Colab Link - https://colab.research.google.com/drive/1usEscu2NP6TNPdvQ9RNG3NegnHhdpoJG?usp=sharing

```
# math
```

```
import math # gets imported as an object
```

```
type(math)
```

```
    module
```

```
# variables present in math.py --> attributes of the object
# functions present in math.py --> methods of the object
```

```
math.pi
```

```
    3.141592653589793
```

```
# math.py

# pi = 3.14
```

Saving...

```
math.factorial(5)
```

```
    120
```

```
math.pow(3, 2)
```

```
    9.0
```

```
math.floor(3.4)
```

```
    3
```

```
math.floor(-3.4)
```

```
    -4
```

```
math.ceil(3.4)
```

```
4
```

```
help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign c

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.

    degrees(x, /)
        Convert angle x from radians to degrees.

    erf(x, /)
        Error function at x.
```

```
    erfc(x, /)
        Complementary error function at x.
```

```
# random
```

```
import random
```

```
random.randint(1, 100) # random integer from 1 to 100
```

```
95
```

```
# random dataset DS/ML - random, different in every run
```

```
random.seed(100)
random.randint(0, 100)
```

```
18
```

```
random.seed(100)
print(random.randint(0, 100))
print(random.randint(0, 100))
print(random.randint(0, 100))
print(random.randint(0, 100))
print(random.randint(0, 100))
# pseudo-random behavior
```

```
18
```

Saving...

```
22
```

```
# os
```

```
import os
```

```
help(os)
```

```
Help on module os:

NAME
    os - OS routines for NT or Posix depending on what system we're on.

MODULE REFERENCE
    https://docs.python.org/3.7/library/os

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.
```

```
    DESCRIPTION
        This exports:
          - all functions from posix or nt, e.g. unlink, stat, etc.
          - os.path is either posixpath or ntpath
          - os.name is either 'posix' or 'nt'
          - os.curdir is a string representing the current directory (always '.')
          - os.pardir is a string representing the parent directory (always '..')
          - os.sep is the (or a most common) pathname separator ('/' or '\\')
          - os.extsep is the extension separator (always '.')
          - os.altsep is the alternate pathname separator (None or '/')
          - os.pathsep is the component separator used in $PATH etc
          - os.linesep is the line separator in text files ('\r' or '\n' or '\r\n'
          - os.defpath is the default search path for executables
          - os.devnull is the file path of the null device ('/dev/null', etc.)

        Programs that import and use 'os' stand a better chance of being
        portable between different platforms.  Of course, they must then
        only use functions that are defined by all platforms (e.g., unlink
        and opendir), and leave all pathname manipulation to os.path
        (e.g., split and join).

    CLASSES
        builtins.Exception(builtins.BaseException)
            builtins.OSError
        builtins.object
            posix.DirEntry
        builtins.tuple(builtins.object)
            stat_result
            statvfs_result
            terminal_size
            posix.sched_param
            posix.times_result
```

Saving...

```
        class DirEntry(builtins.object)
         |  Methods defined here:
         |
         |  __fspath__(self, /)
         |      Returns the path for the entry.
         |
         |  __repr__(self, /)
         |      Return repr(self).
         |
```

```
os.getcwd()
```

```
'/content'
```

```
os.mkdir("hello")
```

```
os.listdir()
```

```
['.config', 'hello', 'sample_data']
```

```
os.removedirs("hello")
```

```
os.listdir()
```

```
['.config', 'sample_data']
```

```
# pathlib
```

```
import pathlib
```

```
path = pathlib.Path("/")
```

```
path
```

```
PosixPath('/')
```

```
path.cwd()
```

```
PosixPath('/content')
```

```
path = pathlib.Path("/content")
```

```
path.is_dir()
```

```
True
```

Saving...

```
False
```

```
help(sum)
```

```
Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

```
sum?
```

```
def hello():
  """
  This docstring is hello function - does nothing
  Input Args: None
  Return: None
```

```
    """
    return None

help(hello)
```

```
    Help on function hello in module __main__:

    hello()
        This docstring is hello function - does nothing
        Input Args: None
        Return: None
```

```
path = pathlib.Path("/dsml-course/lecture-1/python-script.py")

# different importing techniques

import math

math.sqrt(4) # verbose
```

```
    2.0
```

```
mth_sqrt = math.sqrt
mth_sqrt(2)
```

```
    1.4142135623730951
```

Saving...

```
rect

m = math

m.sqrt(2)
```

```
    1.4142135623730951
```

```
# popular alias
# - np - numpy
# - pd - pandas
# - plt - matplotlib.pyplot

import numpy
np = numpy

import numpy as np

import math as m
```

```
m.sqrt(2)
```

```
1.4142135623730951
```

```
from math import pi
```

```
pi
```

```
3.141592653589793
```

```
from math import pi, sqrt
```

```
sqrt(2)
```

```
1.4142135623730951
```

```
from math import pi as anants_fav_no
```

```
anants_fav_no
```

```
3.141592653589793
```

```
from math import *
```

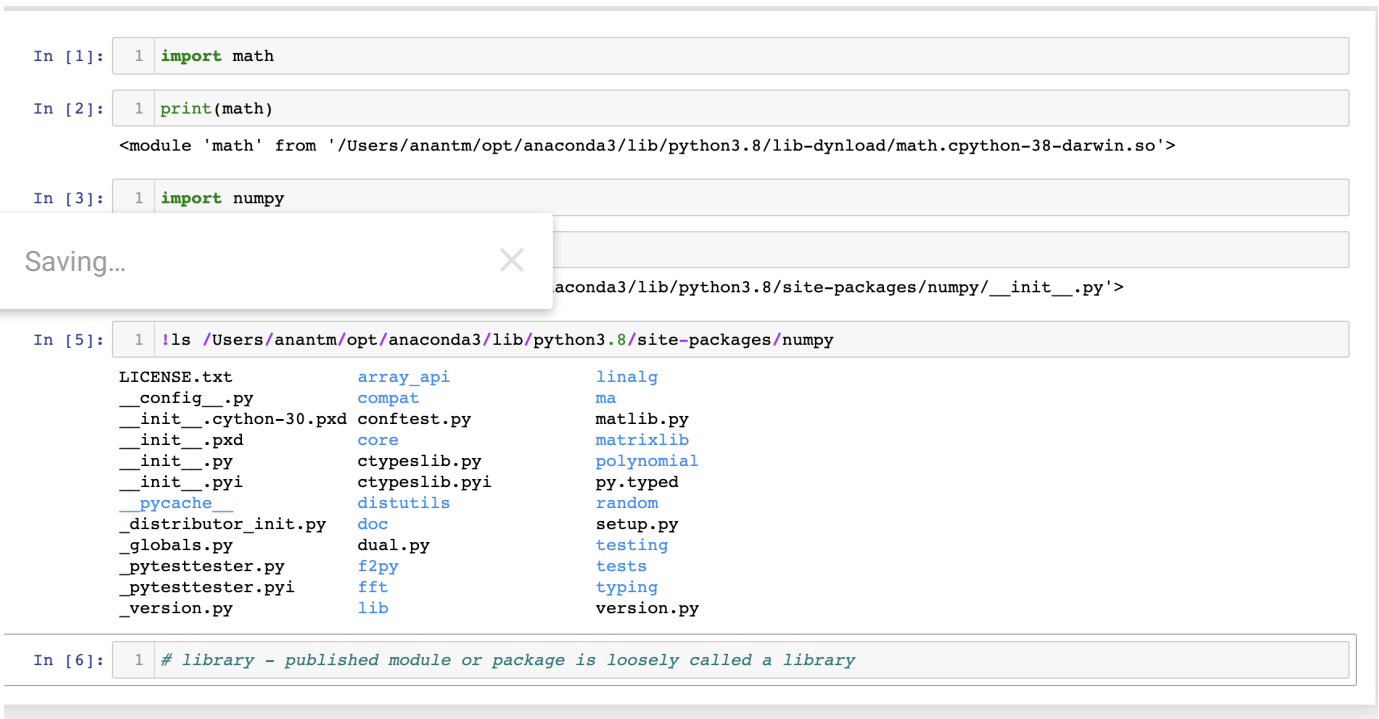


```
720
```

```
# abc.py

# def a():
#   pass

# def b():
#   pass


# xyz.py

# def x():
#   pass

# def b():
#   pass
```

```
# from abc import *
# from xyz import *
# b()

# using from ___ import * is not a good practice


# please please read the post-read


# module, package, library?


# module is simply a .py script


# package is a collection of multiple modules, sub-packages


import math


print(math)
```

```
    <module 'math' (built-in)>
```



```
In [1]: 1 import math

In [2]: 1 print(math)
        <module 'math' from '/Users/anantm/opt/anaconda3/lib/python3.8/lib-dynload/math.cpython-38-darwin.so'>

In [3]: 1 import numpy

        aconda3/lib/python3.8/site-packages/numpy/__init__.py'>

In [5]: 1 !ls /Users/anantm/opt/anaconda3/lib/python3.8/site-packages/numpy
        LICENSE.txt             array_api               linalg
        __config__.py           compat                  ma
        __init__.cython-30.pxd  conftest.py             matlib.py
        __init__.pxd            core                    matrixlib
        __init__.py             ctypeslib.py            polynomial
        __init__.pyi            ctypeslib.pyi           py.typed
        __pycache__             distutils               random
        _distributor_init.py    doc                     setup.py
        _globals.py             dual.py                 testing
        _pytesttester.py        f2py                    tests
        _pytesttester.pyi       fft                     typing
        _version.py             lib                     version.py

In [6]: 1 # library - published module or package is loosely called a library
```



```
import numpy


print(type(numpy))
```

```
    <class 'module'>
```

```
numpy.random.randint()
```

```
from math import factorial
factorial()
```

✓ 0s completed at 23:48

Saving...