Class, Object & Method

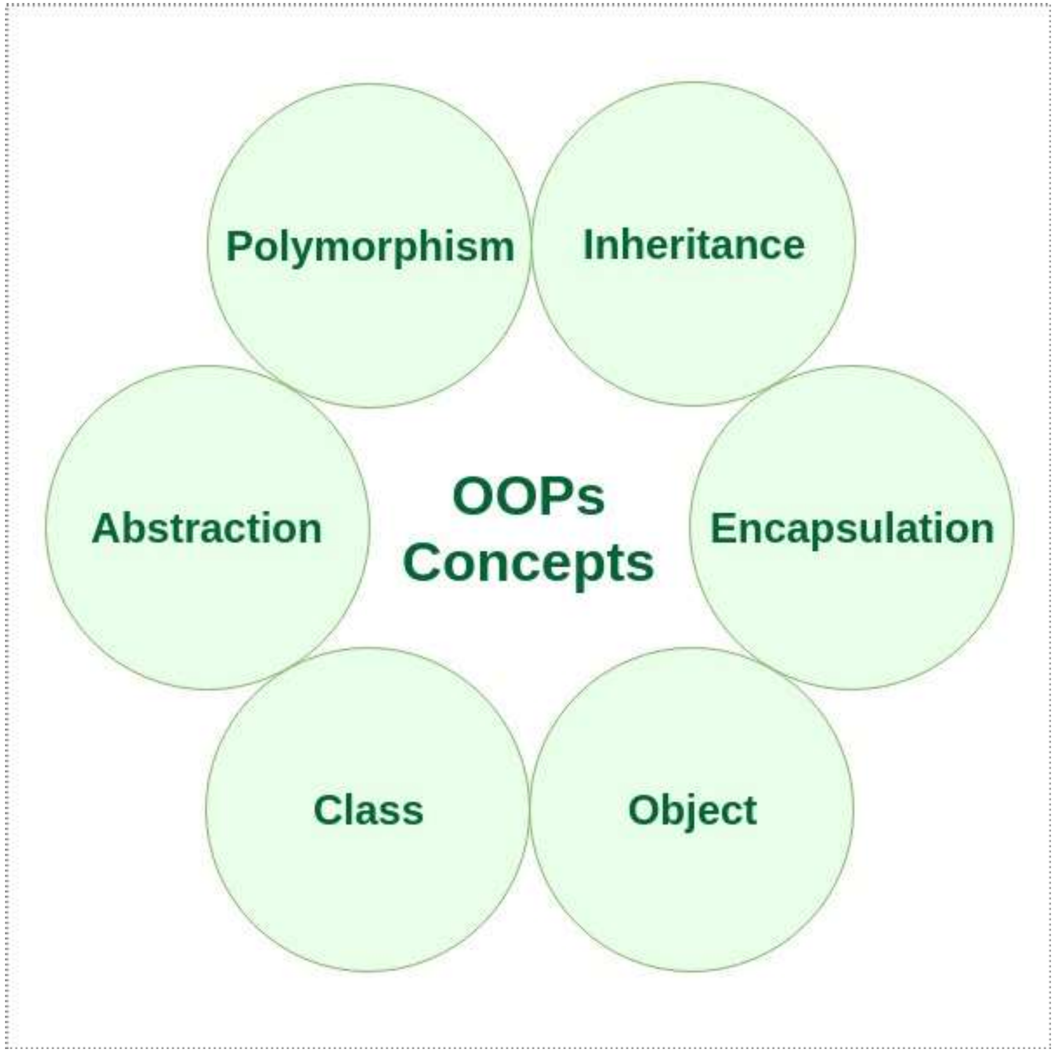
By Aksadur Rahman
aksadur@yahoo.com

Agenda

Object Oriented Programing (OOP)
Classes
Objects
Introducing Method
Default Constructors
Parameterized Constructor
Pass Statement
Intro to Inheritance
Single Inheritance
Hierarchical Inheritance
Multilevel Inheritance
Multiple Inheritance
Method Overloading
Method Overriding
Encapsulation
Polymorphism

Object Oriented Programing (OOP)

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.



Classes

Classes are user-defined data types that act as the blueprint for individual objects, attributes and methods

An example of a class is the class Student. Students usually have a roll and gpa; these are attributes.

```
class student :
   roll= ""
   gpa = ""
```

Objects

An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values.

```
kamal = student()
kamal.roll = 10
kamal.gpa = 3.75
print(f"Roll ={kamal.roll}, GPA={kamal.gpa}")
```

Introducing Method

A method is a function that “belongs to” an object.

```
class student:
    def set_value(self, a, b):
        self.roll = a
        self.gpa = b

    def display(self):
        print(f"Roll ={self.roll}, GPA={self.gpa}")

kamal = student()

kamal.set_value(10, 3.75)
kamal.display()
```

Default Constructors

Constructors are generally used for instantiating an object

```
class student:
    def __init__(self):
        self.section="A"

    def display(self):
        print(f"section = {self.section}")

kamal = student()
kamal.display()
```

Parameterized Constructors

Constructors are generally used for instantiating an object

```
class student:
    def __init__(self, roll, gpa):
        self.roll=roll
        self.gpa=gpa

    def display(self):
        print(f"Roll ={self.roll}, GPA={self.gpa}")

kamal = student(10, 3.75)
kamal.display()
```

Pass Statement

Create a placeholder for future code:

```
class Person:
    pass
```

```
def myfunction():
  pass
```

Intro to Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.
Child class is the class that inherits from another class, also called derived class.

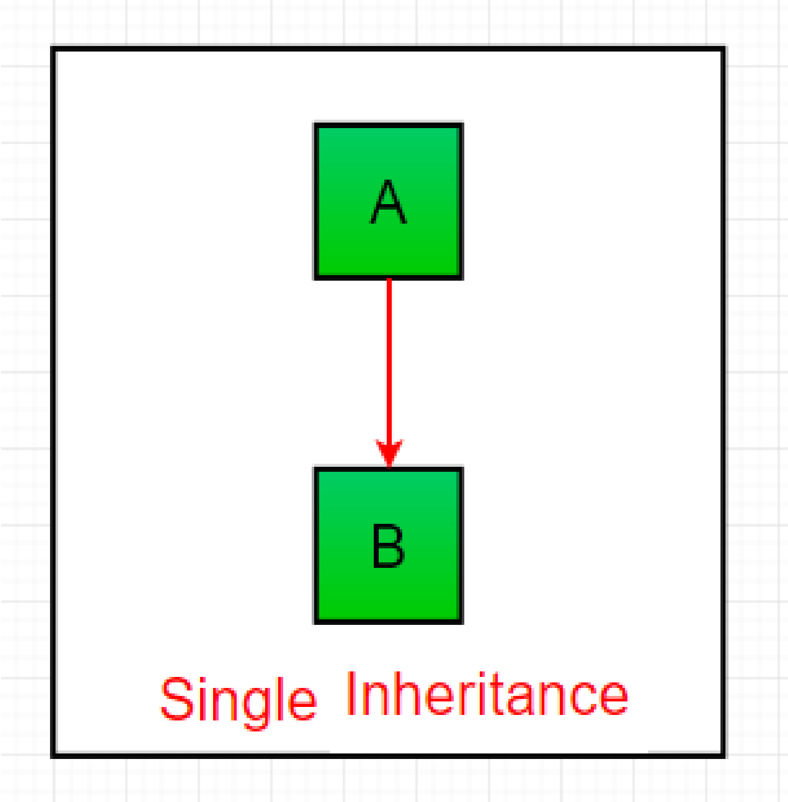
Parent class

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)
```

Child class

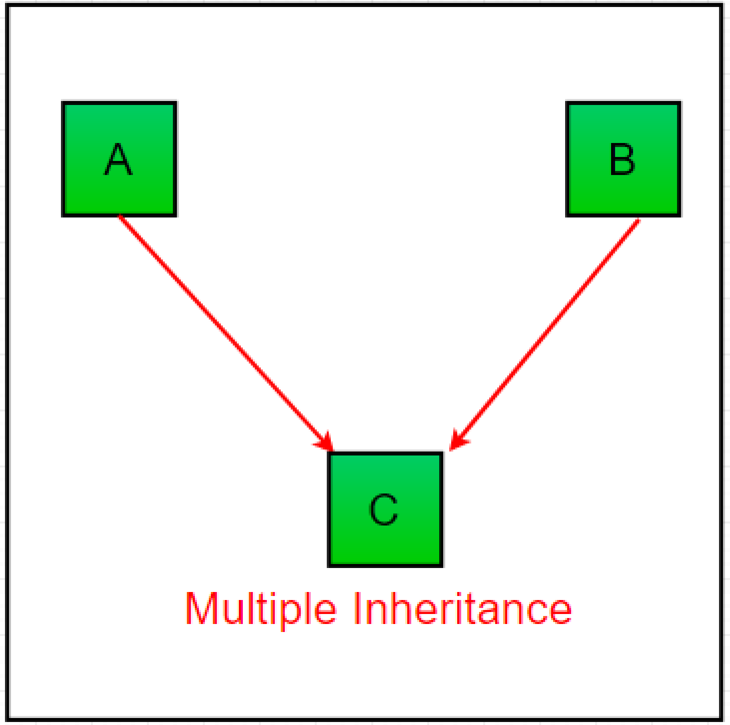
```
class Student(Person):
  pass
#--------------------------------
y = Student("Abul", "Hossain")
y.printname()
```

Single Inheritance



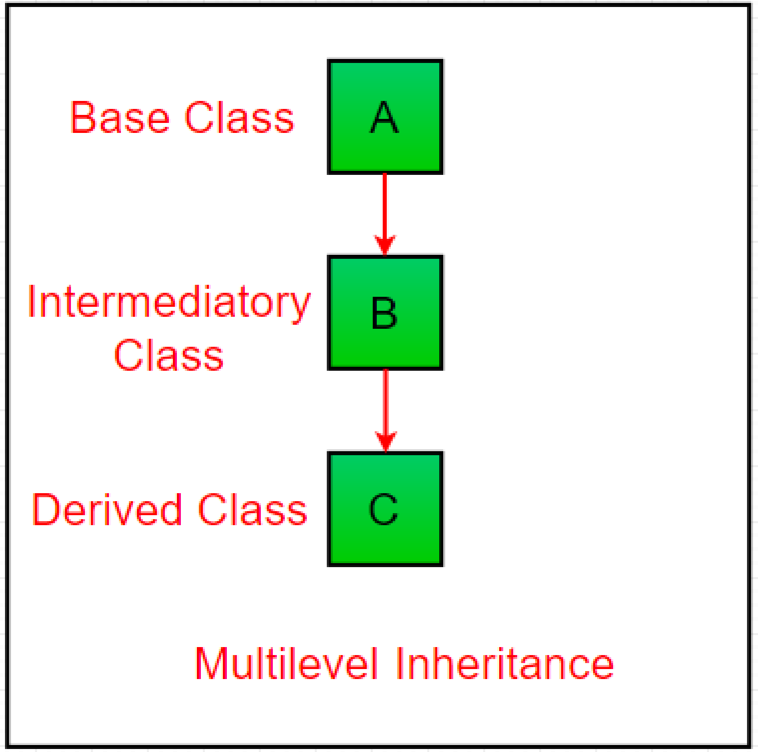

```
class A:
    def display1(self):
        print("This is class A")

class B(A):
    def display2(self):
        print("This is class B")

objB = B()
objB.display1()
objB.display2()
```

Multiple Inheritance



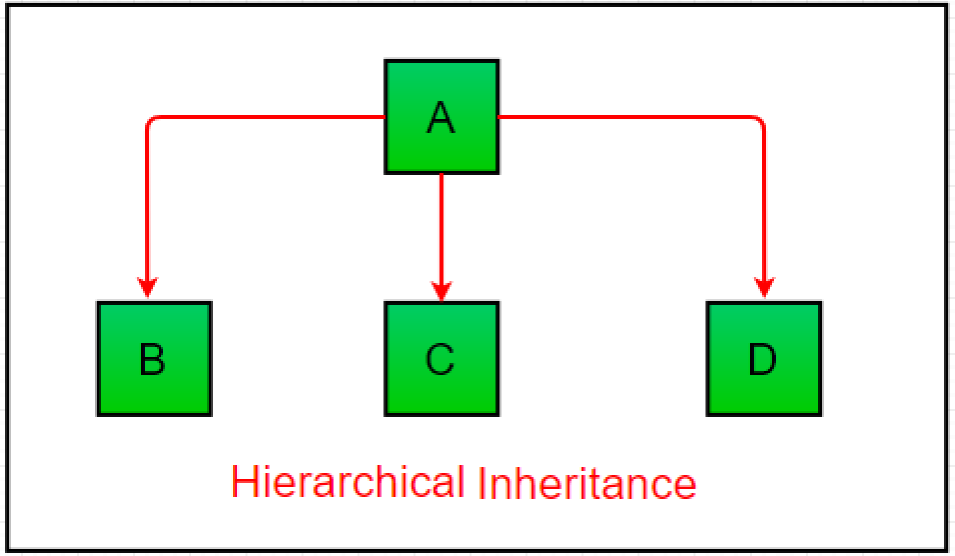

```
class A:
    def display1(self):
        print("This is class A")

class B:
    def display2(self):
        print("This is class B")

class C(A, B):
    def display3(self):
        print("This is class C")

objC = C()
objC.display1()
objC.display2()
objC.display3()
```

Multilevel Inheritance




```
class A:
    def display1(self):
        print("This is class A")

class B(A):
    def display2(self):
        print("This is class B")

class C(B):
    def display3(self):
        print("This is class C")

objC = C()

objC.display1()
objC.display2()
objC.display3()
```

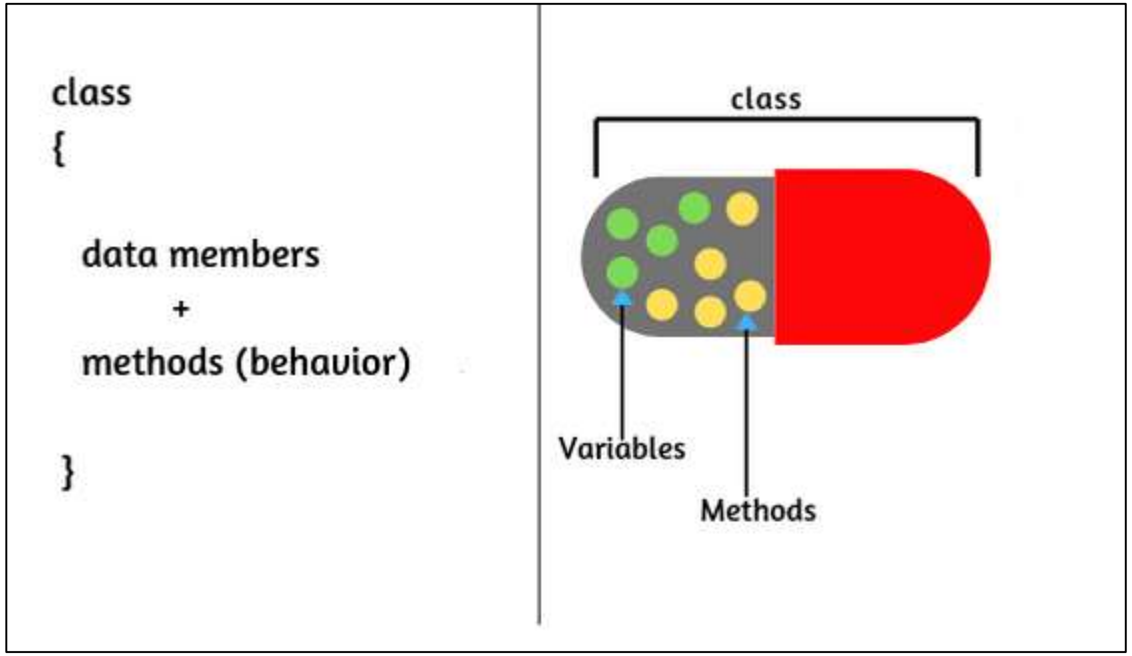
Hierarchical Inheritance




```python
class Parent:  # Base class
    def func1(self):
        print("This function is in parent class.")

class Child1(Parent): # Derived class1
    def func2(self):
        print("This function is in child 1.")

class Child2(Parent): # Derivied class2
    def func3(self):
        print("This function is in child 2.")

# Driver's code
object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

Encapsulation

Encapsulation in Python describes the concept of bundling data and methods within a single unit. So, for example, when you create a class, it means you are implementing encapsulation.



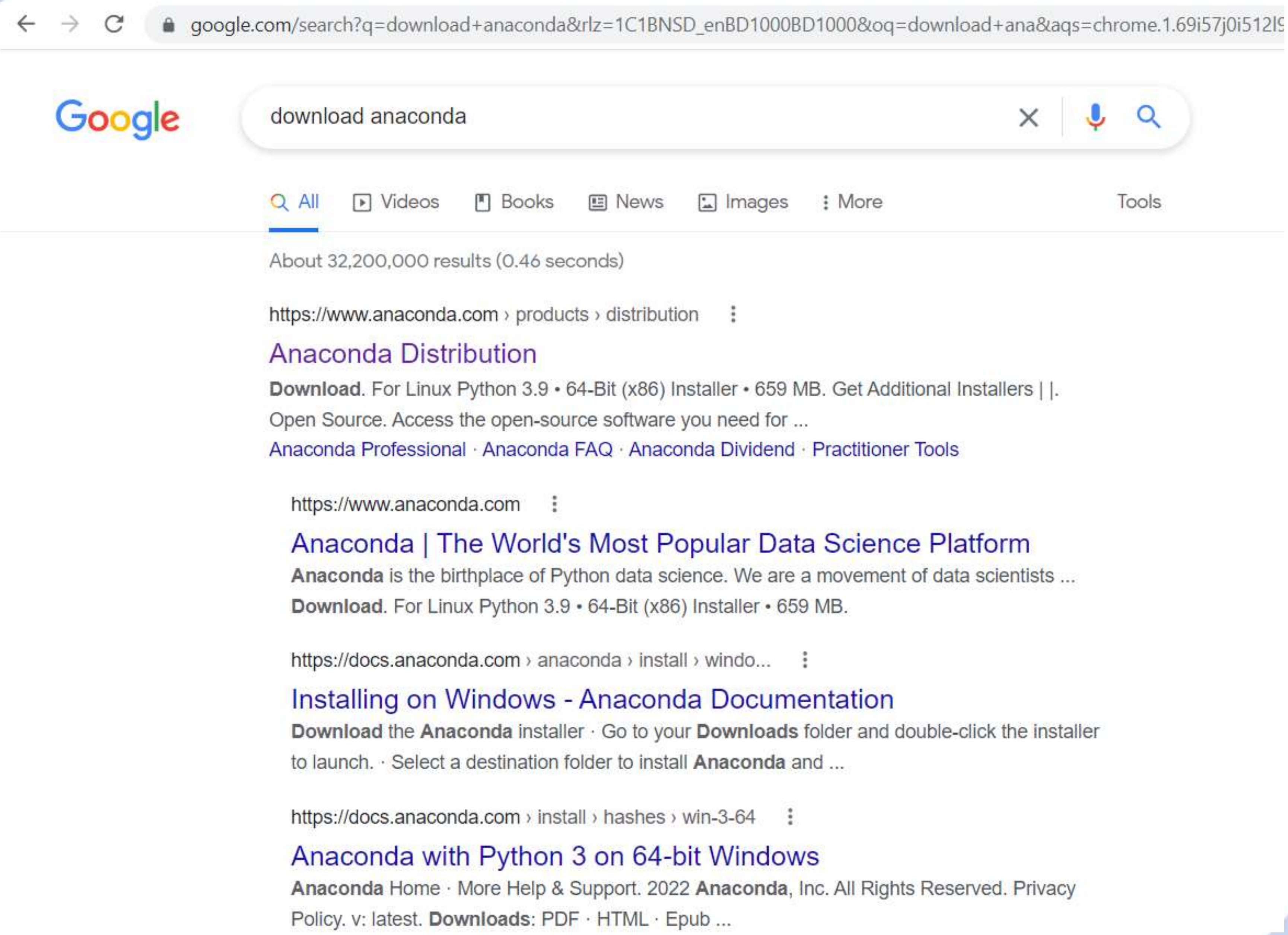
Polymorphism

Polymorphism is taken from the Greek words Poly (many) and morphism (forms). It means that the same function name can be used for different types.

```
#Built in Polymorphic function
print(len("Aksadur Rahman"))
print(len([10, 20, 30]))

#User define polymorphic function
def add(x, y, z=0):
    return x+y+z

print(add(30, 20))
print(add(10, 30, 20))
```

Data Analysis using NumPy

By Aksadur Rahman
aksadur@yahoo.com


google.com/search?q=download+anaconda&rlz=1C1BNSD_enBD1000BD1000&oq=download+ana&aqs=chrome.1.69i57j0i512l9
Google
download anaconda
All
Videos
Books
News
Images
More
Tools
About 32,200,000 results (0.46 seconds)
https://www.anaconda.com › products › distribution
Anaconda Distribution
Download. For Linux Python 3.9 • 64-Bit (x86) Installer • 659 MB. Get Additional Installers | |. Open Source. Access the open-source software you need for ...
Anaconda Professional · Anaconda FAQ · Anaconda Dividend · Practitioner Tools
https://www.anaconda.com
Anaconda | The World's Most Popular Data Science Platform
Anaconda is the birthplace of Python data science. We are a movement of data scientists ... Download. For Linux Python 3.9 • 64-Bit (x86) Installer • 659 MB.
https://docs.anaconda.com › anaconda › install › windo...
Installing on Windows - Anaconda Documentation
Download the Anaconda installer · Go to your Downloads folder and double-click the installer to launch. · Select a destination folder to install Anaconda and ...
https://docs.anaconda.com › install › hashes › win-3-64
Anaconda with Python 3 on 64-bit Windows
Anaconda Home · More Help & Support. 2022 Anaconda, Inc. All Rights Reserved. Privacy Policy. v: latest. Downloads: PDF · HTML · Epub ...

anaconda.com/products/distribution

ANACONDA.

Products | Pricing | Solutions | Resources | Partners | Blog | Company

Contact Sales

Individual Edition is now

ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform

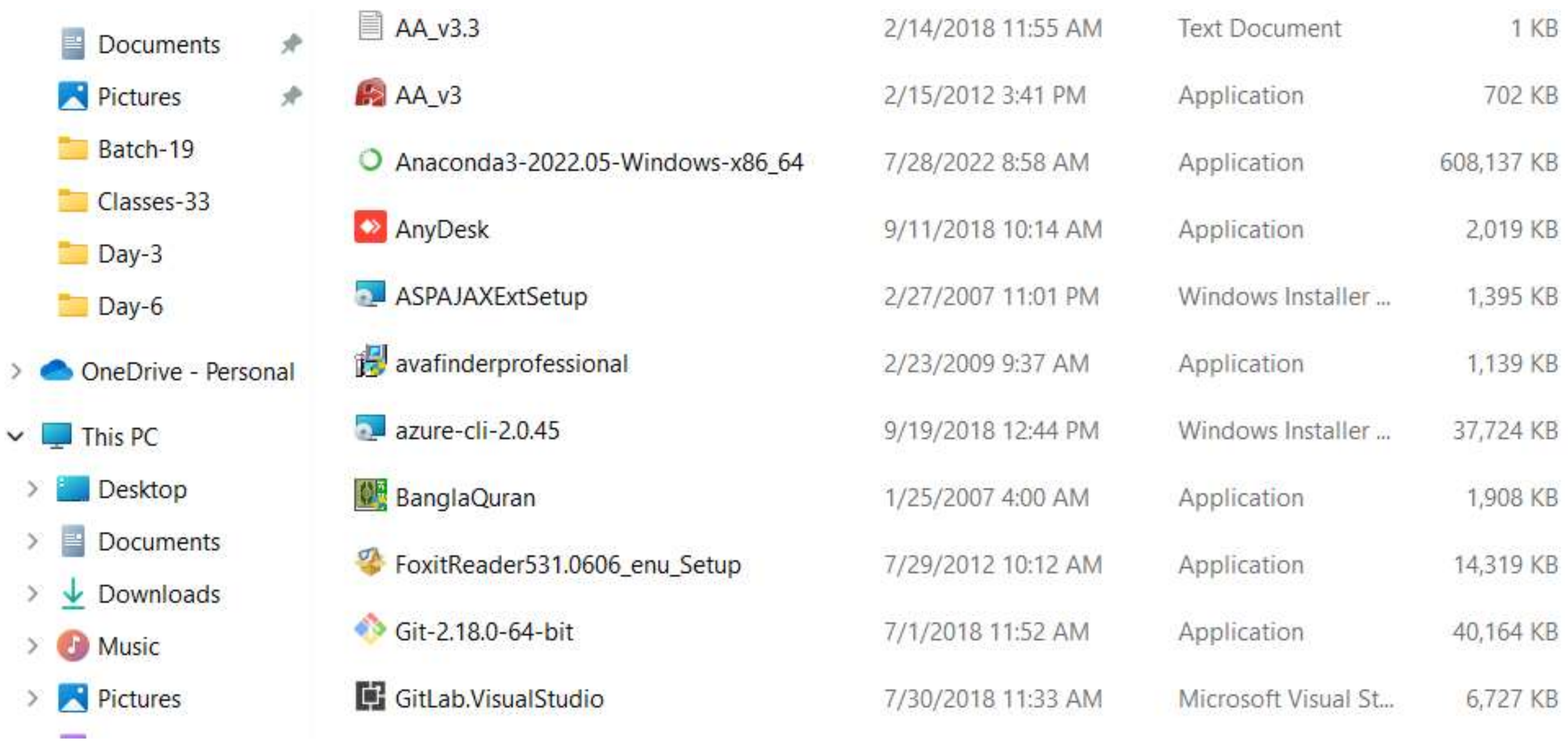
Anaconda Distribution

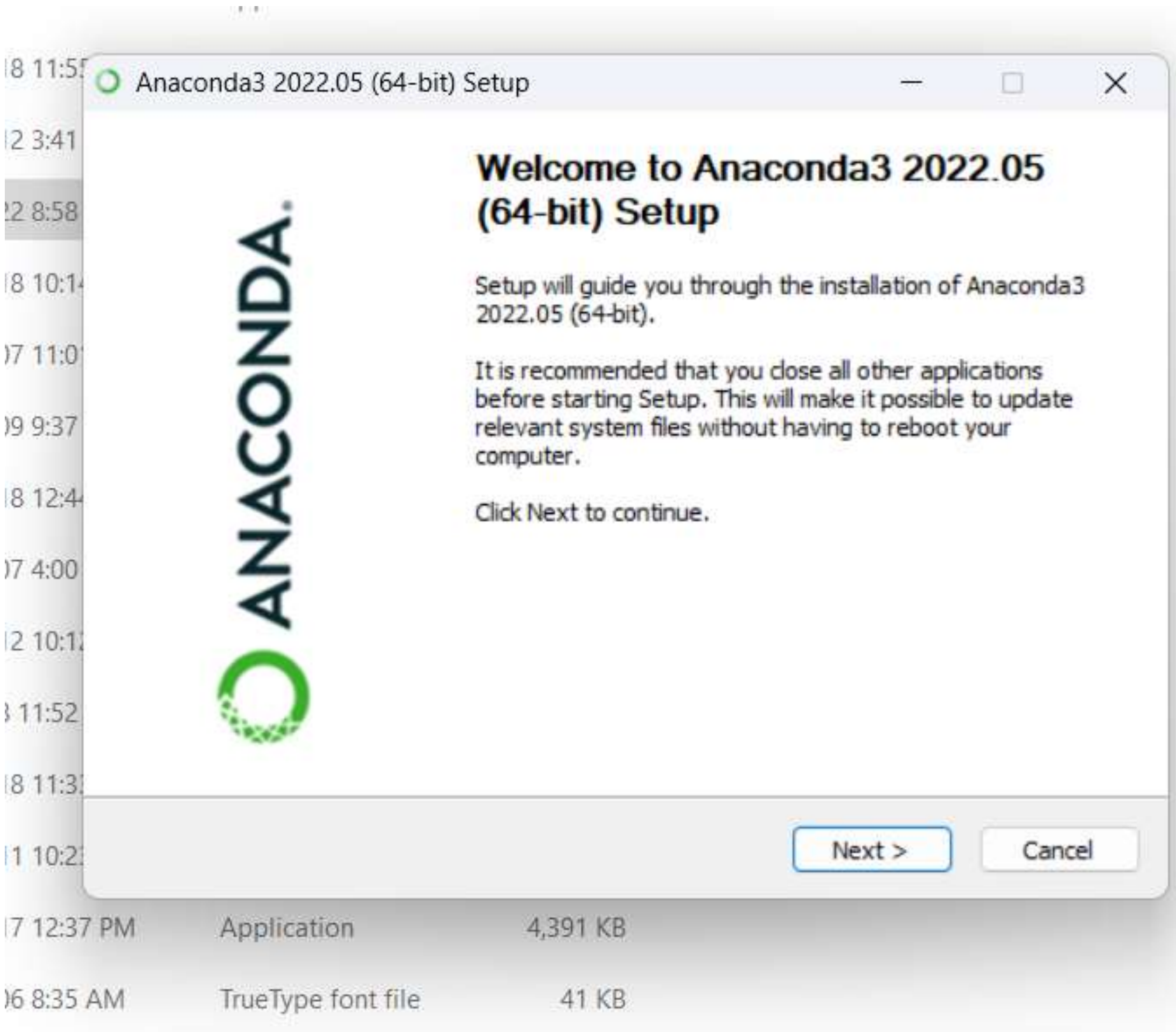
Download

For Windows

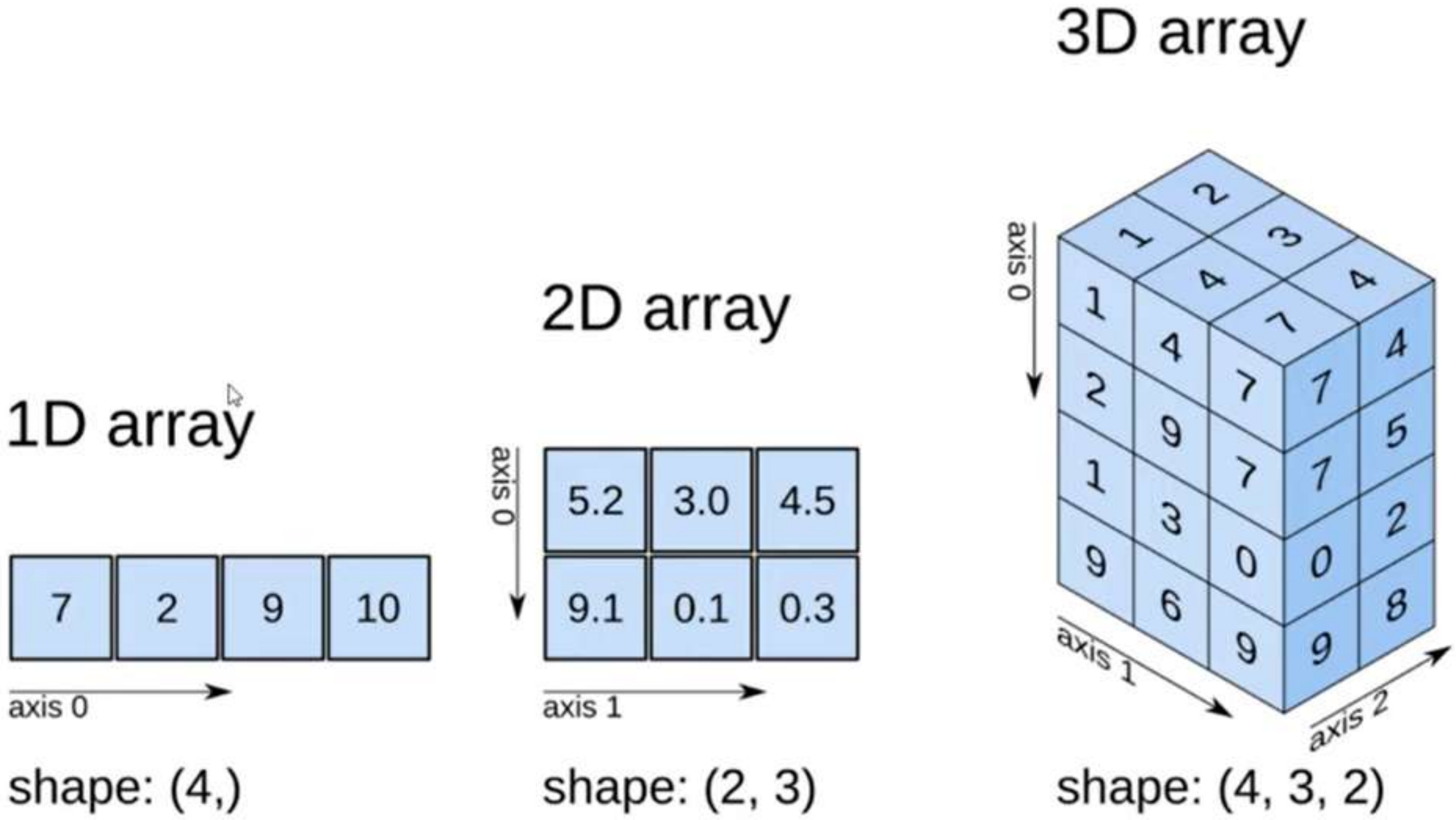
Python 3.9 • 64-Bit Graphical Installer • 594 MB

Get Additional Installers


Documents
Pictures
Batch-19
Classes-33
Day-3
Day-6
OneDrive - Personal
This PC
Desktop
Documents
Downloads
Music
Pictures
AA_v3.3 2/14/2018 11:55 AM Text Document 1 KB
AA_v3 2/15/2012 3:41 PM Application 702 KB
Anaconda3-2022.05-Windows-x86_64 7/28/2022 8:58 AM Application 608,137 KB
AnyDesk 9/11/2018 10:14 AM Application 2,019 KB
ASPAJAXExtSetup 2/27/2007 11:01 PM Windows Installer ... 1,395 KB
avafinderprofessional 2/23/2009 9:37 AM Application 1,139 KB
azure-cli-2.0.45 9/19/2018 12:44 PM Windows Installer ... 37,724 KB
BanglaQuran 1/25/2007 4:00 AM Application 1,908 KB
FoxitReader531.0606_enu_Setup 7/29/2012 10:12 AM Application 14,319 KB
Git-2.18.0-64-bit 7/1/2018 11:52 AM Application 40,164 KB
GitLab.VisualStudio 7/30/2018 11:33 AM Microsoft Visual St... 6,727 KB

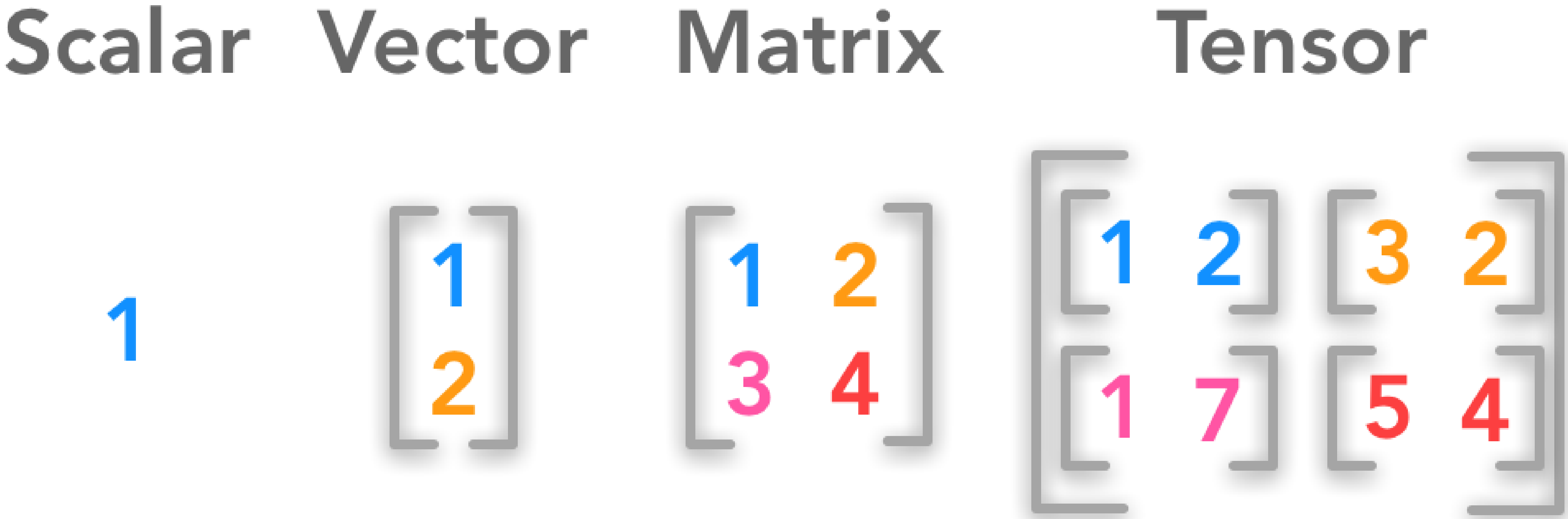

Anaconda3 2022.05 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 2022.05 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel
Application
4,391 KB
TrueType font file
41 KB

Numpy Array



1D array
7
2
9
10
axis 0
shape: (4,)
2D array
axis 0
5.2
3.0
4.5
9.1
0.1
0.3
axis 1
shape: (2, 3)
3D array
axis 0
1
2
3
4
7
4
1
4
7
7
4
2
9
7
7
5
1
3
7
2
9
0
0
6
9
8
9
axis 1
axis 2
shape: (4, 3, 2)

Numpy Array



List vs Numpy Array

Similarities between list and Numpy array:

Storing Data

Can be Indexed

Mutable

Slicing Operation

List vs Numpy Array

Difference between list and Numpy array:

List : Different Datatypes [1, 2.1, "a", 1]

Array: Similar Datatypes [1,2,3,4]

Numpy Array: Install Numpy

List : Built_in

List vs Numpy Array

A list cannot directly handle a mathematical operations, while array can

```
In [26]: list = [0,1,2]

In [27]: list*2
Out[27]: [0, 1, 2, 0, 1, 2]

In [28]: arr = np.array([0,1,2])

In [29]: arr*2
Out[29]: array([0, 2, 4])
```

List vs Numpy Array

An array consumes less memory than a list

```
In [41]: # importing system module
         import sys

In [43]: list
Out[43]: [0, 1, 2]

In [35]: sys.getsizeof(list)
Out[35]: 80

In [44]: arr
Out[44]: array([0, 1, 2])

In [40]: arr.itemsize
Out[40]: 4
```

Using an array is faster than a list

A list is easier to modify