (dimensions) of the array.

- **ndarray.shape**

  the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with *n* rows and *m* columns, `shape` will be `(n,m)`. The length of the `shape` tuple is therefore the number of axes, `ndim`.

- **ndarray.size**

  the total number of elements of the array. This is equal to the product of the elements of `shape`.

- **ndarray.dtype**

  an object describing the type of the elements in the array. One can create or specify dtype using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

Output Range	Output Type	Bytes per Element	Output Class
-128 to 127	Signed 8-bit integer	1	`int8`
-32,768 to 32,767	Signed 16-bit integer	2	`int16`
-2,147,483,648 to 2,147,483,647	Signed 32-bit integer	4	`int32`
-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer	8	`int64`

- **ndarray.ndim**

  the size in bytes of each element of the array. For example, an array of elements of type `float64` has `itemsize` 8 (=64/8), while one of type `complex32` has `itemsize` 4 (=32/8). It is equivalent to `ndarray.dtype.itemsize`.

○ Pandas

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a pandas data structure

■ DataFrame

Say that we want to store passenger data of the Titanic. For a number of passengers, I know the name (characters), age (integers) and sex (male/female) data.To manually store data in a table, create a DataFrame. When using a Python dictionary of lists, the dictionary keys will be used as column headers and the values in each list as columns of the DataFrame.

Each column in a DataFrame is a Series

A DataFrame is a 2-dimensional data structure that can store data of different types (including characters, integers, floating point values, categorical data, etc..) in columns.

For example:-

```
df = pd.DataFrame(
    {
        "Name": [
            "Braund, Mr. Owen Harris",
            "Allen, Mr. William Henry",
            "Bonnell, Miss. Elizabeth",
        ],
        "Age": [22, 35, 58],
        "Sex": ["male", "male", "female"],
    }
)
```

	Name	Age	Sex
0	Braund, Mr. Owen Harris	22	male
1	Allen, Mr. William Henry	35	male
2	Bonnell, Miss. Elizabeth	58	female

■ Read/Write Tabular

Pandas provides the **read_csv()** function to read data stored as a csv file into a pandas DataFrame. pandas supports many different file formats or data sources out of the box (csv, excel, sql, json, parquet, …), each of them with the prefix read_*.

titanic = pd.read_csv("titanic.csv")

	PassengerId	Survived	Pclass	Name	...	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	...	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	...	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	...	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	...	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	...	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	...	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	...	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	...	349909	21.0750	NaN	S

```
titanic.head(2)
```

Gets the First (2) rows of Tabular

```
titanic.tail(2)
```

Gets the Last (2) rows of Tabular

```
titanic.info()
```

Gets the Informations of the whole Tabular

```
In [9]: titanic.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

○ Scipy

SciPy in Python: is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extention. SciPy is also pronounced as "Sigh Pi."

Why use SciPy: it contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.

- SciPy package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.
- SciPy is built in top of the NumPy
- SciPy module in Python is a fully-featured version of Linear Algebra while Numpy contains only a few features.

Most new Data Science features are available in Scipy rather than Numpy.

Special Function package

★ **scipy.special** package contains numerous functions of mathematical physics.

★ **SciPy special function**
includes Cubic Root, Exponential, Log sum Exponential, Lambert, Permutation and Combinations, Gamma, Bessel, hypergeometric, Kelvin, beta, parabolic cylinder, Relative Error Exponential, etc..

★ **Integration (scipy.integrate)**
The scipy.integrate sub-package provides several integration techniques including an ordinary differential equation integrator. An overview of the module is provided by the help command:

★ Examples of Special Functions with Scipy

```
print('Cube Root is: ', cbrt(125))
print('Exponential is: ', exp10([1,10]))
print('combination is: ', comb(5, 2, exact = False, repetition=True))
print('Permutation is: ', perm(5, 2, exact = True))
```

```
Cube Root is:  5.0
Exponential is:  [1.e+01 1.e+10]
combination is:  15.0
Permutation is:  20
```

★ Examples of Matrices with Scipy

```
Square Matrix
matA = np.array([[4, 5], [3, 2]])
matB = np.array([[4, 7], [2, 6]])

Determinant function det()
print('\nDet of Mat(A) = ', linalg.det(matA), '\n')

Inverse function inv()
print('Inv of Mat(B):\n', linalg.inv(matB), '\n')
```

```
Det of Mat(A) =  -7.0

Inv of Mat(B):
 [[ 0.6 -0.7]
 [-0.2  0.4]]
```

★ Examples of Integrations with Scipy

General integration (quad)

The function quad is provided to integrate a function of one variable between two points. The points can be $\pm\infty$ ($\pm$ `inf`) to indicate infinite limits. For example, suppose you wish to integrate a bessel function `jv(2.5, x)` along the interval $[0, 4.5]$.

$$I = \int_0^{4.5} J_{2.5}(x)\, dx.$$

This could be computed using quad:

```
>>> import scipy.integrate as integrate
>>> import scipy.special as special
>>> result = integrate.quad(lambda x: special.jv(2.5,x), 0, 4.5)
>>> result
(1.1178179380783249, 7.8663172481899801e-09)
```

Math

The `math` module is a standard module in Python and is always available. To use mathematical functions under this module, you have to import the module using `import math`.

- **sqrt(x)** Returns the square root of x
- **pow(x, y)** Returns x raised to the power y
- **ceil(x)** Returns the smallest integer greater than or equal to x.
- **floor(x)** Returns the largest integer less than or equal to x
- **fabs(x)** Returns the absolute value of x
- **factorial(x)** Returns the factorial of x
- **fmod(x, y)** Returns the remainder when x is divided by y
- **exp(x)** Returns e**x
- **log(x[, b])** Returns the logarithm of x to the base b (defaults to e)
- **sin(x)** Returns the sine of x
- **cos(x)** Returns the cosine of x
- **tan(x)** Returns the tangent of x
- **asin(x)** Returns the arc sine of x
- **acos(x)** Returns the arc cosine of x
- **atan(x)** Returns the arc tangent of x
- **degrees(x)** Converts angle x from radians to degrees
- **radians(x)** Converts angle x from degrees to radians
- **pi** Mathematical constant, ratio of circumference/diameter
- **e** Mathematical constant e

- **Arrow**
  - Arrow is a Python library that offers a sensible and human-friendly approach to creating, manipulating, formatting and converting dates, times and timestamps. It implements and updates the datetime type, plugging gaps in functionality and providing an intelligent module API that supports many common creation scenarios. Simply put,
  - it helps you work with dates and times with fewer imports and a lot less code.

```
arrow.now()
<Arrow [2021-04-29T21:56:31.716436+02:00]>
```

  - You can Shift time too:

```
arrow.now().Shift(hours = -1)
<Arrow [2021-04-29T20:56:31.716436+02:00]>
```

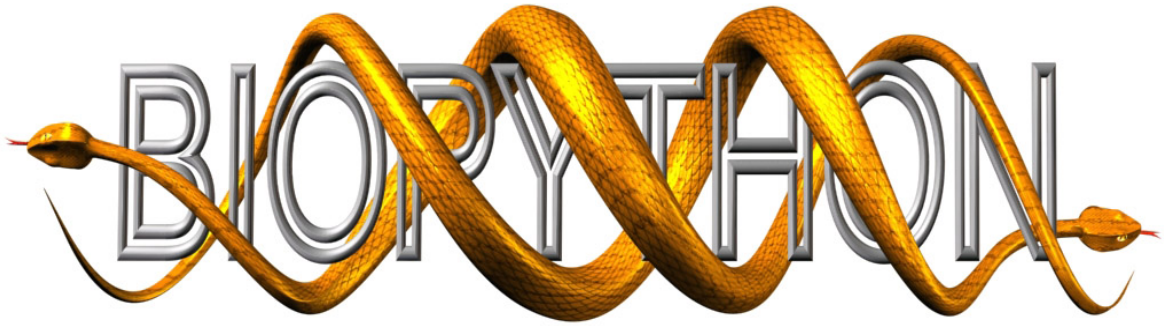
- **Requests**

  The requests library is the standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API.

```
x = requests.get('https://w3schools.com/python/demopage.htm')
print(x. text, '\n')
#Check If the Page exists
if x.status_code == 200:
    print('\tSuccess!')
elif x.status_code == 404:
    print('\tNot Found!')
```

Introduction to Bio-python

Introduction



Biopython is the largest *bioinformatics* package for Python and it is the most popular for Computational Molecular Biology.
It contains a number of different sub-modules for common bioinformatics tasks. It is developed by Chapman and Chang, mainly written in Python. It also contains C code to optimize the complex computation part of the software. It runs on Windows, Linux, Mac OS , etc.

Basically, Biopython is a collection of python modules that provide functions to deal with DNA, RNA & protein sequence

operations such as reverse complementing of a DNA string, finding motifs in protein sequences, etc.

It provides lot of parsers to read all major genetic databases like GenBank, SwissPort, FASTA, etc., as well as wrappers/interfaces to run other popular bioinformatics software/tools like NCBI BLASTN, Entrez, etc.

inside the python environment. It has sibling projects like BioPerl, BioJava and BioRuby.

Installation

You can install Biopython by typing this command in Terminal:-

```
pip install biopython
```

And wait until the package is downloaded and installed successfully.

- **Features**

  Biopython is portable, clear and has easy to learn syntax. Some of the salient features are listed below

  - Interpreted, interactive and object oriented.
  - Supports ***FASTA***, PDB, GenBank, Blast, SCOP, PubMed/Medline, ExPASy-related formats.
  - Option to deal with sequence formats.
  - Tools to manage protein structures.
  - BioSQL – Standard set of SQL tables for storing sequences plus features and annotations.
  - Access to online services and databases, including NCBI services (Blast, Entrez, PubMed) and ExPASY services (SwissProt, Prosite).
  - Access to local services, including Blast, Clustalw, EMBOSS.

- **Advantages**

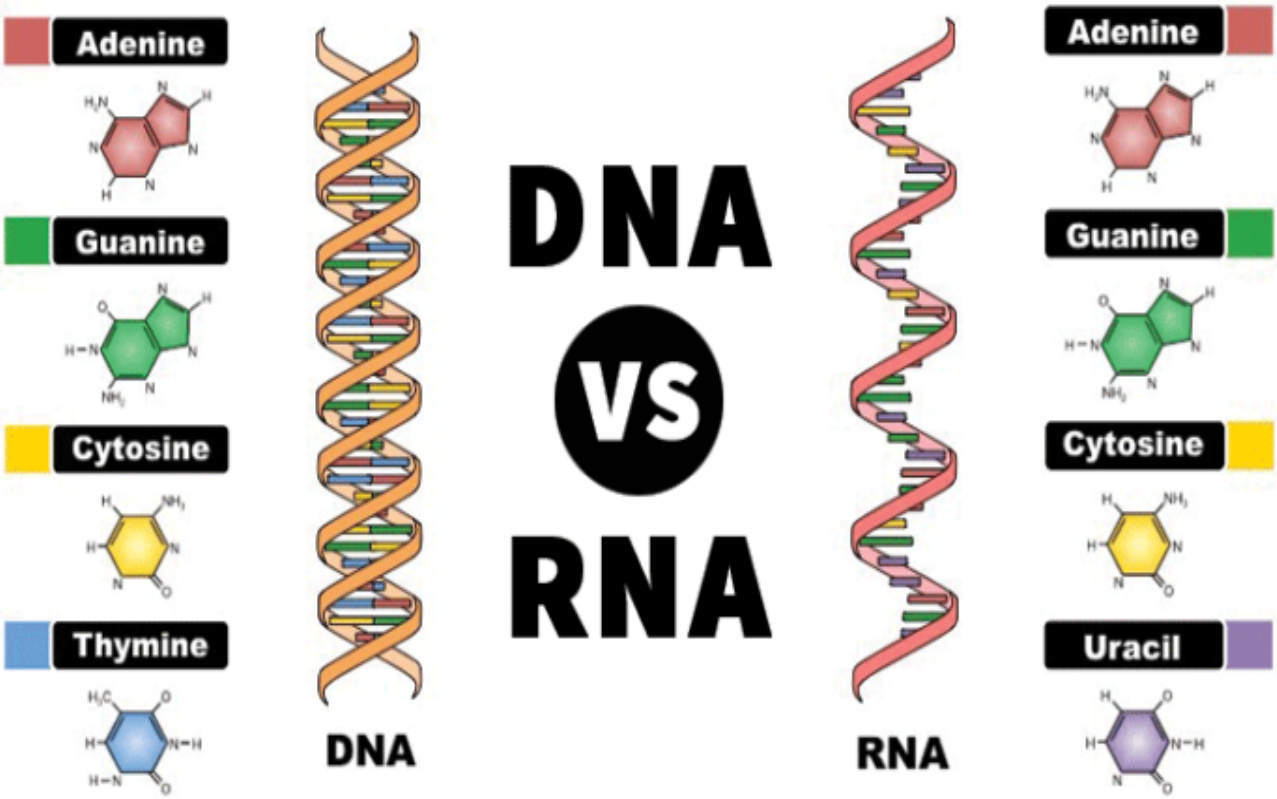
  Biopython requires very less code and comes up with the following advantages

  - Provides microarray data type used in clustering.
  - Reads and writes Tree-View type files.
  - Supports structure data used for PDB parsing, representation and analysis.
  - Supports journal data used in Medline applications.
  - Supports BioSQL database, which is a widely used standard database amongst all bioinformatics projects.
  - Supports parser development by providing modules to parse a bioinformatics file into a *format specific record object or a generic class of sequence plus features.*
  - Clear documentation based on cookbook-style.

- **Goals**

The goal of Biopython is to provide simple, standard and extensive access to bioinformatics through python language. The specific goals of the Biopython are listed below

- Providing standardized access to bioinformatics resources.
- High-quality, reusable modules and scripts.
- Fast array manipulation that can be used in Cluster code, PDB, NaiveBayes and Markov Model.
- Genomic data analysis.



DNA stands for: Deoxyribonucleic Acid: Double Helix

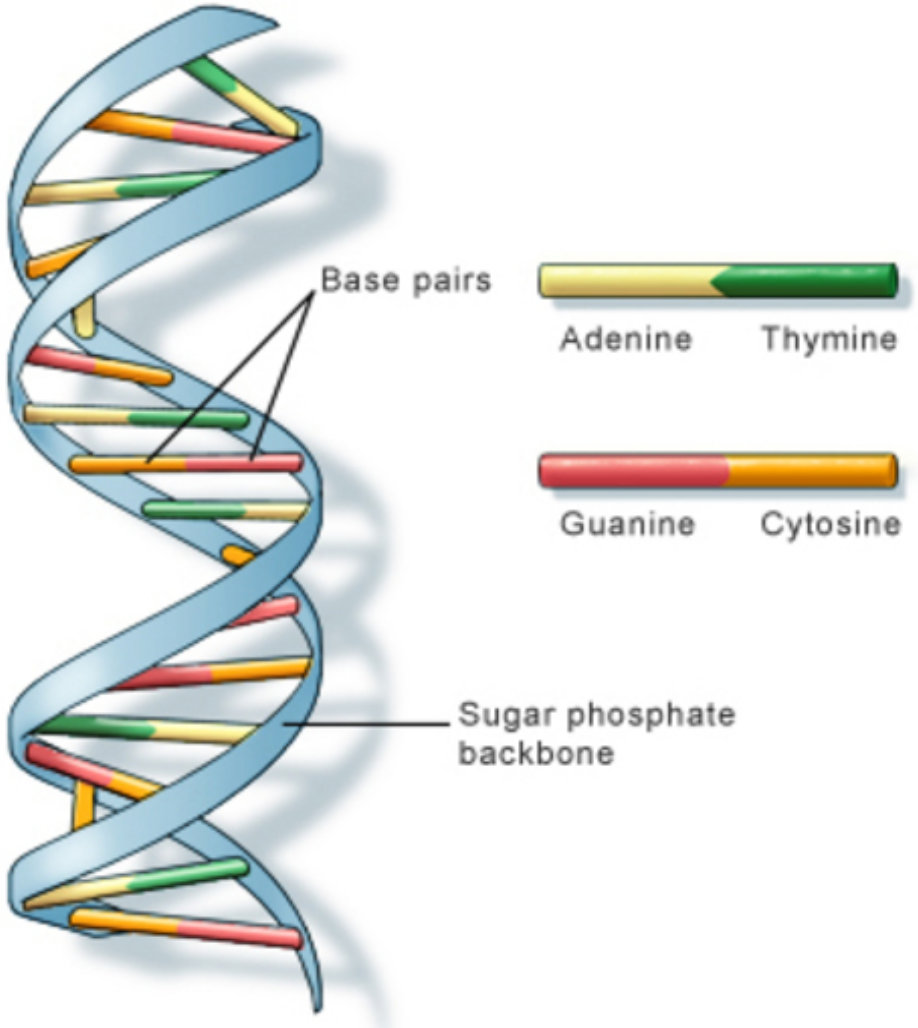
RNA stands for: Ribonucleic Acid: Single Stranded

DNA Review

- **What is DNA**

  DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. Nearly every cell in a person's body has the same DNA. *Most DNA is located in the cell nucleus* (where it is called nuclear DNA), *but a small amount of DNA can also be found in the mitochondria* (where it is called mitochondrial DNA or mtDNA). Mitochondria are structures within cells that convert the energy from food into a form that cells can use.

- **DNA Structure**



  - **The information in DNA is stored as code made up of four chemical bases: adenine(A), thymine(T), guanine(G), cytosine(C).**

- **Human DNA consists of about 3 billion bases, and more than 99% of those bases are the same in all people.**

- **The order, or sequence, of these bases determines the information available for building and maintaining an organism, similar to the way in which letters of the alphabet appear in a certain order to form words and sentences. DNA bases pair up with each other, A with T and C with G, to form units called base pairs.**

- ***Each base is also attached to a sugar molecule and a phosphate molecule.*** **To form nucleotide.**

- **Nucleotides are a base, sugar, and phosphate, arranged into two long strands that form a spiral called a *double helix*.**

- **The structure of the double helix** is somewhat like a ***ladder***, with the **base pairs forming the ladder's rungs** and the **sugar and phosphate molecules forming the side pieces of the ladder.**

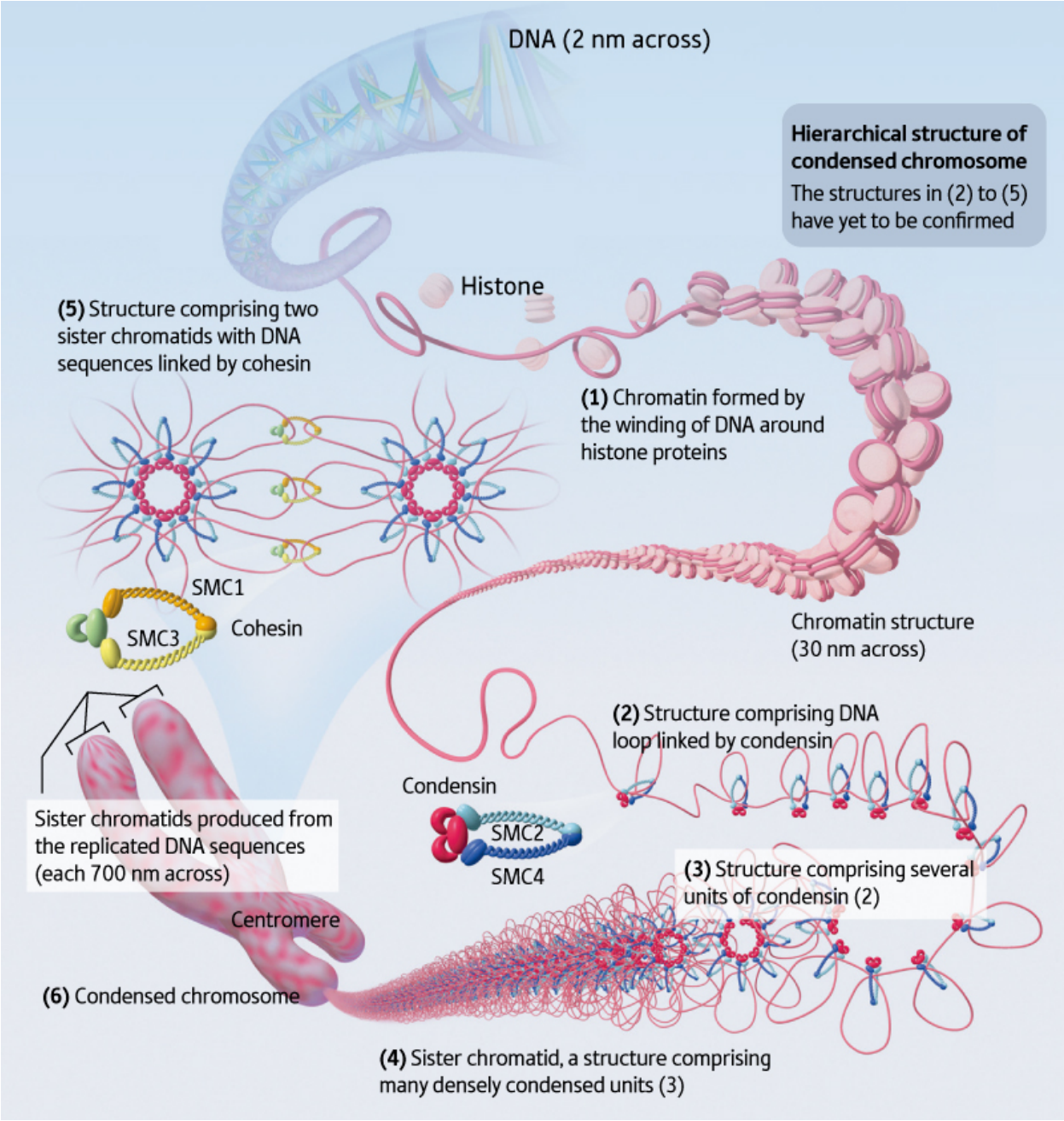
Presence & Form

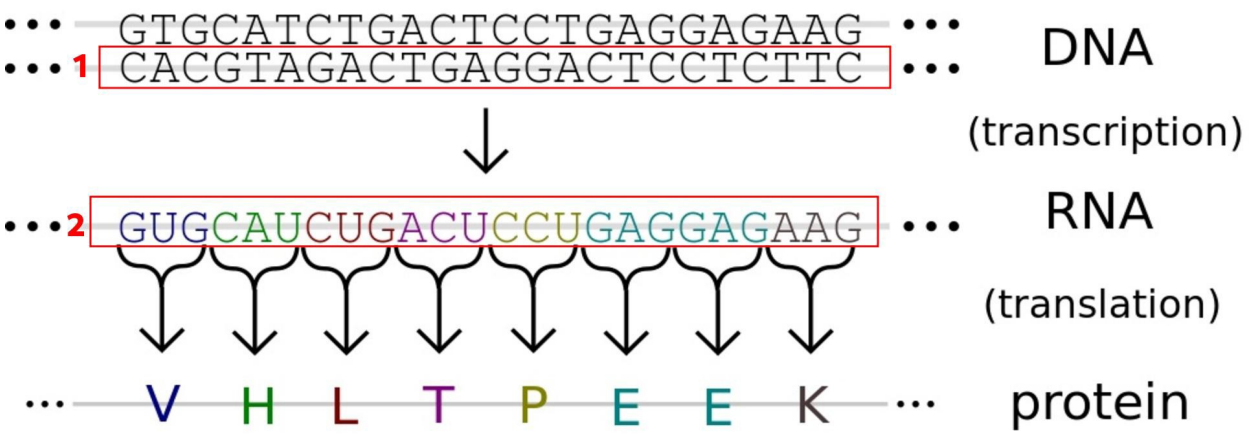
- **We have so much DNA (2 meters) in each cell) and our nuclei are so small, DNA has to be packaged incredibly neatly. So Strands of DNA are looped, coiled and wrapped around proteins called *histones*. In this *coiled state, it is called chromatin*, It is further condensed, through a process called supercoiling, and it is then packaged into structures called *chromosomes*. These chromosomes form the familiar "X" shape, Each chromosome contains one DNA molecule. Humans have 23 pairs of chromosomes or 46 chromosomes in total.**

- **Each length of DNA that codes for a specific protein is called a Gene.**

- **Chromosome 1 is the largest with around 8,000 genes**

  Chromosome 21 is the Smallest with around 3,000 genes.


DNA (2 nm across)
Hierarchical structure of condensed chromosome
The structures in (2) to (5) have yet to be confirmed
Histone
(5) Structure comprising two sister chromatids with DNA sequences linked by cohesin
(1) Chromatin formed by the winding of DNA around histone proteins
SMC1
SMC3
Cohesin
Chromatin structure (30 nm across)
(2) Structure comprising DNA loop linked by condensin
Condensin
SMC2
SMC4
Sister chromatids produced from the replicated DNA sequences (each 700 nm across)
(3) Structure comprising several units of condensin (2)
Centromere
(6) Condensed chromosome
(4) Sister chromatid, a structure comprising many densely condensed units (3)


GTGCATCTGACTCCTGAGGAGAAG
1 CACGTAGACTGAGGACTCCTCTTC
DNA
(transcription)
2 GUGCAUCUGACUCCUGAGGAGAAG
RNA
(translation)
V H L T P E E K
protein

● Introduction to Rosalind

○ What is Rosalind

Rosalind is a platform for learning bioinformatics and programming through problem solving.

Python Village

If you are completely new to programming, try these initial problems to learn a few basics about the Python programming language. You'll get familiar with the operations needed to start solving bioinformatics challenges in the Stronghold.

Bioinformatics Stronghold

Discover the algorithms underlying a variety of bioinformatics topics: computational mass spectrometry, alignment, dynamic programming, genome assembly, genome rearrangements, phylogeny, probability, string algorithms and others.

Bioinformatics Armory

Ready-to-use software tools abound for bioinformatics analysis. Whereas in the Stronghold you implement algorithms on your own, in the Armory you solve similar problems by using existing tools.

Bioinformatics Textbook Track

A collection of exercises to accompany Bioinformatics Algorithms: An Active-Learning Approach by Phillip Compeau & Pavel Pevzner. A full version of this text is hosted on stepic.org

Algorithmic Heights

A collection of exercises in introductory algorithms to accompany "Algorithms", the popular textbook by Dasgupta, Papadimitriou, and Vazirani.

○ Getting Started

You can get started
by going to Rosalind site:->

Click here:-> Rosalind

Register a new user and Start your Journey.