Class Notes

Colin Gibbons-Fly

Python Calculator (9/18)

```python
## Example 1
3+2*4
```

```
11
```

```python
## Example 2
(3+2)*4
```

```
20
```

```python
## Example 3
4**2
```

```
16
```

```python
## Example 4
3==2
```

```
False
```

```python
## Example 5
4<5
```

```
True
```

```
## Example 6
3>=3
```

```
True
```

```
## Example 7
3>1 or 4>2
```

```
True
```

```
## Example 8
3>1 and 4>2
```

```
True
```

Variables (9/18)

```
## Example 1
x = 3
print(x)
```

```
3
```

```
## Example 2
x = 4
print(2*x)
```

```
8
```

```
## Example 3
x = 3
y = 4
x = y
```

```
print(x,",",y)
```

```
4 , 4
```

```
## Example 4
x = 3
x = x + 4
print(x)
```

```
7
```

If/Else Statements (9/20)

```
## Example 1: Implement the absolute value

x = -6

if x > 0:
  absVal = x
else:
  absVal = -x

print(absVal)
```

```
6
```

```
## Example 2: Execute the sgn function
x = 20

if x > 0:
  sgn = 1
elif x < 0:
  sgn = -1
else:
  sgn = 0
```

```
print(sgn)
```

```
1
```

```
## Excercise 14: Implement Heaviside function H

x = 25

if x >=0:
  H = 1
else:
  H = 0

print(H)
```

```
1
```

```
## Excercise 15: Updating the Heaviside function H

x = -1

if x > 0:
  H = 1
elif x == 0:
  H = 1/2
else:
  H = 0

print(H)
```

```
0
```

Functions (9/27)

```
## Example 1: Distance Function
import math

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

dist(1,2,-5,0)
```

```
6.324555320336759
```

```
## Example 1.5: Distance Function
import math

p1=[1,2]
p2=[-5,0]

def dist(p1,p2):
  d = math.sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)
  return(d)

dist(p1,p2)
```

```
6.324555320336759
```

Functions (10/2)

```
##Excercise 18: Square function
def square(x):
  return (x**2)

square(-2)
```

```
4
```

```
## Excercise 19: Average function
def avg2(a,b):
  return (a+b)/2

avg2(2,4)
```

```
3.0
```

Lists (10/2)

```
## Example 1: List a
a = [1, 3, 6, -1, 5]
print(a[0])
print(a[1])
print(a[-1])
print(a[-2])
```

```
1
3
5
-1
```

```
## Example 2: Functions and Operators on lists
b = [1, 3, 6, -1, 5]
c = [2, 4, 7, -2, 6, 897]

len(b)
len(c)
b + c
c + c
b * 2
c * 1
3 in b
3 in c
```

```
False
```

```
## Example 3: Slicing
myList = ['cake', 'pie', 'ice cream', 'donut', 'brownie', 'fruit']

print(myList[2:5])  # Contains elements with index 2,3,4.
print(myList[:3])   # Contains elements with index 0,1,2.
print(myList[1:])   # Contains elements with index >= 1.
print(myList[2::2]) # Contains every other element with index >= 2.
print(myList[::-1]) # Reverses the list.
```

```
['ice cream', 'donut', 'brownie']
['cake', 'pie', 'ice cream']
['pie', 'ice cream', 'donut', 'brownie', 'fruit']
['ice cream', 'brownie']
['fruit', 'brownie', 'donut', 'ice cream', 'pie', 'cake']
```

```
##Example 4: Updating 2 of 2 lists

a = [1,2,3]
b = a
b[1] = -7

print(a)
print(b)
```

```
[1, -7, 3]
[1, -7, 3]
```

```
## Example 5: Updating 1 of 2 lists

a = [1,2,3]
b = a[:]
b[1] = -7

print(a)
print(b)
```

```
[1, 2, 3]
[1, -7, 3]
```

Strings (10/4)

```
##Excercise 20
a = 'foot'
b = 'ball'

print(a+b)
print(a*3)
print(b+"s")
print("base"+b+"s")
print('f' in a)
print(len(a))
print(len(a+b))
```

```
football
footfootfoot
balls
baseballs
True
4
8
```

```
## Excecise 21

myString = "Hello, world!"
testString = " "

print(myString[2:5])
print(myString[:3])
print(myString[1:])
print(myString[::-1])
print(myString[2::2])
```

```
llo
Hel
ello, world!
!dlrow ,olleH
lo ol!
```

```
## Palindrome Function - First convert string to all lower case

def pldrm(str):
  if str == str[::-1] :
    print("palindrome")
  else:
    print("not a palindrome")

pldrm("racecar")
```

```
palindrome
```

```
## Excecise 27

a = [4,5,2,1,6]

a[2:]
a[:2]
a[::2]
a[::-1]
```

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

```
## Excecise 28: Find the intials

first = "Colin"
last = "Gibbons-Fly"

def initials (first,last):
  return first[0] + last[0]

initials(first, last)
```

```
'CG'
```

```
## Excecise 28.1: Find the intials
```

```
def initials (first,last):
  return first[0] + last[0]

initials("Colin", "Gibbons-Fly")
```

```
'CG'
```

```
## Excecise 28.2: Find the intials from a user generated input
def initials():
  name = input("Please enter your name: ")
```

```
## Excercise 29: Function that takes 2 lists and returns the length of the longer list

a = [2,4,3,1]
b = [3,4]

def longer(a,b):
  if len(a) >= len(b):
    return len(a)
  else:
    return len(b)

longer(a,b)
```

```
4
```

```
## Excercise 29.1: Function that takes 2 lists and returns the length of the longer list

a = [2,4,3,1]
b = [3,4]

def longer(a,b):
    return len(a) if len(a) > len(b) else len(b)

longer(a,b)
```

```
4
```

```
## Exercise 29.2: Function that takes 2 lists and returns the length of the longer list

a = [2,4,3,1]
b = [3,4]

def longer(a,b):
  len_a = len(a)
  len_b = len(b)
  return len_a if len_a > len(b) else len_b

longer(a,b)
```

```
4
```

```
## Exercise 29.3: Function that takes 2 lists and returns the length of the longer list

a = [2,4,3,1]
b = [3,4]

def longer(a,b):
    return max(len(a),len(b))

longer(a,b)
```

```
4
```

For Loops (10/4

```
for n in [0,1,2,3]:
  print(n)
```

```
0
1
2
3
```

```
test = [0,1,2,3]

for n in test :
  print(n)
```

```
0
1
2
3
```

```
test = [0,1,2,3]

for n in test :
  print(n*100)
```

```
0
100
200
300
```

```
for n in range(0,6):
  print(n)
```

```
0
1
2
3
4
5
```

```
list = [1,2,3,4,5]
num_previous = list[-1]

for i in list:
  print(i)
```

```
1
2
3
4
5
```

Loops (10/10)

```
## Example 26

def mySum(n):
        s = 0
        for i in range(1,n+1):
                s += i
        return s

print(mySum(100))
```

```
5050
```

```
## Example 27

n = 100
s = 0
i = 1

while i <= n:
  s += i
  i += 1

print(s)
```

```
5050
```

List Comprehensions (10/11)

```
## Example 1

squares = []

for i in range(1,11):
  squares.append(i**2)
squares
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
##Example 1.1

squares = []

for i in range(1,11):
  squares += [i**2]
squares
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
## Example 1.2

[i**2 for i in range(1,11)]
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
## Example 1.2.1

[i**2 for i in range(0,11) if i >= 5]
```

```
[25, 36, 49, 64, 81, 100]
```

```
## Example 2

[i**3 for i in range(2,11) if i % 2 == 0]
```

```
[8, 64, 216, 512, 1000]
```

```
## Example 2.1

[x**3 for x in range(2,11,2)]
```

```
[8, 64, 216, 512, 1000]
```

```
## Example 3: Return only the words with an a from the "fruits" list

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
a_fruits = []

for c in fruits:
  if "a" in c:
    a_fruits.append(c)

print(a_fruits)
```

```
['apple', 'banana', 'mango']
```

```
## Example 3.1: Return only the words with an a from the "fruits" list

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
a_fruits = [c for c in fruits if "a" in c]

print(a_fruits)
```

```
['apple', 'banana', 'mango']
```

Numpy Arrays (10/11)

```
## Example 1
import numpy as np
x = np.arange(0,10)
x**2
```

```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

```
##Example 1.1

from numpy import arange

x=arange(0,10,2)
x**2
```

```
array([ 0,  4, 16, 36, 64])
```

```
import numpy as np
pi = np.pi

pi
```

```
3.141592653589793
```

Cont. Numpy Arrays (10/16)

```
from numpy import *

def g(x):
  return x**2+2*x-4

def f(x):
  return sin(x)*exp(-2*x)

x = 1.2 # float object
y = f(x) # y is float too
z = g(x)
```

```
print("x = ",x)
print("y = ",y)
print("z = ",z)
```

```
x =  1.2
y =  0.08455267826468156
z =  -0.16000000000000014
```

```
from numpy import *

def g(x):
  return x**2+2*x-4

def f(x):
  return sin(x)*exp(-2*x)

x = linspace(0, 3, 11) # an array with 11 numbers in [0,3]
y = f(x) # y is an array of length 11
z = g(x) # z is an array of length 11

print("x = ",x)
print("y = ",y)
print("z = ",z)
```

```
x =  [0.  0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3. ]
y =  [0.         0.16218493 0.17006704 0.12948307 0.08455268 0.04966235
 0.02660914 0.01294432 0.00555889 0.0019303  0.0003498 ]
z =  [-4.   -3.31 -2.44 -1.39 -0.16  1.25  2.84  4.61  6.56  8.69 11.  ]
```

```
import numpy as np

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

L
x
```

```
array([1, 2, 3, 4, 5])
```

```
np.sin(x)
```

```
array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.95892427])
```

```
## Everytime you are created a graph of a function, use linspace
np.linspace(0,1,10)
```

```
array([0.        , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
       0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.        ])
```

```
np.stack((x,x**2))
```

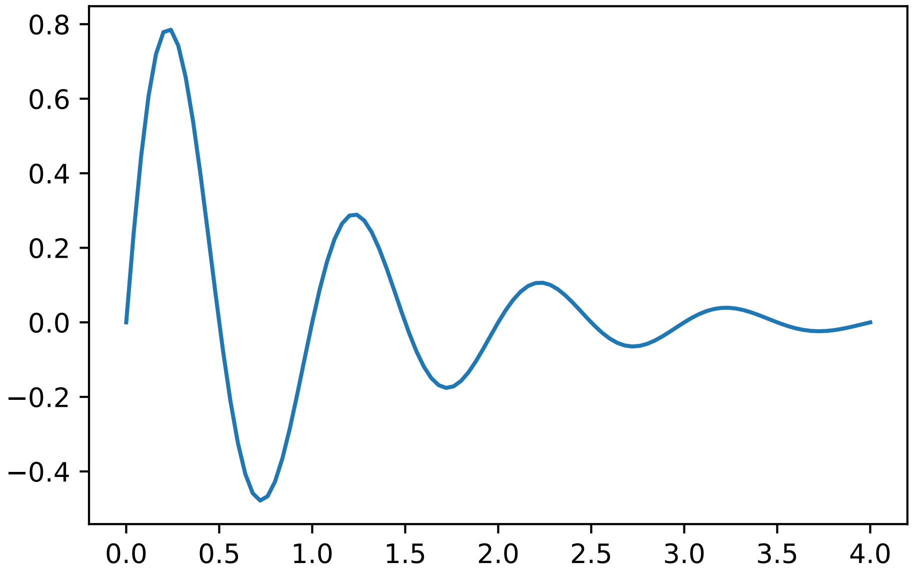
```
array([[ 1,  2,  3,  4,  5],
       [ 1,  4,  9, 16, 25]])
```

```
np.hstack((x,x**2))
```

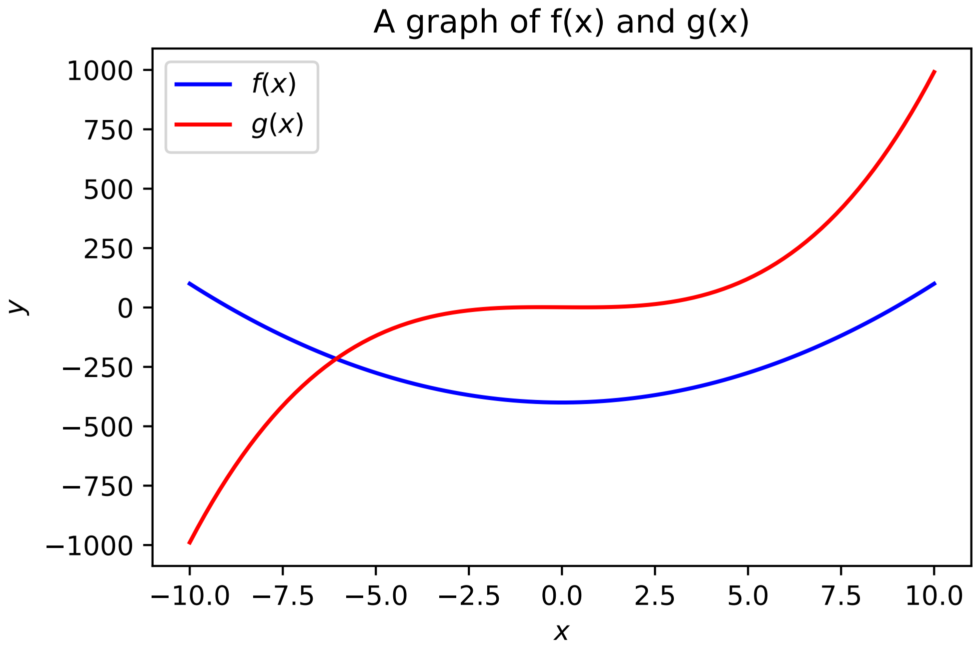
```
array([ 1,  2,  3,  4,  5,  1,  4,  9, 16, 25])
```

Matplotlib (10/16 - 10/23)

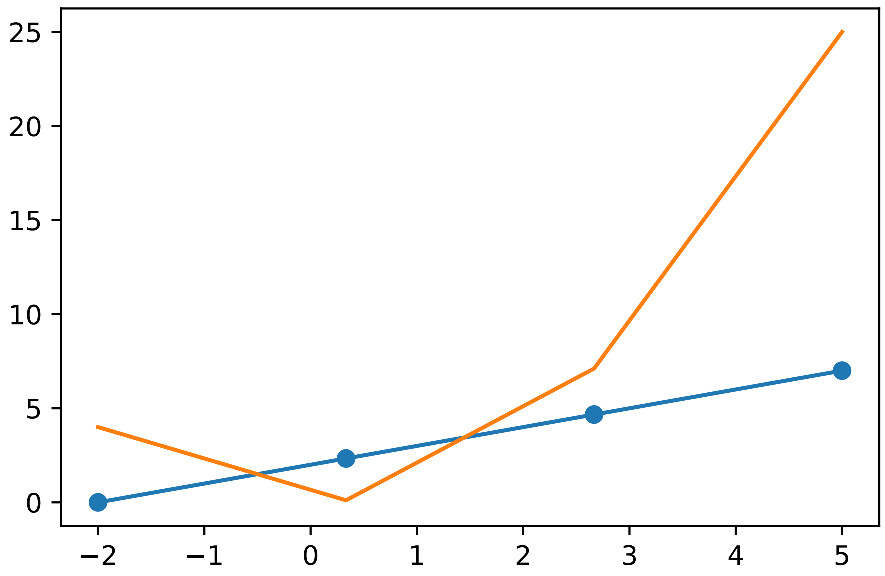
```
import numpy as np
import matplotlib.pyplot as plt

n = 100
x = linspace(0,4,n+1)
y = np.exp(-x)*np.sin(2*np.pi*x)

plt.plot(x,y)
plt.show()
```



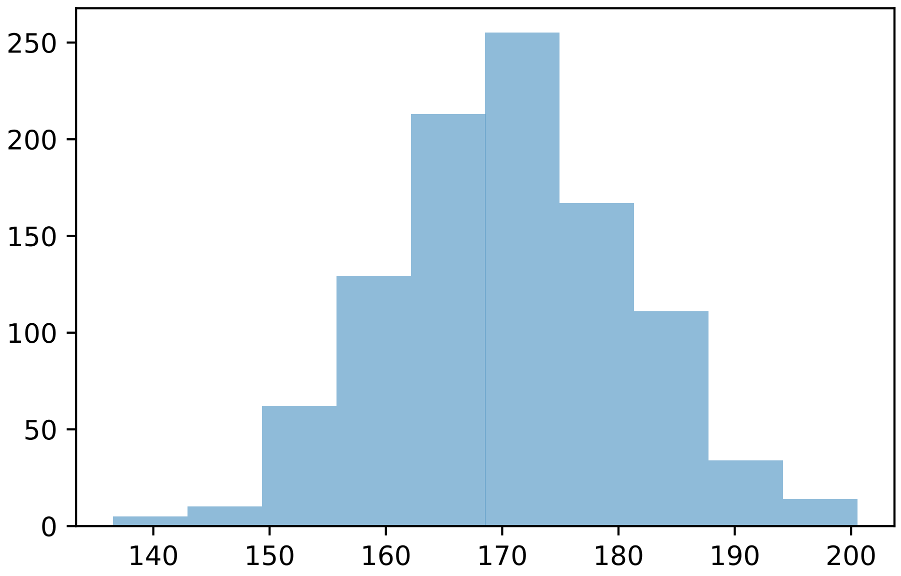
```
import numpy as np
import matplotlib.pyplot as plt

plt.clf()

x = np.linspace(-10,10,100)
y1 = 5*x**2-400
y2 = x**3-x+1

plt.plot(x,y1,color='blue',label='$f(x)$')
plt.plot(x,y2,color='red', label='$g(x)$')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend()
plt.title("A graph of f(x) and g(x)")
plt.show()
```



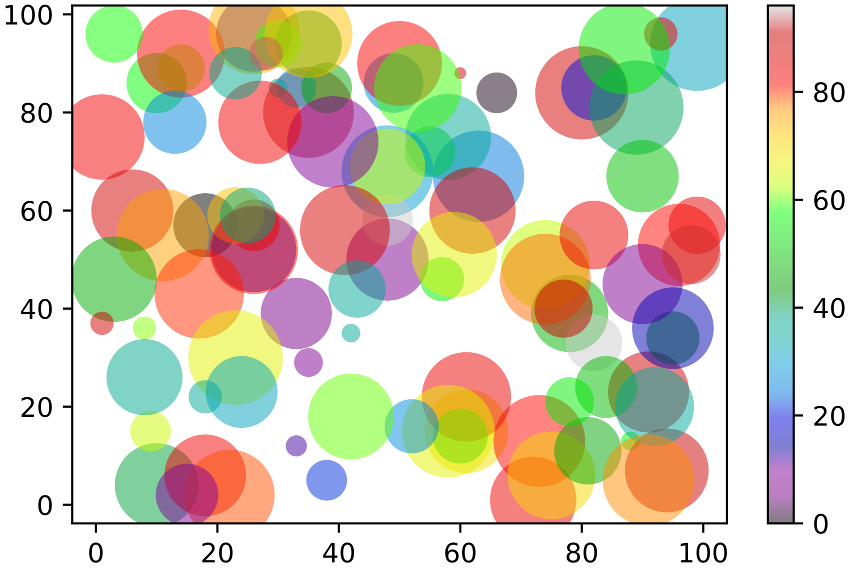
```
import matplotlib.pyplot as plt
import numpy as np

plt.clf()

x = np.linspace(-2,5,4)
y1 = x+2
y2 = x**2

plt.plot(x,y1,marker ='o')
plt.plot(x,y2)
plt.show()
```



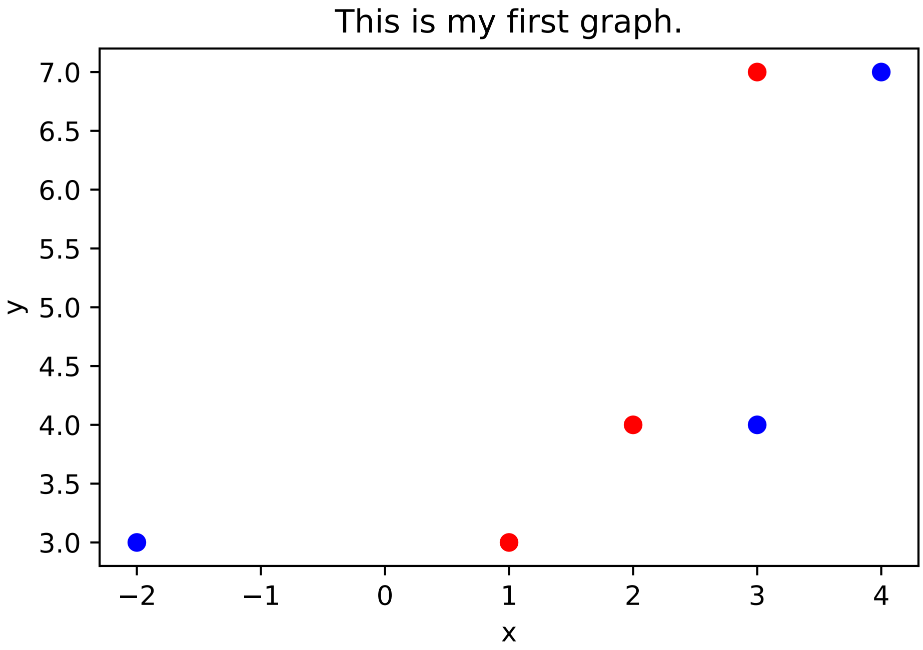
```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()

x = np.random.normal(170,10,1000)

plt.hist(x, alpha=0.5)
plt.show()
```



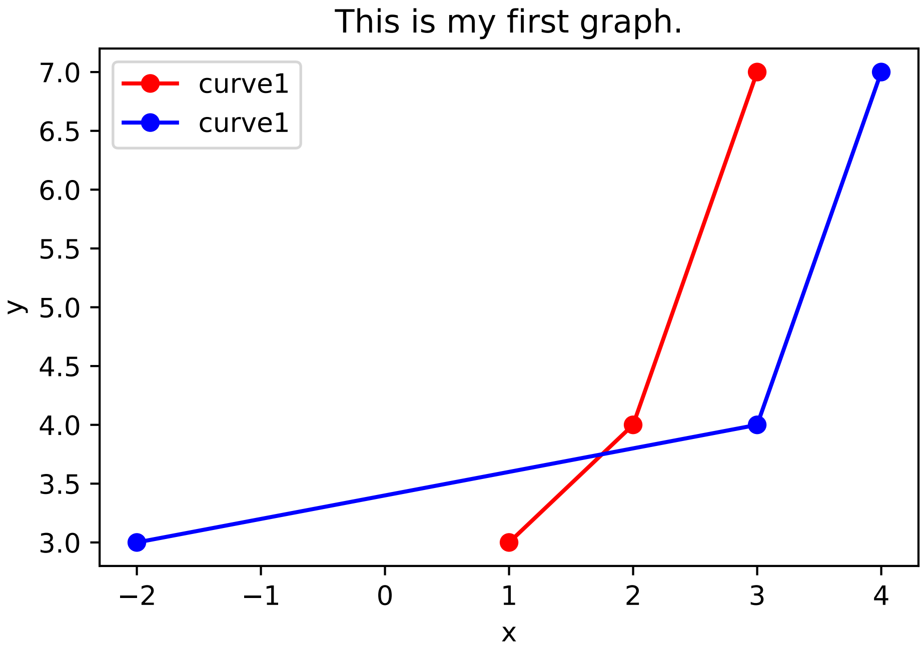
```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()

x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 12 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```



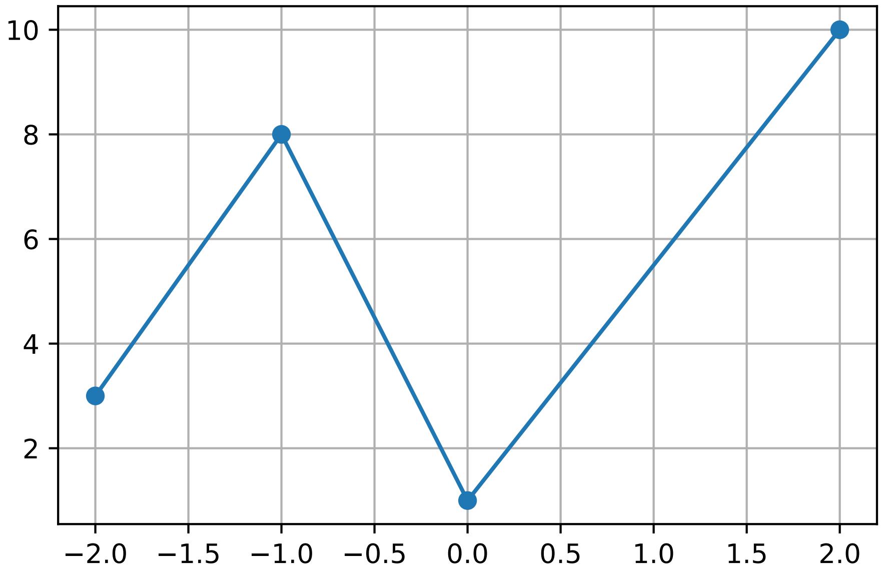
```
import matplotlib.pyplot as plt
plt.clf()

x1 = [1,2,3]
x2 = [-2,3,4]
y = [3,4,7]

plt.scatter(x1, y, color='red')
plt.scatter(x2, y, color='blue')
plt.xlabel('x')
plt.ylabel('y')
plt.title('This is my first graph.')
plt.show()
```



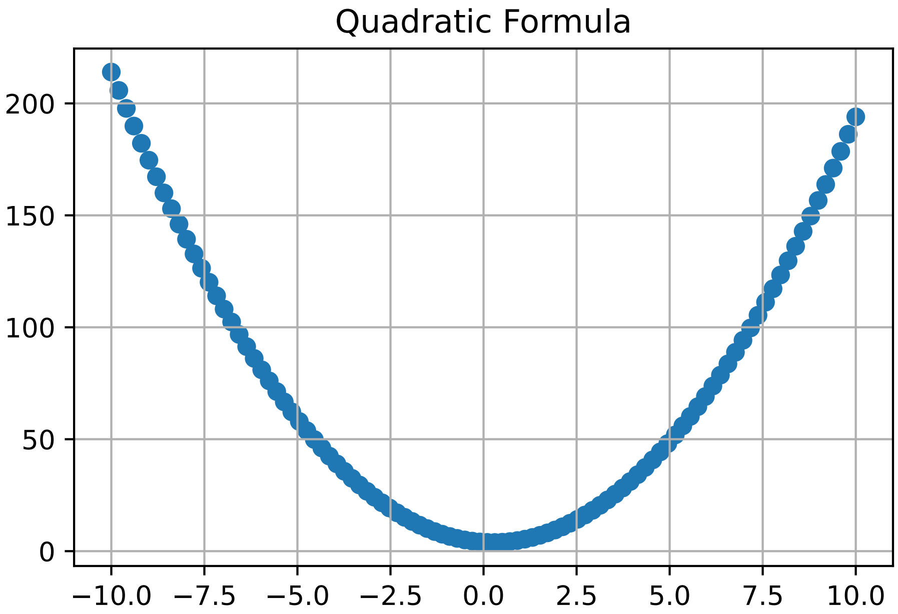
```
import matplotlib.pyplot as plt
plt.clf()

x1 = [1,2,3]
x2 = [-2,3,4]
y = [3,4,7]

plt.plot(x1, y, color='red', label='curve1', marker='o')
plt.plot(x2, y, color='blue', label='curve1', marker='o')
plt.xlabel('x')
plt.ylabel('y')
plt.title('This is my first graph.')
plt.legend()
plt.show()
```



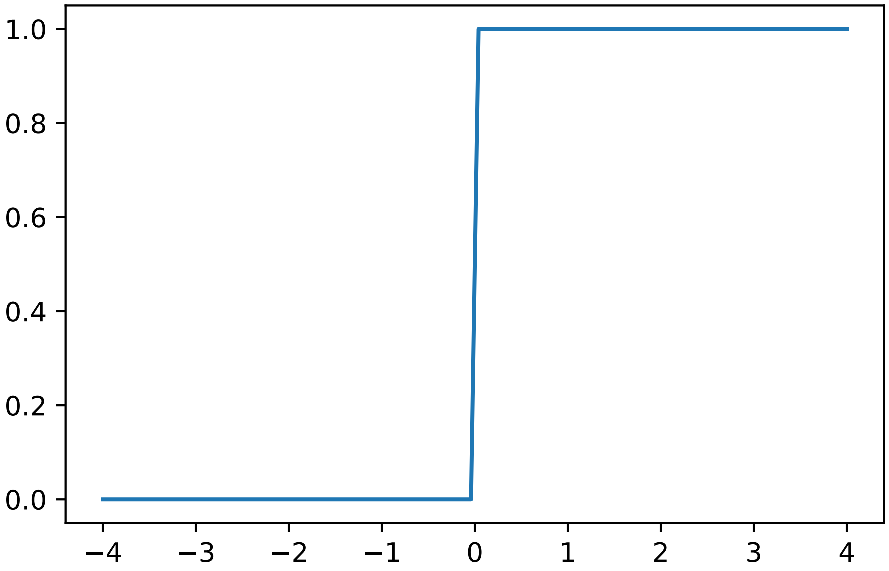
```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()

ypoints = np.array([3, 8, 1, 10])
xpoints = np.array([-2, -1, 0, 2])

plt.plot(xpoints, ypoints, marker = 'o')
plt.grid()
plt.show()
```



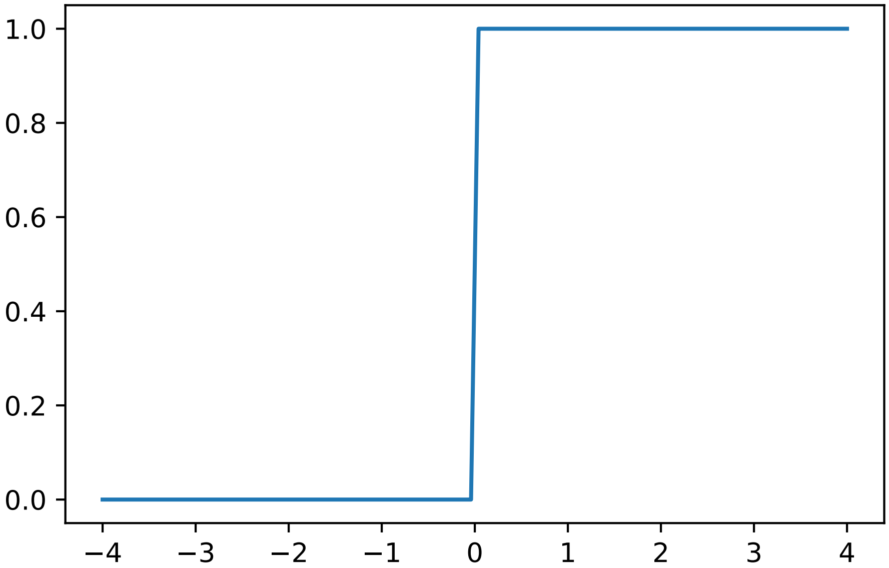
```
import numpy as np
import matplotlib.pyplot as plt
plt.clf()

def plotQuad (a,b,c):
  x = np.linspace(-10,10,100)
  y = (a*x**2)+(b*x)+c
  plt.scatter(x,y)
  plt.title("Quadratic Formula")
  plt.grid()
  plt.show()


plotQuad(2,-1,4)
```



```
## Excercise 56
import numpy as np
import matplotlib.pyplot as plt

# define a vectorized function
def H(x):
  return np.where(x>=0, 1.0, 0.0)

x = np.linspace(-4,4,100)

plt.clf()
plt.plot(x,H(x))
plt.show()
```



```
## Excercise 56.1
import numpy as np
import matplotlib.pyplot as plt
plt.clf()

def H(x):
  return np.where(x>=0, 1.0, 0.0)

x = np.linspace(-4,4,100)

plt.plot(x,H(x))
plt.show()
```



Time Value Money (11/1 - 11/6 - 11/8)

```
#Principal Interest
Pold= 1000000
#Rate
r= 3
#Timnig (1=year, 12=month, 52=week, 365=day) If monthly but for 10 years (12*10)
n=12
#Contribution per compounding period / #If making a payment(negative) / If positive(saving
c=-1000
#New Amount
Pnew = Pold*(1+(r/n))+c
print(Pnew)
```

```
1249000.0
```

```
#Time to double an investment

#initialize constant Variables
RATE = 4.5/100 #4.5% interest rate
N = 12 # number of compounding periods per year (monthly)
INITIAL_BALANCE = 100000
```

```
TARGET = 2*INITIAL_BALANCE

#initialize variables within the loop
balance = INITIAL_BALANCE
period = 0

#count years required for investment to double
while balance < TARGET:
  period += 1
  interest = balance * RATE/N
  balance = balance + interest

# print the values
print("It will take approximately",  period/12, "years")
```

```
It will take approximately 15.5 years
```

```
#Exercise 59
```

```
#Exercise 60 - A

#Intialize Variables
Pold=1000
r=0.06 ## interest rate
n=12 # compounding periods
t=20 # time in years
N=t*n #total number of compounding periods
c=0 # contribution per period

Pnew = Pold*(1+r/n)**N
print(Pnew)
```

```
3310.204475807364
```

```
#Exercise 60 - A - Loop through list, print last value of list

#Intialize Variables
Pold=1000
```

```
r=0.06 ## interest rate
n=12 # compounding periods
t=20 # time in years
N=t*n #total number of compounding periods
c=0 # contribution per period
values =[Pold]

#Run Loop
for i in range(1,N+1):
  Pold = Pold*(1+r/n)+c
  values.append(Pold)
print(values[N])
```

```
3310.2044758073625
```

```
#Excercise 60 - A - Loop to get value

P=1000

for i in range(N):
  P*=(1+r/n)
print(P)
```

```
3310.2044758073625
```

```
#Exercise 60 - B

#Intialize Variables
Pold=1000
r=0.06 ## interest rate
n=12 # compounding periods
t=20 # time in years
N=t*n #total number of compounding periods
c=50 # contribution per period

#Print Pnew
Pnew = Pold*(1+r/n)**N + (c*n/r*((1+r/n)**N - 1))
print(Pnew)
```

```
26412.249233881004
```

```
#Exercise 60 - B - Loop through list to get final value of list
import numpy as np
import matplotlib.pyplot as plt
plt.clf()

Pold=1000
r=0.06 ## interest rate
n=12 # compounding periods
t=20 # time in years
N=t*n #total number of compounding periods
c=50 # contribution per period
values =[Pold]

for i in range(1,N+1):
  Pold = Pold*(1+r/n)+c
  values.append(Pold)
print(values[N])
```

```
26412.249233881488
```

```
<Figure size 1650x1050 with 0 Axes>
```

```
#Excercise 60 - B - Loop to get value

P=1000
r=0.06 ## interest rate
n=12 # compounding periods
t=20 # time in years
N=t*n #total number of compounding periods
c=50 # contribution per period
values =[Pold]

for i in range(N):
  P=P*(1+r/n)+c
print(P)
```

```
26412.249233881488
```

```
# Excercise 61

p1= 1000 # intial investment for A
p2= 0 # initial investment for B
R = 2.5/100 # interest rate
N = 12 # compounding periods
T = 15 # timeframe
NT = n*t # total compounding periods
c1 = 0 # monthly contributions for A
c2 = 50 # monthly contributions for B

PA=p1*(1+r/n)**t
PB=(c*n)/r*((1+r/n)**t - 1)

print(PA)
print(PB)
```

```
1104.8955771867284
1048.9557718672838
```

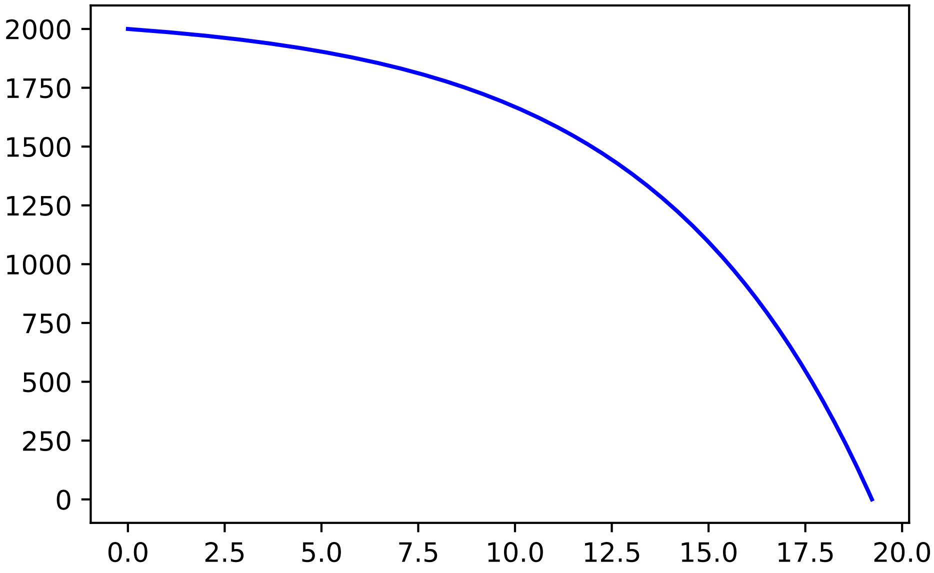
```
# Exercise 61 - Solution using a for loop
PA=1000 # initial investment for part A
PB=0.   # initial investment for part B
for i in range(t):
  PA=PA*(1+r/n)     # update the principal for part A
  PB=PB*(1+r/n) + c # update the principal for part B
# print the values
PA
PB
```

```
1048.9557718673068
```

```
# Exercise 62.1
D0 = 2000
r = 18/100
n = 12
interest = D0*r

print(interest/n)
```

```
30.0
```

```
#Exercise 62.2
import matplotlib.pyplot as plt
import numpy as np


def D(years,c):
  return D0*(1+r/n)**(years*n) - (c*n/r*((1+r/n)**(years*n) - 1))

years = np.linspace(0,19.22,200)

plt.clf()
plt.plot(years,D(years,c=31), color='blue')
plt.show()
```



```
#Exercice 62.3
c=31
D=2000
periods=0
while D>0:
  D=D*(1+r/n) - c #Calculate prinicpal balance
  periods += 1
```

```
periods/n

round(D,5)
```

```
-10.9456
```

```
#Exercise 63 - Formula
car=15000
down_payment=3000
#Loan Amount
loan=car-down_payment
#Rate
r=4.2/100
#Compounding Period
n=12
#Loan period
period=60 # number of periods to pay off the loan
#Calculate monthly payment
monthly_payment=r*loan/n*(1+r/n)**period/((1+r/n)**period - 1)

monthly_payment
total = monthly_payment * 60
total
```

```
13324.978565673213
```

```
#Exercise 63 - For Loop
#How to find the monthly payment using a loop
car=15000
down_payment=3000
#Loan Amount
loan=car-down_payment
#Rate
r=4.2/100
#Compounding Period
n=12
#Loan period
period=60 # number of periods to pay off the loan
```

```
def P(c):
  P=loan
  for months in range(t):
    P=P*(1+r/n) - c # c doesnt have to be given numerically because its in function
  return P

c=0
while P(c) > 0:
  c +=0.01

print(c,P(c))
```

```
622.2999999995791 -0.12529685220408737
```