Python Coding Platform

- Google Colab (Online)
- Anaconda Full version (Offline)
- IDE: VSCode, PyCharm, Spyder, Jupyter notebook

What is google colab?

- Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.
- Google Colab Setup

Variable

- dynamically typed (C variables are statically typed)
- do not need explicit declaration
- declaration happens automatically when you assign a value to a variable

```
a = 'hello world'
print(a, type(a))
a = "hello world"
print(a)
```

```
hello world <class 'str'>
hello world
```

```
a = 5
b = 10

a, b = 5, 10

print(a, 'apples')
# formatted print; similar to printf(f"{a} apples");
print(f"{a} apples")

print(f"{a} apples, {b} oranges") # 5 apples, 10 oranges
print("%2d apples, %2d oranges" % (a, b))

print(a, 'apples', b, 'oranges')
```

```
5 apples
5 apples
5 apples, 10 oranges
```

```
 5 apples, 10 oranges
5 apples 10 oranges
```

```
a = 3.5
a = a + 1 # 4.5
a = a - 2 # 2.5
a = a * 2 # 5
a = a / 2 # 2.5
a = a ** 2 # 2.5^2=6.25
a = a % 5 # 1.25 # modulus
print(a)
```

```
1.25
```

```
b = 3 ** 5 # 3^5
b = 6 % 4 # 2
print(b)
```

```
2
```

```
s = 'hello'
s = s + 'UIU'
print(s) # 'helloUIU'
s = s * 3
print(s) # 'helloUIUhelloUIUhelloUIU'
```

```
helloUIU
helloUIUhelloUIUhelloUIU
```

Typecast

```
a = '12343'
print(type(a)) # <class 'str'>

a = int(a)
print(a, type(a)) # <class 'int'>
```

```
<class 'str'>
12343 <class 'int'>
```

```
a = str(100)
print(a+a)

a = 'fariha'
x = 500
s = a + str(x)
print(s)
```

```
100100
fariha500
```

Take input

- python input is string unless you typecast

```
a = int(input())
print(a, type(a))
print(a*2)

100
100 <class 'int'>
200

x = input('Enter your name ')
print('you entered', x)

Enter your name FTI
you entered FTI

n = input()
print('input is', n)
n = n * 2
print('input * 2 is', n)

100
input is 100
input * 2 is 100100

n = float(input('Enter a number: '))
print('input is', n)
n = n * 2
print('input * 2 is', n)

Enter a number: 100
input is 100
input * 2 is 200
```

If Else

- if
- elif
- else
- NO CURLY BRACES. **INDENTATION MATTERS.**

```
a, b = 2, 33
if (a < b):
    print('hello')
    if a < 10:
        print( a, ' is less than 10')
```

```
hello
2  is less than 10
```

```
# a = 2
# b = 5
a, b = 20, 5
if a > b:
    print('a is greater than b') # yes
    if a < 10:
        print('a is less than 10') # no
    elif a < 100: # else if
        print('a is greater than 9 and less than 100') # yes
else:
    print('a is not greater than b') # no
```

```
a is greater than b
a is greater than 9 and less than 100
```

```
a = 23
b = 2456632412434312513343
if a > b:
    print(a, 'is greater than', b)
elif a == b:
    print(a, 'is equal to', b)
else:
    print(a, 'is less than', b)
```

```
23 is less than 2456632412434312513343
```

Operators

```
a = 2
if a > 1 and a <= 10: # yes
    print('1 < a <= 10')
if a == 2 or a >= 1: # yes
    print('a is greater than or equal to 1')
if a == 2: # yes
    print('a is 2')
if a != 2: # no
    print('a is not 2')

if a is 2: # yes
    print('a is 2')
if a is not 4: # yes
    print('a is not 4')
```

```
1<a<=10
a is greater than or equal to 1
a is 2
```

```
a is 2
a is not 4
```

For loop

```
# for (int i=0; i<10; i++)
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for i in range(0, 10, 2):
    print(i)
```

```
0
2
4
6
8
```

```
for i in range(0, 9, 2):
    print(i)
```

```
0
2
4
6
8
```

```
for i in range(1, 11): # 0 <= i <3
    print(i)
```

```
1
2
3
4
5
6
7
8
```

```
9
10
```

```
for i in range(0, 10):
    if i==3: break
    print(i)
print('reached end')
```

```
0
1
2
reached end
```

```
# 3, 7, 11, 15, 19
for i in range(3, 20, 4):
    print(i)
```

```
3
7
11
15
19
```

```
for i in range(0, 10, 3): # start, end, stepsize
    print(i)
```

```
0 <class 'int'>
3 <class 'int'>
6 <class 'int'>
9 <class 'int'>
```

While loop

```
n = 5/2
print(n)
```

```
2
```

```
n = 0
while True:
    print(n)
    # n++, n--, --n, ++n is not supported
    n += 1
    n = n+1
    n = n-1
    n -= 1
    n *= 2
    n /= 2
```

```
    n += 1
    if n == 5: break
```

```
0
1.0
2.0
3.0
4.0
```

```
i = 1
while i < 6:
    print(i)
    i += 2
```

```
1
3
5
```

```
i = 1
while i < 100:
    if i % 10 == 0:
        print(i)
    elif i % 23 == 0:
        print(i, 'need to break')
        break
    i += 3
```

```
10
40
46 need to break
```

List

- Heterogenous
- Multiple data type can be present in a single list

```
arr = ['apple', 'grape', 100, 3.4, [1, 2] , (2, 3)]
print(arr)
```

```
['apple', 'grape', 100, 3.4, [1, 2], (2, 3)]
```

indexing

```
print(arr[0])
print(arr[4], arr[4][0], arr[4][1])
```

```
apple
[1, 2] 1 2
```

insert

```
arr.append('orange') # inserts at the last
print(arr)
```

```
['apple', 'grape', 100, 3.4, [1, 2], 'orange']
```

```
arr.insert(1, 560)
print(arr)
```

```
['apple', 560, 'grape', 100, 3.4, [1, 2], 'orange']
```

delete

```
print(arr.pop(1)) # delete from index
print(arr)
```

```
560
['apple', 'grape', 100, 3.4, [1, 2], 'orange']
```

```
arr = ['orange', 'apple', 'orange']
arr.remove("orange")
print(arr)
```

```
['apple', 'orange']
```

search

```
if 'grape' in arr:
    print('found grape')
if 'orange' not in arr:
    print('orange not found')
```

```
found grape
orange not found
```

iterating a list

```
print(len(arr))
```

```
5
```

way1

```
for item in arr:
    print(item)
```

```
apple
grape
100
```

```
3.4
[1, 2]
```

way2

```
for i in range(0, len(arr)):
    print(arr[i])

apple
grape
100
3.4
[1, 2]

arr =  []
for i in range(2, 30, 2):
    arr.append(i)
print(arr)

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

slicing

```
arr = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

temp = arr[1:5]
print(temp)

[4, 6, 8, 10]

temp = arr[:]
print(temp)

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

print(arr)

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

temp = arr[3:7] # 8 10 12 14
temp = temp[1:3] # 10 12
print(temp)

[10, 12]

temp = arr[:4+1]
print(temp)

[2, 4, 6, 8, 10]

temp = arr[5:]
print(temp)
```

```
[12, 14, 16, 18, 20, 22, 24, 26, 28]

print(arr * 3)
print(arr + arr)

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 2, 4, 6, 8, 10,
12, 14, 16, 18, 20, 22, 24, 26, 28, 2, 4, 6, 8, 10, 12, 14, 16, 18,
20, 22, 24, 26, 28]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 2, 4, 6, 8, 10,
12, 14, 16, 18, 20, 22, 24, 26, 28]

a = []
for i in range(0, 5):
    x = int(input())
    a.append(x)

4
5
6
7
2
```

copy a list

```
foo = [1, 2, 3]
bar = foo
bar.append(4)
print(foo, bar, '\n')

foo = [1, 2, 3]
bar = foo.copy()
bar.append(5)
print(foo, bar, '\n')

[1, 2, 3, 4] [1, 2, 3, 4]

[1, 2, 3] [1, 2, 3, 5]


foo = [1, 2, []]
bar = foo
bar[-1].append(4)
print(foo, bar, '\n')

foo = [1, 2, []]
bar = foo.copy()
bar[-1].append(5)
print(foo, bar, '\n')

import copy
foo = [1, 2, []]
```

```
bar = copy.deepcopy(foo)
bar[-1].append(5)
print(foo, bar, '\n')
```

```
[1, 2, [4]] [1, 2, [4]]

[1, 2, [5]] [1, 2, [5]]

[1, 2, []] [1, 2, [5]]
```

Practice problem 1

```
## print all pair of numbers from a list

l = [1, 2, 3, 4, 5]

for i in range(len(l)):
    for j in range(i+1, len(l)):
        print(l[i], l[j])
```

```
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
4 5
```

Dictionary

```
dictionary = {
    "name" : "fariha",
    "email" : "fariha@cse.uiu.ac.bd",
    "joined" : 2019
}

d = {
    "name" : "fariha",
    "email" : 'fariha@cse.uiu.ac.bd',
    "position" : "lecturer",
    "joined" : 2019,
    3 : "???",
    1 : "what",
```

```
    "info" : [1,3, "blue"]
}

print(d)
print(d.keys())
print(d.values())
```

```
{'name': 'fariha', 'email': 'fariha@cse.uiu.ac.bd', 'position':
'lecturer', 'joined': 2019, 3: '???', 1: 'what', 'info': [1, 3,
'blue']}
dict_keys(['name', 'email', 'position', 'joined', 3, 1, 'info'])
dict_values(['fariha', 'fariha@cse.uiu.ac.bd', 'lecturer', 2019,
'???', 'what', [1, 3, 'blue']])
```

get

```
print(d.get('joined'))
print(d['joined'])
print("name" in d)
```

```
2019
2019
True
```

iterate

```
for key in d:
    print(key, " --- ", d[key])
```

```
name  ---  fariha
email  ---  fariha@cse.uiu.ac.bd
position  ---  lecturer
joined  ---  2019
3  ---  ???
1  ---  what
info  ---  [1, 3, 'blue']
```

change value, delete key-value pair

```
print(d.pop('position'))
print(d)

d['name'] = 'FTI'
print(d)
```

```
lecturer
{'name': 'fariha', 'email': 'fariha@cse.uiu.ac.bd', 'joined': 2019, 3:
'???', 'info': [1, 3, 'blue']}
{'name': 'FTI', 'email': 'fariha@cse.uiu.ac.bd', 'joined': 2019, 3:
'???', 'info': [1, 3, 'blue']}
```

```
for x in d:
    print(x, ':', d[x])
```

```
name : FTI
email : fariha@cse.uiu.ac.bd
joined : 2019
3 : ???
info : [1, 3, '?']
```

Practice problem 2

```
## write a code that counts the frequency of each letter in a sentence

sentence = "I study in UIU"

freq = {}

for i in range(len(sentence)):
    # print(sentence[i])
    char = sentence[i]
    if char not in freq:
        freq[char] = 0
    freq[char] = freq[char]+1

for char in freq:
    print(f"{char}: {freq[char]}")
```

```
I: 2
 : 3
s: 1
t: 1
u: 1
d: 1
y: 1
i: 1
n: 1
U: 2
```

Function

```
# int sum(int a, int b){
#     return a + b;
# }

def sum(a, b):
    return a + b
```

```
print(sum(3, 4))
print(sum('a', 'b'))
```

```
7
ab
```

```
def hello(name):
    name = name.capitalize()
    print('Hello',name)

hello('fariha')

# hello(1) # error
```

```
Hello Fariha
```

```
def test(arr):
  return (arr + arr)

print(test([1,2,4,6,7]))
print(test(100))
print(test("good"))
```

```
[1, 2, 4, 6, 7, 1, 2, 4, 6, 7]
```

```
def minmax(arr):
  return min(arr), max(arr)

listt = [1, 0, 9, 8, -1]
min_, max_ = minmax(listt)
print(f'min: {min_}, max: {max_}')

a = minmax(listt)
print(a[0], a[1])
```

```
min: -1, max: 9
-1 9
```

```
def canvote(age=0):
    if age >= 18:
        print("can vote at the age", age)
        return True
    else:
        print("can't vote at the age", age)
        return False

print(canvote(132), '\n')
print(canvote(age=31), '\n')
print(canvote(), '\n')
```

```
can vote at the age 132
True
```

```
can vote at the age 31
True

can't vote at the age 0
False
```

```
def canvote(is_citizen, age=0):
    if not is_citizen:
        print("can't vote", age)
        return False
    if age >= 18:
        print("can vote at the age", age)
        return True
    else:
        print("can't vote at the age", age)
        return False

print(canvote(True, 132), '\n')
print(canvote(False, age=31), '\n')
print(canvote(), '\n') # error: missing 1 argument
# print(canvote(is_citizen=False, 31), '\n') # error: positional
argument follows keyword argument
```

```
can vote at the age 132
True

can't vote 31
False
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-42-35610293c841> in <module>()
     12 print(canvote(True, 132), '\n')
     13 print(canvote(False, age=31), '\n')
---> 14 print(canvote(), '\n') # error: missing 1 argument
     15 # print(canvote(is_citizen=False, 31), '\n') # error:
positional argument follows keyword argument

TypeError: canvote() missing 1 required positional argument:
'is_citizen'
```

Practice problem 3

```
# calculate the distance between two points
import math

def distance(x1, y1, x2, y2):
    return math.sqrt((x1-x2)**2+(y1-y2)**2)

distance(1, 1, 4, 5)
```

```
5.0
```

Class in python

```
print('Hello')

class Employee:
  emp_num = 0 # works like static variable common to all objects of a
class

  def __init__(self, first, last, pay): # constructor
    self.first = first # member variable
    self.last = last # member variable
    self.email = first + '.' + last + '@email.com' # member variable
    self.pay = pay # member variable
    apple = 1 # not member variable
    Employee.emp_num += 1 # static variable

  def fullname(self): # self is comparable to this keyword of Java
    return f'{self.first} {self.last}'# "%s %s", self.first, self.last

  def applyRaise(self, raise_):
    self.pay = int(self.pay * raise_)

  def __str__(self): # toString()
    return f'{self.fullname()} - {self.email}: {self.pay}'

  @staticmethod # (Decorator) same working procedure for all objects
of a class
  def is_holiday(day):
    holidays = [1, 30, 49, 350]
    if day in holidays:
        return True
    else:
      return False


print(Employee.emp_num) # 0
```

```
emp_1 = Employee('Corey', 'Schafer', 50000)
emp_2 = Employee('Test', 'Employee', 60000)


print(Employee.emp_num) # 2

print(emp_1.fullname()) # Corey Schafer

emp_1.applyRaise(1.5)
print(emp_1.pay) # 75000

print(Employee.is_holiday(30)) # True
print(Employee.is_holiday(31)) # False

print(emp_1) # Corey Schafer corey.schafer@gmail.com 75000

class Employee:
  emp_num = 0
  def __init__(self, first, last, pay):
    self.first = first
    self.last = last
    self.email = first + '.' + last + '@email.com'
    self.pay = pay
    Employee.emp_num += 1

  def fullname(self):
    return f'{self.first} {self.last}'

  def applyRaise(self,raise_):
    self.pay = int(self.pay * raise_)


class Developer(Employee):
    def __init__(self, first, last, pay, prog_lang):
        super().__init__(first, last, pay) # call parent class
constructor
        self.prog_lang = prog_lang


class Manager(Employee):
  def __init__(self, first, last, pay, employees=None):
      super().__init__(first, last, pay)
      if employees is None:
          self.employees = []
      else:
          self.employees = employees

  def add_emp(self, emp):
      if emp not in self.employees:
          self.employees.append(emp)
```

```
  def remove_emp(self, emp):
      if emp in self.employees:
          self.employees.remove(emp)

  def print_emps(self):
      for emp in self.employees:
          print('-->', emp.fullname())

dev_1 = Developer('Corey', 'Schafer', 50000, 'Python')
dev_2 = Developer('Test', 'Employee', 60000, 'Java')

mgr_1 = Manager('Sue', 'Smith', 90000, [dev_1])
print(mgr_1.email)

mgr_1.add_emp(dev_2)
mgr_1.print_emps()
print()

mgr_1.remove_emp(dev_2)
mgr_1.print_emps()
print()

print(isinstance(mgr_1, Manager))  # Is mgr_1 an instance of Manager?
print(isinstance(mgr_1, Employee))
print(isinstance(mgr_1, Developer))
print()

print(issubclass(Developer, Employee)) # Is Developer class a subclass
of Employee class??
```

```
Sue.Smith@email.com
--> Corey Schafer
--> Test Employee

--> Corey Schafer

True
True
False

True
```

```
print('hi')
def main():
    print('hello world')

if __name__=='__main__':
    main()
```

```
hi
hello world
```

Does Python support function overloading?

Python does not support function overloading. When we define multiple functions with the same name, the later one always overrides the prior and thus, in the namespace, there will always be a single entry against each function name

- https://www.geeksforgeeks.org/python-method-overloading/

Practice problem 4

```
# write a class Point to store the x,y coords of 2D points.
# The class should have the methods move_left, move_right, move_up,
move down.
# move_left moves the point 1 cell to the left, move_right moves the
point 1 cell to the right
# move_up moves the point 1 cell up, and move_down moves the point 1
cell down
# write a distance function to calculate the manhattan distance
between two points

class Point:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def move_left(self):
        self.x += 1

    def move_right(self):
        self.x -= 1

    def move_up(self):
        self.y += 1

    def move_down(self):
        self.y -= 1

    def distance(self, p):
        return Point.euclid_dist(self, p)

    @staticmethod
    def euclid_dist(p1, p2):
        return ((p1.x-p2.x)**2+(p1.y-p2.y)**2) ** 0.5

p1 = Point(2, 4)
p2 = Point(0, 0)
print('distance %.2f'%Point.euclid_dist(p1,p2))
```

```
p2.move_down()
print('distance %.2f'%Point.euclid_dist(p1,p2))
```

Numpy array

- stores and manipulates n-dimensional arrays
- stores values of same type

```
# numpy tutorial - http://cs231n.github.io/python-numpy-tutorial/#numpy-arrays
# numpy uses the facility of GPU. So, try to use it instead of for loop
import numpy as np

import numpy as np

a = [3, 6, 7]
print(type(a))
arr = np.array(a)
print(type(arr))

print(arr.shape)

<class 'list'>
<class 'numpy.ndarray'>
(3,)

a = [
     [1, 4],
     [3, 5]
]
a = np.array(a)
print(a.shape)

print(a[1][1], a[1, 1])

(2, 2)
5 5

a = np.array([3, 5, 4])
print(type(a))
print(a.shape)
print(a[0])
print(a[10]) # index out of bound

<class 'numpy.ndarray'>
(3,)
3
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-32-faf308b02608> in <module>()
      3 print(a.shape)
      4 print(a[0])
----> 5 print(a[10]) # index out of bound

IndexError: index 10 is out of bounds for axis 0 with size 3
```

```
b = np.array([
              [1,2,3],
              [4,5,6]
              ])
print(b.shape)
print(b[0][0], b[0, 1], b[1, 0])
```

```
(2, 3)
1 2 4
```

```
a = np.random.random((4, 5))
print(a)
a = np.random.randint(0, 10, (4, 5))
# a = np.random.randint(low=0, high=10, size=(4, 5))
print(a)

# a = [
#      []
#      []
#      []
# ]
```

```
[[0.43540843 0.65050988 0.17429344 0.962465   0.0901969 ]
 [0.26805427 0.20619395 0.7396871  0.74612016 0.37360491]
 [0.59078922 0.65314588 0.3933885  0.1486195  0.94967011]
 [0.97790219 0.72027091 0.29309308 0.09829989 0.4278171 ]]
[[6 9 1 3 4]
 [1 5 1 8 5]
 [4 7 9 4 5]
 [2 2 1 1 9]]
```

```
a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)               # Prints "[[ 7.  7.]
```

```
                        #          [ 7.  7.]]"

d = np.eye(2)          # Create a 2x2 identity matrix
print(d)               # Prints "[[ 1.  0.]
                       #          [ 0.  1.]]"

a = np.random.random((3,4))
print(a)
a = np.random.randint(0,10,(3,4))
print(a)
```

```
[[2.20368888e-01 4.00672876e-01 6.95204434e-01 3.00053004e-01]
 [3.65197281e-01 2.14527247e-04 1.67528770e-01 6.61755272e-01]
 [9.48512594e-01 1.17165515e-01 6.21485142e-01 8.06927375e-01]]
[[7 2 6 6]
 [3 3 9 9]
 [2 9 0 1]]
```

slicing

```
a = np.array([
              [1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 0, 1, 2]
              ])

print(a[1:3,:], '\n')
print(a[1:3,2:], '\n')
print(a[1:3,2:3], '\n')
```

```
[[5 6 7 8]
 [9 0 1 2]]

[[7 8]
 [1 2]]

[[7]
 [1]]
```

datatype

```
# you can explicitly specify data type
# basic data types https://numpy.org/doc/stable/user/basics.types.html
x = np.array([[1,2],[3,4]], dtype=np.float32)
y = np.array([[5,6],[7,8]], dtype=np.float64)

a = np.array()
print(a)
```

```
[3 4]
```

```
# input 2D array
arr = []
for i in range(0,1):
    arr.append([])
    for j in range(0,2):
        x = int(input())
        arr[i].append(x)

arr = np.array(arr)
print('\n', arr)
```

operations: add, mult, div etc.

```
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

print(x + y)
print(np.add(x, y)) #time complexity O(1) if we use GPU

print( x+100 )
print((x+100) / 2)
```

```
[[ 6  8]
 [10 12]]
[[ 6  8]
 [10 12]]
[[101 102]
 [103 104]]
[[50.5 51. ]
 [51.5 52. ]]
```

```
z= x/y
print(z)
```

```
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

```
z = x*y # element wise multiplication
print(z)
```

```
[[ 5 12]
 [21 32]]
```

```
print(x.dot(y)) # matrix multiplication / dot product
print(np.dot(x, y))

# https://www.mathsisfun.com/algebra/matrix-multiplying.html  ->
matrix dot product
```

```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

```
x = np.array([
              [1,2,5],
              [3,4,1]
              ])

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6
6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[8 8]"
```

```
16
[4 6 6]
[8 8]
```

```
import numpy as np
x = np.array([
                [
                    [1,2,5],
                    [3,4,6]
                ],[
                    [7,8,9],
                    [10,11,12]
                ]
            ]
              )
print(x.shape)
print(x[0,1,2])
print(np.sum(x, axis=0))
print(np.sum(x, axis=1))
print(np.sum(x, axis=2))
```

```
(2, 2, 3)
6
[[ 8 10 14]
 [13 15 18]]
[[ 4  6 11]
 [17 19 21]]
[[ 8 13]
 [24 33]]
```

```
# print(np.sum(x, axis=0))
for j in range(0, 2):
    for k in range(0, 3):
        sum = 0
        for i in range(0, 2):
```

```
            sum += x[i, j, k]
        print(sum, end=' ')
    print()
```

```
8 10 14
13 15 18
```

```
x = np.array([
              [1,2,5],
              [3,4,1]
              ])
print(x)
print(x.T)
```

```
[[1 2 5]
 [3 4 1]]
[[1 3]
 [2 4]
 [5 1]]
```

```
print(np.sqrt(x))
```

```
[[1.         1.41421356 2.23606798]
 [1.73205081 2.         1.        ]]
```

```
# conversion
a = [[1,2,3],[4,5,6]]
print(type(a))

a= np.array(a)
print(type(a))

a= a.tolist()
print(a)
print(type(a))
```

```
<class 'list'>
<class 'numpy.ndarray'>
[[1, 2, 3], [4, 5, 6]]
<class 'list'>
```

```
a = [[1,2,3],[4,5,6]]
print(a)
b = a # copy by reference
# b = a.copy() # copy by value
b[0] = 999
print(a)
```

```
[[1, 2, 3], [4, 5, 6]]
[999, [4, 5, 6]]
```

reshape

```
v = np.array([1,2,3])  # v has shape (3,)
w = np.array([4,5])    # w has shape (2,)
# print(v * w) # can't do this, need to reshape

v = np.reshape(v, (3, 1))
print(v.shape)

print(v * w)
```

```
(3, 1)
[[ 4  5]
 [ 8 10]
 [12 15]]
```

```
v = np.array([
              [1,2,3],
              [4,5,6]
              ])
w = np.array([[6,7]])
print(v.shape, w.shape)
# print(v.dot(w)) # can't do this

v = np.reshape(w, (2, 1))
print(v.shape)

print(v.dot(w))

y = np.array([
              [1,2,3],
              [4,5,6]
              ])
# print(y.dot(w)) # can't do this
print(y.shape, w.shape)
print(y.T.dot(w.T))
```

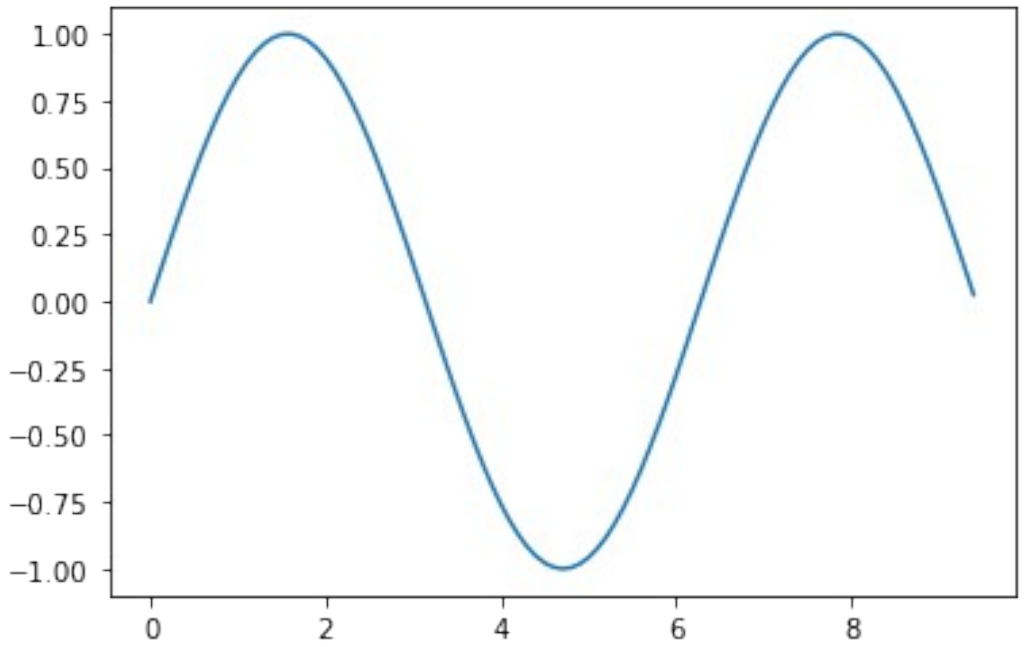
```
(2, 3) (1, 2)
(2, 1)
[[36 42]
 [42 49]]
(2, 3) (1, 2)
[[34]
 [47]
 [60]]
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
import math

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1) # [0, 0.1, 0.2, 0.3, ... 3*3.1416]
y = np.sin(x)
# y = 2*x

plt.plot(x, y)
plt.show()  # You must call plt.show() to make graphics appear
```



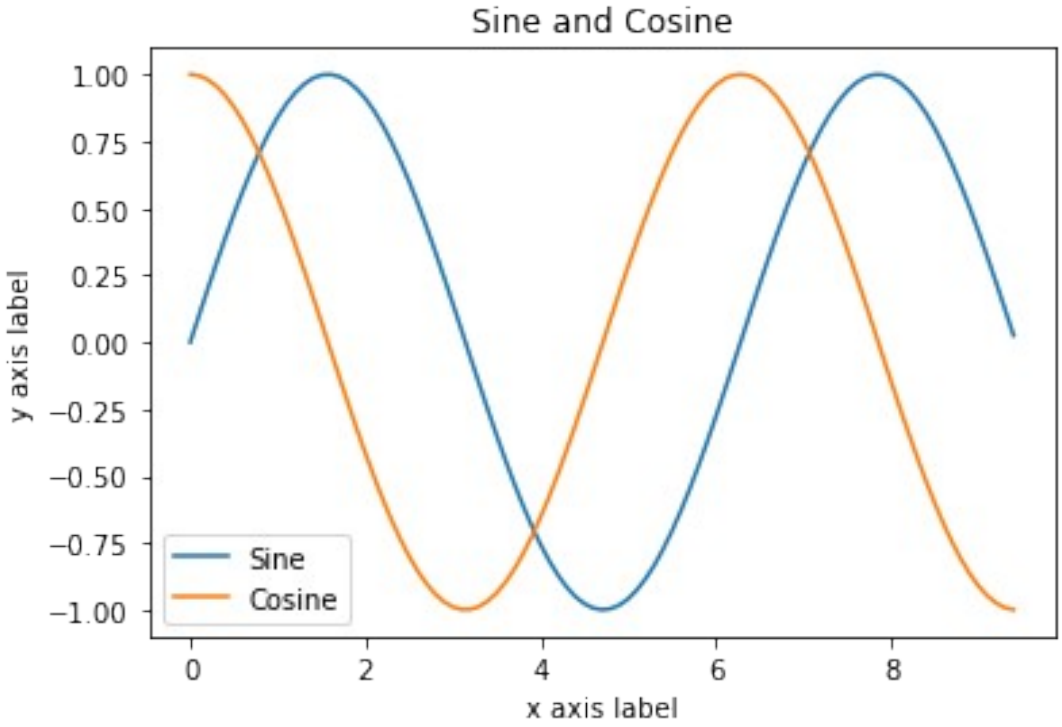

```
x = np.arange(0, 2, 0.5)
print(x)
y = 2 * x
print(y)
```

```
[0.  0.5 1.  1.5]
[0. 1. 2. 3.]
```

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
```

```
# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```



Priority Queue

```
A = [
     ['S', 10],
     ['A', 12],
     ['G', 0]
]
```