SW-HW1-

May 20, 2020

1. Basic

Problem 1

[1]:
```
n,m=17,18
print("1. %d" % (n // 10 + m % 10))
print("2. %d" % (n % 10 + m % 2))
print("3. %d" % ((m+n) // 2))
print("4. %d" % ((m+n) / 2.0))
print("5. %d" % (int(0.5*(m+n))))
print("6. %d" % (int(round(0.5*(m+n)))))
```

```
1. 9
2. 7
3. 17
4. 17
5. 17
6. 18
```

Problem 2

[2]:
```
s,t = " Hello", " World"
print("1. %d" % (len(s) + len(t)))
print("2. %s" % (s[1] + s[2]))
print("3. %s" % (s[len(s) // 2]))
print("4. %s" % (s+t))
print("5. %s" % (t+s))
print("6. %s" % (s * 2))
```

```
1. 12
2. He
3. l
4.  Hello World
5.  World Hello
6.  Hello Hello
```

Problem 3

[3]:
```
def func(v0,g,t,d):
    h = -0.5*g*(t**2) + v0*t + d
    return h
```

```
print(func(100,32,3.2,50), ' feet')
```

```
206.15999999999997  feet
```

---

2. Datatype

Problem 1

[4]:
```
a = -11
b = 11
c = 9.0
d = b/a
e = c/a
s = 'b / a = %g' % (b/a)
print('d :', d,'e :', e,'s :', s)
```

```
d : -1.0 e : -0.8181818181818182 s : b / a = -1
```

Problem 2

[5]:
```
a = 3
print(a)
b = float(a)
print(b)
c = 3.9
print(c)
d = int(c)
print(d)
e = round(c)
print(e)
f = int(round(c))
print(f)
d = str(c)
print(d)
e = '-4.2'
print(e)
f = float(e)
print(f)
```

```
3
3.0
3.9
3
4
4
3.9
-4.2
```

```
-4.2
```

Problem 3

[6]:
```
a = 3
print(a)
b = float(a)
print(b)
c = 3.9
print(c)
d = int(c)
print(d)
e = round(c)
print(e)
f = int(round(c))
print(f)
d = str(c)
print(d)
e = '-4.2'
print(e)
f = float(e)
print(f)
```

```
3
3.0
3.9
3
4
4
3.9
-4.2
-4.2
```

Problem 4

[7]:
```
import math

def func(x):
    a = math.sinh(x)
    b = (math.e**x - math.exp(-x)) / 2
    if a==b:
         return True
    else:
         return False

print(func(math.pi))
```

```
True
```

Problem 5

[8]:
```
import math

def func(v0,c,y0,x):
    g = 9.81
    y = x*math.tan(c) - (g*x**2) / (2*v0*math.cos(c)**2) + y0
    return y

print(func(1,1,1,1))
```

```
-14.244762091441489
```

Problem 6

[9]:
```
def func(a,p,n):
    x = a * (1+p/100)**n
    return x

print(func(10000000,0.05,3))
```

```
10015007.501249997
```

---

3. Module

Problem 1 Module

area.py

[ ]:
```
def square(s):
    """get A of square with s"""
    A = s**2
    return A

def rectangle(a,b):
    """get A of rectangle with s"""
    A = a*b
    return A

from math import pi

def circle(r):
    """get A of circle with r"""
    A = pi * r**2
    return A

def triangle(b,h):
    """get A of triangle with b, h"""
    A = 0.5*b*h
    return A
```

```
def parallelogram(b,h):
    """get A of parallelogram with b,h"""
    A = b*h
    return A

from math import pi

def circular_sector(r,c):
    """get A of circular_sector with r,c"""
    A = pi * (r**2) * (c/360)
    return A

from math import pi

def circle_ring(R,r):
    """get A of circle_ring with R,r"""
    A = pi * (R**2 - r**2)
    return A

def trapezoid(h,a,b):
    """get A of trapezoid with h,a,b"""
    A = h * (a+b) / 2
    return A

def rectangular_box(a,b,c):
    """get A of rectangular_box with a,b,c"""
    A = 2*a*b + 2*b*c + 2*a*c
    return A

def cube(l):
    """get A of cube with l"""
    A=6 * (l**2)
    return A

from math import pi

def cylinder(r,h):
    """get A of cylinder with r,h"""
    A = 2 * pi * r * (r+h)
    return A

from math import pi

def right_circular_cone(r,s):
    """get A of right_circular_cone with r,s"""
    A = pi * (r**2) + math.pi * r * s
```

```
    return A

from math import pi

def sphere(r):
    """get S of sphere with r"""
    S = 4 * pi * (r**2)
    return S
```

busbar.py

[ ]:
```
from math import sqrt

def right_circular_cone(r,h):
    """get s of right_circular_cone with r,h"""
    s = sqrt((r**2) + (h**2))
    return s
```

perimeter.py

[ ]:
```
def square(s):
    '''get P of square with s'''
    return 4*s

def parallelogram(a, b):
    '''get P of parrallelogram with a, b'''
    return 2*a + 2*b

from math import pi

def circle(r):
    '''get P of circle with r'''
    return 2*pi*r

def triangle(a, b, c):
    '''get P of triangle with a,b,c'''
    return a+b+c

def rectangle(a,b):
    '''get P of rentangle with a,b'''
    return 2*(a+b)

def trapezoid(a, b, c, d):
    '''get P of trapezoid with a,b,c,d'''
    return a+b+c+d;

def circular_sector(r, seta):
    '''get P(length) of circular sector with r, seta'''
```

```
    return r * seta
```

pythagorean.py

[ ]:
```
from math import sqrt

def pythagorean_theorem(a,b):
    """get c of pythagorean_theorem with a,b"""
    c = sqrt((a**2) + (b**2))
    return c
```

volume.py

[ ]:
```
from math import pi
def sphere(r):
    '''get volume of sphere with r'''
    return 4 * pi * r ** 3 / 3

def rectangular_box(a,b,c):
    '''get volume of rectangular_box with r'''
    return a * b * c

def right_circular_cone(r, h):
    '''get volume of right_circular_cone with r, h'''
    return (1/3) * pi * r ** 2 * h

def cube(l):
    '''get volume of cube with l'''
    return l ** 3

def cylinder(r, h):
    '''get volume of cylinder with r, h'''
    return pi * r ** 2 * h

def frustum_of_a_cone(r, R, h):
    '''get volume of frustum of a cone with r, R, h'''
    return (1/3) * pi * h * (r**2 + r*R + R**2)
```

Problem 1

[2]:
```
print("Geometry Calulator\n",'### menu ###\n', 'p - perimeter\n',
     'a - area\n', 'v - volume\n', 'b - busbar\n', 'pytha - pythagorean␣
 ↪theorem\n')

user_menu = str(input())

if user_menu == 'p':
```

```
   import perimeter as p

   print("### perimeter - shape ###\nYou can type\nsquare, rectangle, circle,␣
↪triangle, parrelleogram, circular sector, trapezoid\n")

   shape = str(input())

   if shape == 'square':

       print("type - s : side")
       s = int(input("s :"))
       print(p.square(s))

   elif shape == 'rectangle':

       print("type - a : width / b : height")
       a = int(input("a :"))
       b = int(input("b :"))
       print(p.rectangle(a,b))

   elif shape == 'circle':

       print("type - r : radius")
       r = int(input("r :"))
       print(p.circle(r))

   elif shape == 'triangle':

       print("type - a : side.1 / b : side.2 / c : side.3")
       a = int(input("a :"))
       b = int(input("b :"))
       c = int(input("c :"))
       print(p.triangle(a,b,c))

   elif shape == 'parallelogram':

       print("type - a : hypotenuse / b : base")
       a = int(input("a :"))
       b = int(input("b :"))
       print(p.parallelogram(a,b))

   elif shape == 'circular sector':

       print("type - r : radius / seta :  ")
       r = int(input("r :"))
       seta = int(input())
       print(p.circular_sector(r, seta))
```

```
    elif shape == 'trapezoid':

        print("type - a : upper side / b,c : hypotenuse / d : base")
        a = int(input("a :"))
        b = int(input("b :"))
        c = int(input("c :"))
        d = int(input("d :"))
        print(p.trapezoid(a,b,c,d))

elif user_menu == 'a':

    import area as ar

    print("### area - shape ###\nYou can type\nsquare, rectangle, circle,␣
 ↪triangle, parallelogram, circular sector, circular ring, trapezoid,␣
 ↪rectangular box, right circular cone, cube, cylinder\n")

    shape = str(input())

    if shape == 'square':

        print("type - s : side")
        s = int(input("s :"))
        print(ar.square(s))

    elif shape == 'rectangle':

        print("type - a : width / b : height")
        a = int(input("a :"))
        b = int(input("b :"))
        print(ar.rectangle(a,b))

    elif shape == 'circle':

        print("type - r : radius")
        r = int(input("r :"))
        print(ar.circle(r))

    elif shape == 'triangle':

        print("type - b : base / h : height")
        b = int(input("b :"))
        h = int(input("h :"))
        print(ar.triangle(b, h))

    elif shape == 'parallelogram':
```

```
        print("type - b : base / h : height")
        b = int(input("b :"))
        h = int(input("h :"))
        print(ar.parallelogram(b, h))

    elif shape == 'circular sector':

        print("type - r : radius / seta :  ")
        r = int(input("r :"))
        seta = int(input())
        print(ar.circular_sector(r, seta))

    elif shape == 'circular ring':

        print("type - R : outside circle radius / r : inside circle radius")
        R = int(input("R :"))
        r = int(input("r :"))
        print(ar.circular_ring(R, r))

    elif shape == 'trapezoid':

        print("type - h : height / a : upper side / b : base")
        h = int(input("h :"))
        a = int(input("a :"))
        b = int(input("b :"))
        print(ar.trapezoid(h,a,b))

    elif shape == 'rectangular box':

        print("type - a : height / b : base / c : top side")
        a = int(input("a :"))
        b = int(input("b :"))
        c = int(input("c :"))
        print(ar.rectangular_box(a, b, c))

    elif shape == 'right circular cone':

        print("type - r : base circle radius / s : busbar")
        r = int(input("r :"))
        s = int(input("s :"))
        print(ar.right_circular_cone(r,s))

    elif shape == 'cube':

        print("type - l : side")
        l = int(input("l :"))
```

```
        print(ar.cube(l))

    elif shape == 'cylinder':

        print("type - r : radius / h : height")
        r = int(input("r :"))
        h = int(input("h :"))
        print(ar.cylinder(r,h))

elif user_menu == 'v':

    import volume as v

    print("### volume - shape ###\nYou can type\nsphere, rectangular box, right␣
 ↪circular cone, cube, cylinder, frustum of a cone\n")

    shape = str(input())

    if shape == 'sphere':

        print("type - r : radius")
        r = int(input("r :"))
        print(v.sphere(r))

    elif shape == 'rectangular box':

        print("type - a : height / b : base / c : top side")
        a = int(input("a :"))
        b = int(input("b :"))
        c = int(input("c :"))
        print(v.rectangular_box(a,b,c))

    elif shape == 'right circular cone':

        print("type - r : radius / h : height")
        r = int(input("r :"))
        h = int(input("h :"))
        print(v.right_circular_cone(r,h))

    elif shape == 'cube':

        print("type - l : side")
        l = int(input("l :"))
        print(v.cube(l))

    elif shape == 'cylinder':
```

```
        print("type - r : radius / h : height")
        r = int(input("r :"))
        h = int(input("h :"))
        print(v.cyliner(r,h))

    elif shape == 'frustum of a cone':

        print("type - r : upper circle radius / R : base circle radius / h :␣
 ↪height")
        r = int(input("r :"))
        R = int(input("R :"))
        h = int(input("h :"))
        print(v.frustum_of_a_cone(r,R,h))

elif user_menu == 'pytha':

    import pythagorean as pytha

    print("### pythagorean - shape ###\nYou can type\npythagorean theorem\n")

    shape = str(input())

    if shape == 'pythagorean theorem':

        print("type - a : side.1 / b : side.2")
        a = int(input("a :"))
        b = int(input("b :"))
        print(pytha.pythagorean_theorem(a,b))

elif user_menu == 'b':

    import busbar as bb

    print("### busbar - shape ###\nYou can type\nright circular cone\n")

    shape = str(input())

    if shape == 'right circular cone':

        print("type - r : radius / h : height")
        r = int(input("r :"))
        h = int(input("h :"))
        print(bb.right_circular_cone(r,h))
```

```
Geometry Calulator
 ### menu ###
 p - perimeter
```

```
 a - area
 v - volume
 b - busbar
 pytha - pythagorean theorem

a
### area - shape ###
You can type
square, rectangle, circle, triangle, parallelogram, circular sector, circular
ring, trapezoid, rectangular box, right circular cone, cube, cylinder

circle
type - r : radius
r :3
28.274333882308138
```

Problem 2 Module

getDistFunc.py

[ ]:
```python
import numpy as np
from numpy.linalg import norm

def dist_euclid(v1, v2):
    '''get Euclidean distance'''
    return np.sqrt(np.sum((v1-v2)**2))

def dist_cityblock(v1, v2):
    '''get Manhattan distance'''
    return np.sum(np.abs(v1-v2))

def dist_hamming(v1, v2):
    '''get Hamming distance'''
    dist = 0
    for i in range(len(v1)):
        if(v1[i] != v2[i]):
            dist += 1
    return dist

def dist_cosin(v1, v2):
    '''get Cosin distance'''
    return np.dot(v1, v2) / (norm(v1) * norm(v2))

def dist_jaccard(v1, v2):
    '''get Tanimoto distance'''
    union = set(v1).union(set(v2))
    intersection = set(v1).intersection(set(v2))
    return len(intersection) / len(union)
```

Problem 2

[3]:
```
import numpy as np

u = np.random.randint(10)
v = np.random.randint(10)

vector1 = np.array([u,v]) # vector => u, v

u = np.random.randint(10)
v = np.random.randint(10)

vector2 = np.array([u,v]) # vector => u, v

import getDistFunc as gdf

print(vector1, vector2)
print(gdf.dist_euclid(vector1, vector2))
print(gdf.dist_cityblock(vector1, vector2))
print(gdf.dist_hamming(vector1, vector2))
print(gdf.dist_cosin(vector1, vector2))
print(gdf.dist_jaccard(vector1, vector2))
```

```
[8 8] [0 1]
10.63014581273465
15
2
0.7071067811865475
0.0
```

---

4. Lists

Problem 1

[4]:
```
def getArea(dots, point):
    P = 0
    for n in range(point):
        x = dots[n]
        for m in range(n+1,point):
            y = dots[m]
            P = P + abs(x[0]*y[1] - y[0]*x[1])
    P = P/2
    return P

dots = [[1,1],[2,4],[3,1],[3.5,5],[4,2.5]]
point = 5
print(getArea(dots,point))
```

29.125

Problem 2

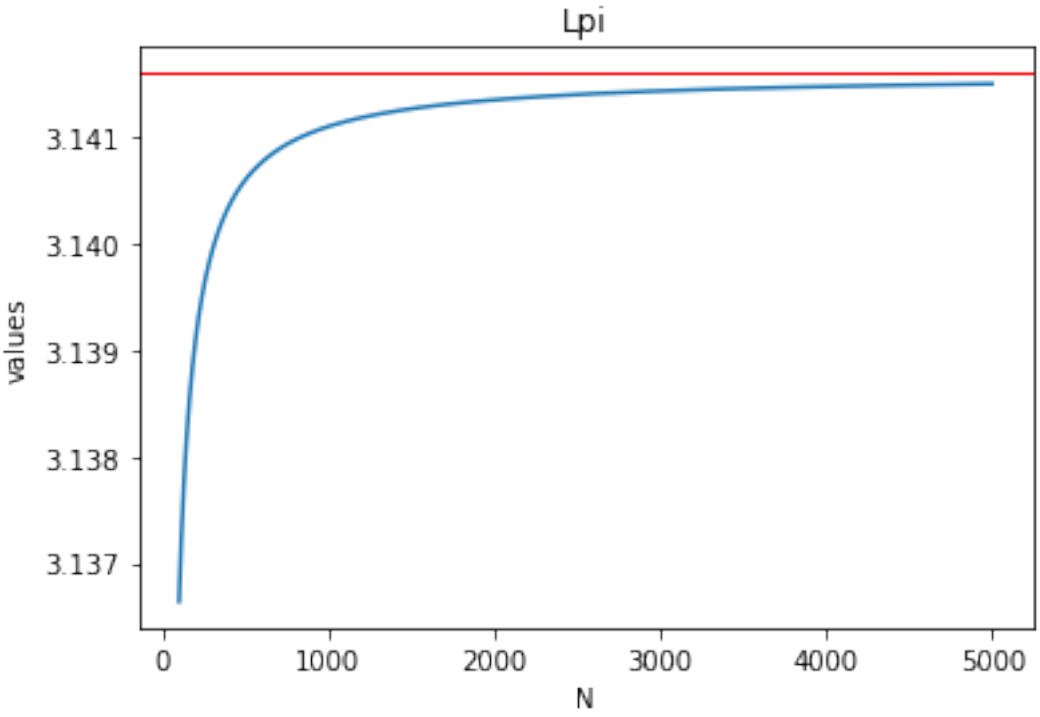
[1]:
```
import numpy as np

def f(x):
    return x

def g(x):
    return x**2

N = int(input("N: "))
E = float(input("E: "))

interval = 8 / N # -4 ~ 4  N    divide
x = np.arange(-4, 4+interval, interval) # x   : -4~4

point = [] # E       point    list

def CalcPoint(x):
    for _ in x:
        val = f(_) - g(_)
        if(np.abs(val) < E):
            point.append(_)
    return

CalcPoint(x)
print("       :",point)
```

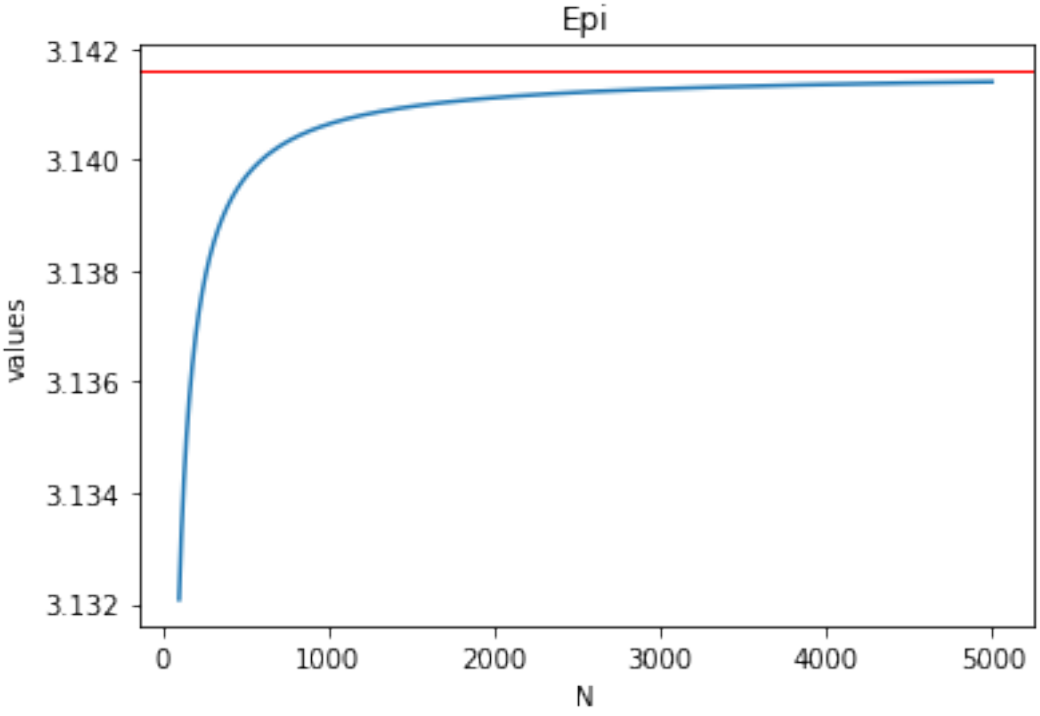
```
N: 400
E: 0.01
       : [3.552713678800501e-15, 1.0000000000000044]
```

Problem 3

[2]:
```
def getLpi(N):
    Lpi = 0
    for k in range(N + 1):
        Lpi = Lpi + (1/((4*k + 1)*(4*k + 3)))
    Lpi = Lpi*8
    return Lpi

from math import sqrt
def getEpi(N):
    Epi = 0
    for k in range(1, N + 1):
        Epi = Epi + (1 / (k**2))
    Epi = sqrt(6*Epi)
```

```
    return Epi


import pylab as pl
import numpy as np
import math

t = range(100,5000)
s=[]
for i in t:
    s.append(getLpi(i))
pl.axhline(y = math.pi, color = 'r', linewidth = 1)
pl.plot(t,s)
pl.title("Lpi")
pl.xlabel('N')
pl.ylabel('values')
pl.show()

t = range(100,5000)
s=[]
for i in t:
    s.append(getEpi(i))
pl.axhline(y = math.pi, color = 'r', linewidth = 1)
pl.plot(t,s)
pl.title("Epi")
pl.xlabel('N')
pl.ylabel('values')
pl.show()
```


Lpi
3.141
3.140
3.139
3.138
3.137
values
0
1000
2000
3000
4000
5000
N

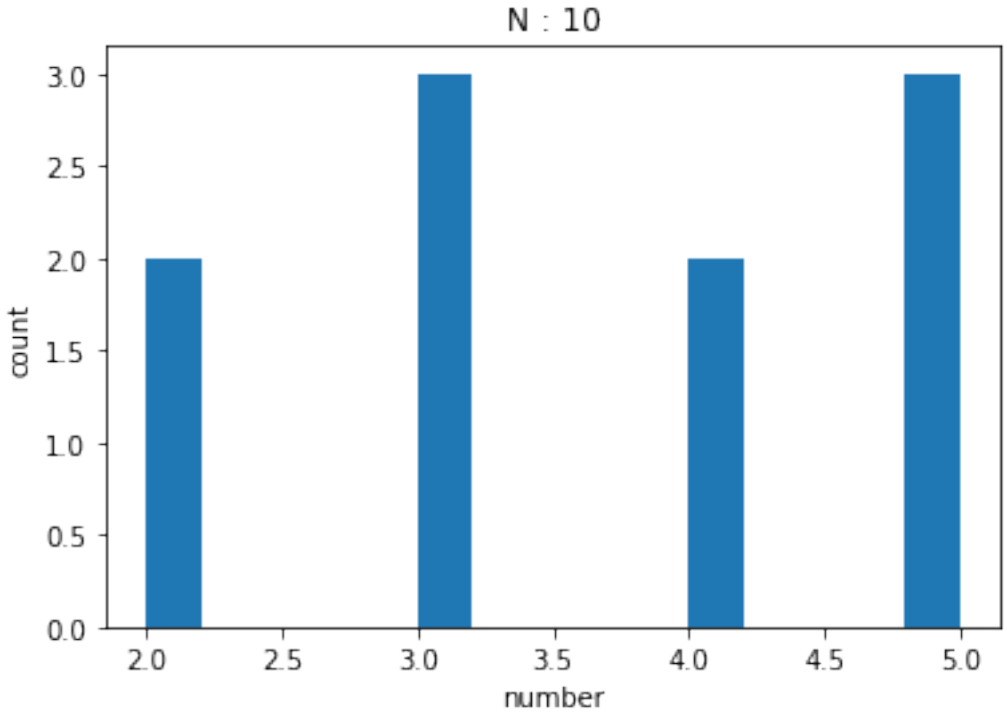

Epi
3.142
3.140
3.138
3.136
3.134
3.132
values
0
1000
2000
3000
4000
5000
N

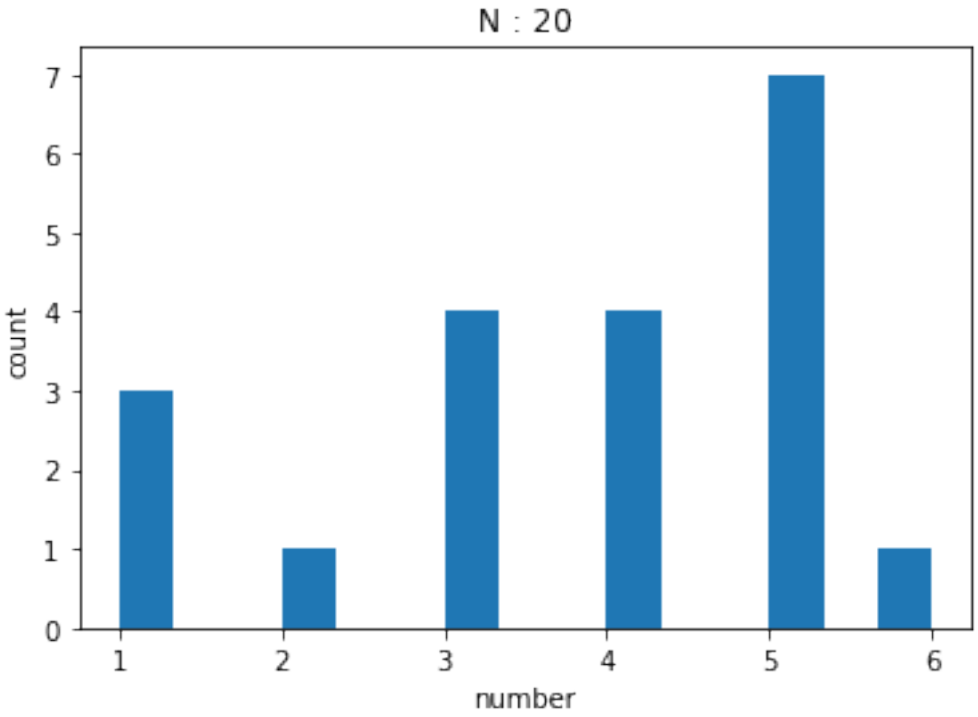
5. Loop Statements

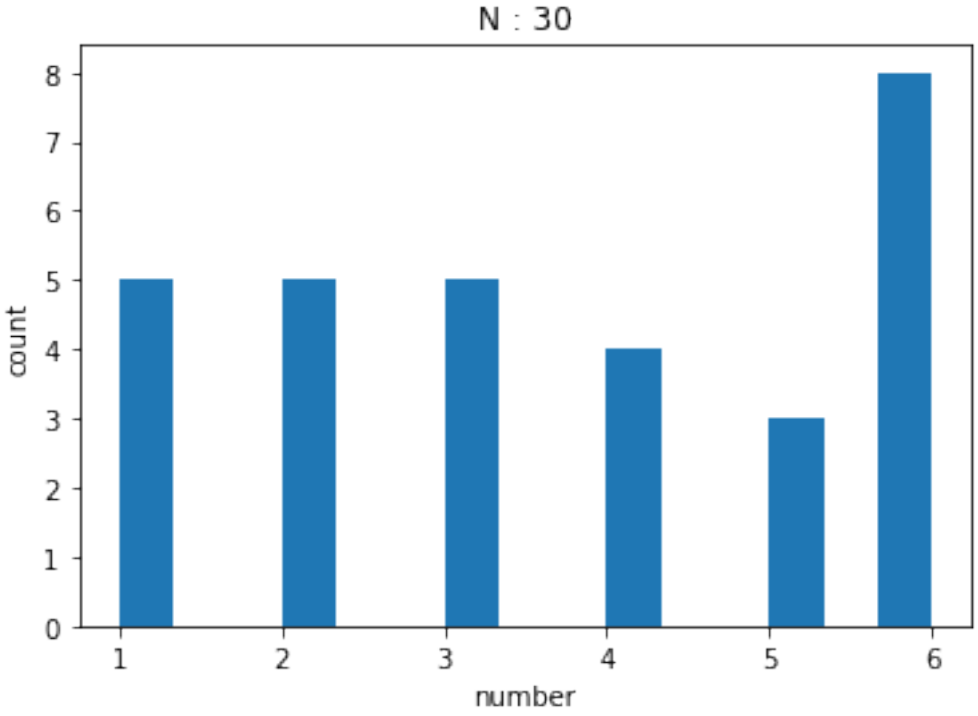
Problem 1

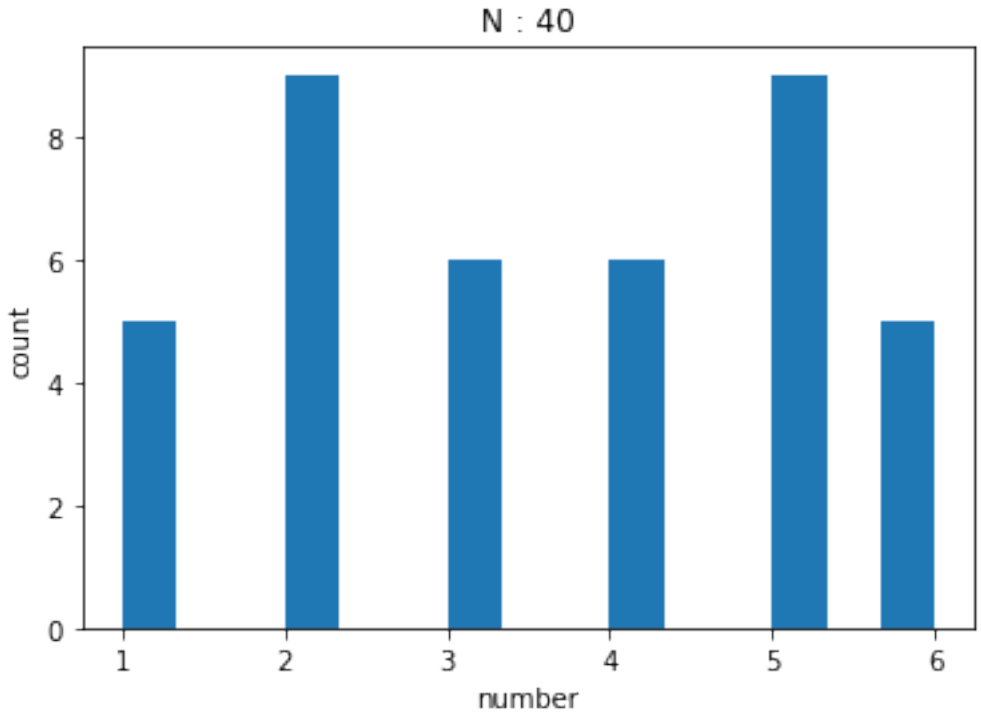
[3]:
```
import pylab as pl
import random

def getRand(N):
    num = []
    for i in range(N):
        ran = random.randint(1,6)
        num.append(ran)
    return num

temp = []
for i in range(10,100+1,10):
    temp.append(i)
x = []
for i in temp:
    x.append(getRand(i))
pl.hist(x[0], bins = 15, label = '10')
pl.title("N : 10")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[1], bins = 15, label = '10')
pl.title("N : 20")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[2], bins = 15, label = '10')
pl.title("N : 30")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[3], bins = 15, label = '10')
pl.title("N : 40")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[4], bins = 15, label = '10')
pl.title("N : 50")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[5], bins = 15, label = '10')
pl.title("N : 60")
```

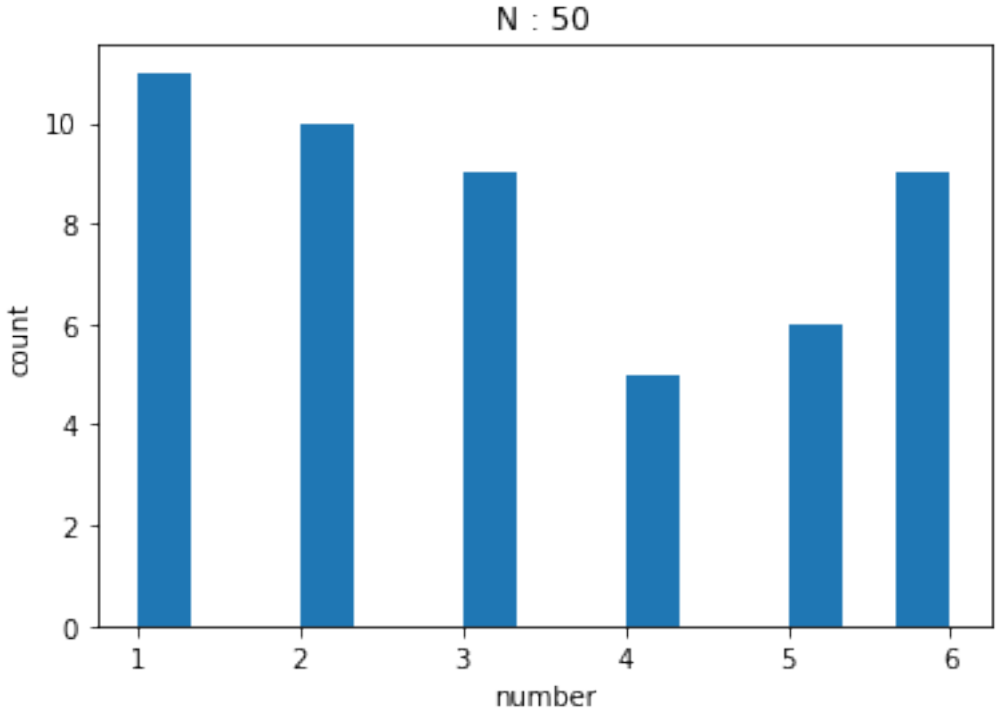
```
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[6], bins = 15, label = '10')
pl.title("N : 70")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[7], bins = 15, label = '10')
pl.title("N : 80")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[8], bins = 15, label = '10')
pl.title("N : 90")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
pl.hist(x[9], bins = 15, label = '10')
pl.title("N : 100")
pl.xlabel("number")
pl.ylabel("count")
pl.show()
```

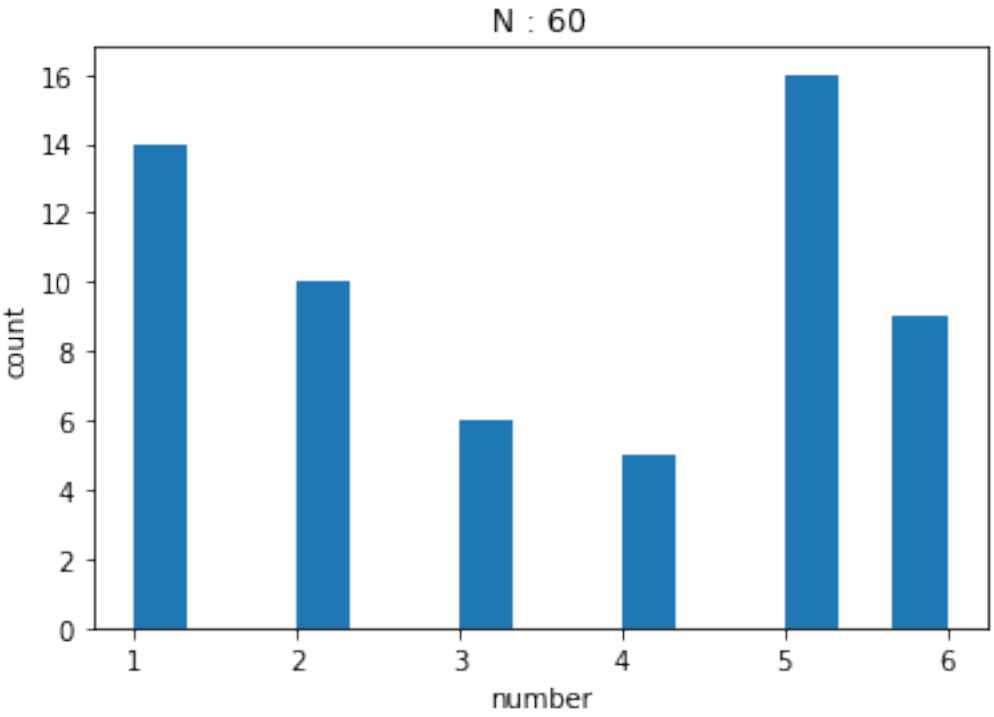


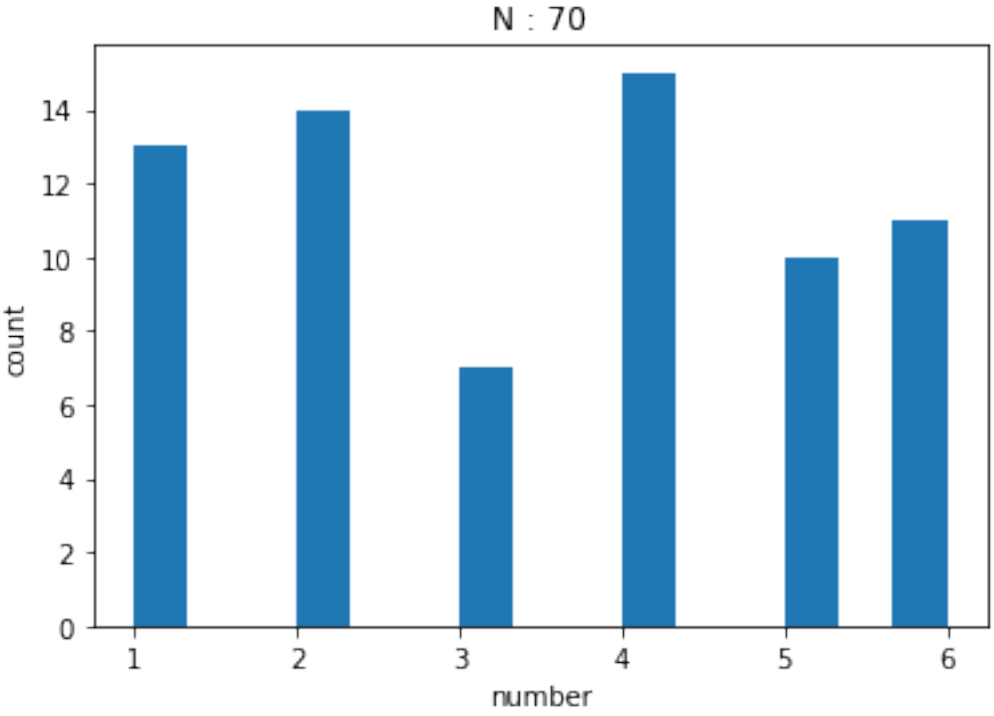

N : 20
count
number
0
1
2
3
4
5
6
7
1
2
3
4
5
6

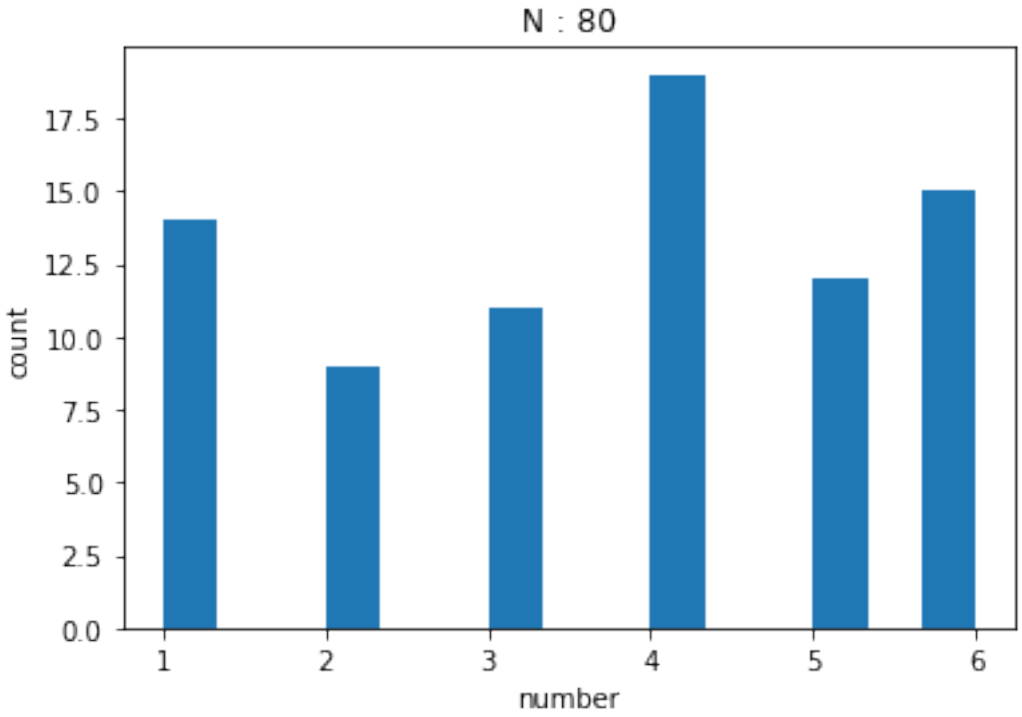

N : 30
count
number
0
1
2
3
4
5
6
7
8
1
2
3
4
5
6

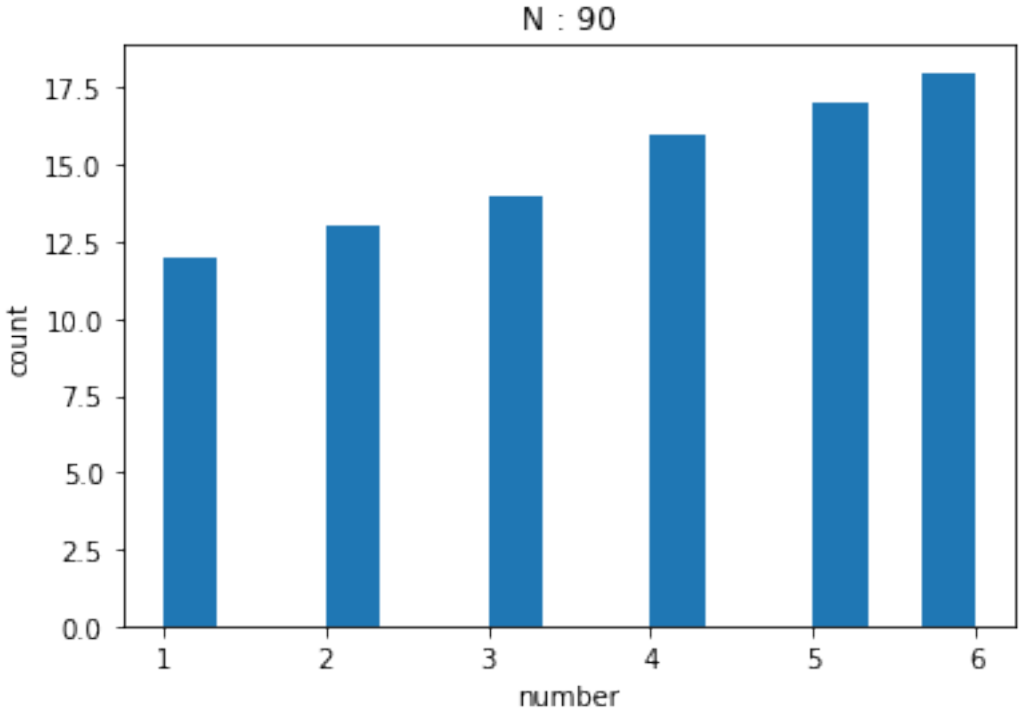

N : 40
count
number
0
2
4
6
8
1
2
3
4
5
6

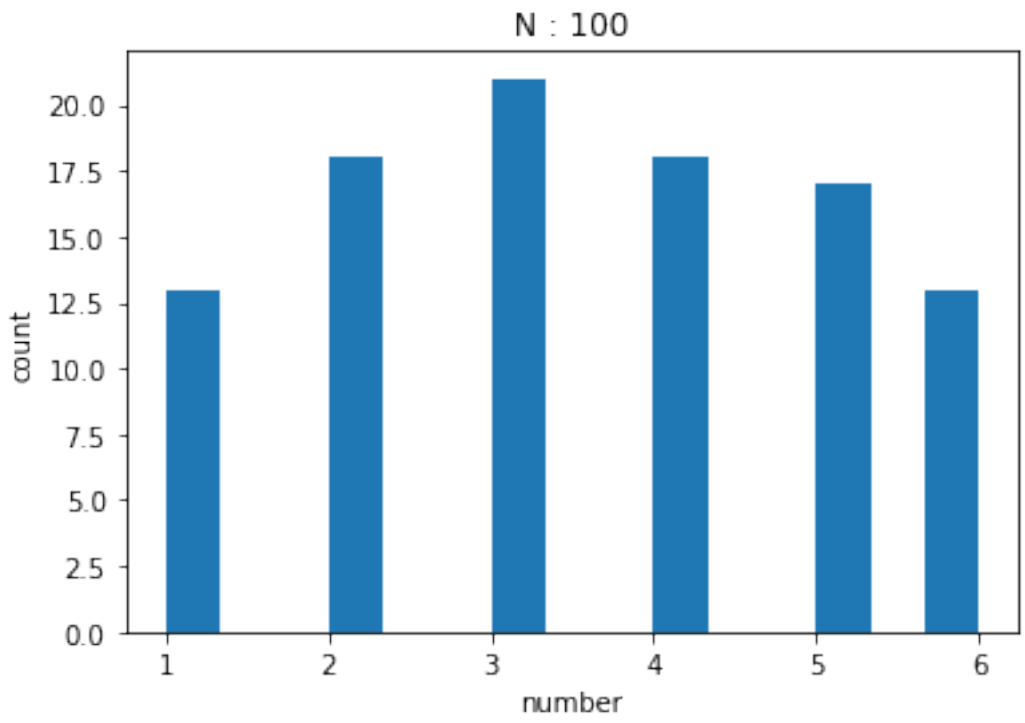

N : 50
count
number
0
2
4
6
8
10
1
2
3
4
5
6


N : 60
count
number
0
2
4
6
8
10
12
14
16
1
2
3
4
5
6


N : 70
count
number
0
2
4
6
8
10
12
14
1
2
3
4
5
6


N : 80
count
number
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
1
2
3
4
5
6


N : 90
count
number
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
1
2
3
4
5
6



Problem 2

[4]:
```
import random
import sys

com = random.randint(1,10) + random.randint(1,10)
user = random.randint(1,10) + random.randint(1,10)

while(1):
    com = com + random.randint(1,10)
    if(com > 16):
        if(com > 21):
            print("Com over 21. You win!!")
            sys.exit()
        break

while(1):
    print("user : %d" %user)
    ans = input("Do you want more card? (y/n)")
    if(ans == 'n'):
        if(user > com):
            print("user : %d" %user)
            print("com : %d" %com)
            print("You win!!")
```

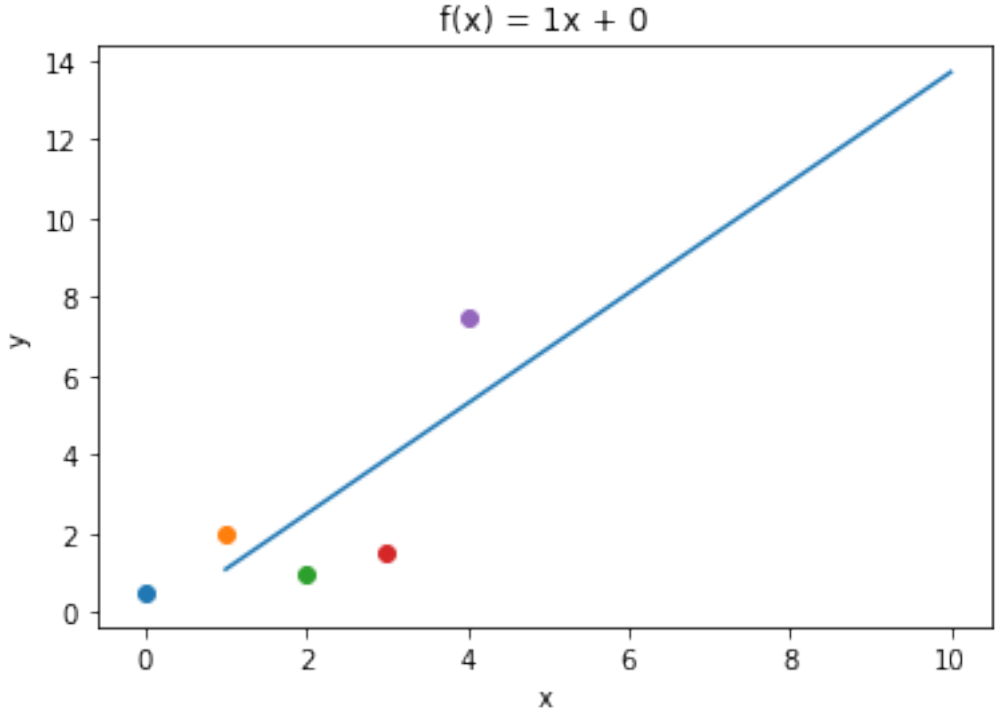
```
        elif(user < com):
            print("user : %d" %user)
            print("com : %d" %com)
            print("You lose...")
        else:
            print("user : %d" %user)
            print("com : %d" %com)
            print("Draw")
        break
    if(ans == 'y'):
        card = random.randint(1,10)
        user = user + card
        if(user > 21):
            print("user : %d" %user)
            print("You lose. Your sum over 21")
            break
        if(user == 21):
            print("user : %d" %user)
            print("BlackJack!! You win")
            break
```

```
user : 10
Do you want more card? (y/n)y
user : 14
Do you want more card? (y/n)y
user : 20
Do you want more card? (y/n)n
user : 20
com : 18
You win!!
```

Problem 3

[5]:
```
import pylab as pl
from math import sqrt
import numpy as np

def compute_error(a,b,D):
    e = 0
    for [i,j] in D:
        e = e + pow(a*i + b - j,2)
    return e

D = [(0,0.5),(1,2.0),(2,1.0),(3,1.5),(4,7.5)]
es = []
a = np.arange(-10,10+0.1,0.1)
b = np.arange(-10,10+0.1,0.1)
min = 100000
```

```
for i in a:
    for j in b:
        temp = compute_error(i,j,D)
        if(abs(temp) < abs(min)):
            min = temp
            ar = i
            br = j

def func(x,a,b):
    y = a*x + b
    return y

t = range(1,10+1)
s = []
for x in t:
    s.append(func(x,ar,br))
pl.plot(t,s)
pl.title("f(x) = %dx + %d"%(ar,br))
pl.xlabel('x')
pl.ylabel('y')
for (i,j) in D:
    pl.scatter(i,j)
pl.show()
```



6. Sets and Dictionaries

Problem 1

[6]:
```
import random

def cntNum():
    nums = dict()
    for i in range(100):
        num = random.randint(1,20)
        if num in nums:
            nums[num] = nums[num] + 1
        else:
            nums[num] = 1

    for i in nums.items():
        print(i)
cntNum()
```

```
(20, 8)
(17, 4)
(14, 10)
(15, 3)
(13, 9)
(10, 8)
(5, 3)
(16, 3)
(6, 4)
(19, 5)
(3, 6)
(2, 5)
(1, 5)
(18, 4)
(11, 2)
(9, 6)
(12, 7)
(4, 3)
(8, 2)
(7, 3)
```

Problem 2

[7]:
```
def count_values1(dic):
    vs = dic.values()
    vs = list(vs)
    count = dict()

    for i in vs:
```

```
        if i in count:
            count[i] = count[i] + 1
        else:
            count[i] = 1

    return len(count)

def count_values2(dic):
    vs = dic.values()
    vs = set(vs)

    return len(vs)

temp = {'red' : 1, 'green' : 1, 'blue' : 2}
print(count_values1(temp))
print(count_values2(temp))
```

```
2
2
```

Problem 3

[8]:
```
def normal_to_sparse(vec):
    sps = dict()

    for i in range(0, len(vec)):
        if vec[i] == 0 : continue
        else:
            sps[i] = vec[i]

    return sps

def change_sign(dic):
    keys = dic.keys()
    for i in keys:
        dic[i] = -dic[i]
    return dic

def add_vector(dic1, dic2):
    rdic = dict()
    for i in dic1.keys():
        for j in dic2.keys():
            if(i == j):
                rdic[i] = dic1[i] + dic2[i]
    for i in dic1.keys():
        if i not in rdic:
            rdic[i] = dic1[i]
    for i in dic2.keys():
```

```
        if i not in rdic:
            rdic[i] = dic2[i]
    return rdic

def minus_vector(dic1, dic2):
    return add_vector(dic1, change_sign(dic2))



vec = [1,0,0,0,0,0,3,0,0,0]
print(normal_to_sparse(vec))
print(change_sign(normal_to_sparse(vec)))
print(add_vector({0:1, 6:3}, {1:2, 6:3}))
print(minus_vector({0:1, 6:3}, {1:2, 6:3}))
```

```
{0: 1, 6: 3}
{0: -1, 6: -3}
{6: 6, 0: 1, 1: 2}
{6: 0, 0: 1, 1: -2}
```

7. Algorithms

Problem 1

[9]:
```
# pseudo code
# def DNA(dna):
#   declare the new object to store new string
#   for i in range(0,length of string):
#       if(dna[i] is 't'): add 'a' to new string and continue to increment
#       if(dna[i] is 'a'): add 't' to new string and continue to increment
#       if(dna[i] is 'c'): add 'g' to new string and continue to increment
#       if(dna[i] is 'g'): add 'c' to new string and continue to increment
#       if(dna[i] is space): add just space to new string
#                            and continue to increment
#       if(dna[i] is not 't','a','c','g' and space): just add dna[i]
#                                                    and continue to increment
#       return the new string


def DNA(dna):
    chgd=""
    for i in range(0,len(dna)):
        if(dna[i]=='t'):
            chgd = chgd+'a'
            continue
        if(dna[i]=='a'):
            chgd = chgd+'t'
            continue
```

```
        if(dna[i]=='c'):
            chgd = chgd+'g'
            continue
        if(dna[i]=='g'):
            chgd = chgd+'c'
            continue
        if(dna[i]==' '):
            chgd = chgd+' '
            continue
        if(dna[i] != 't','a','c','g',' '):
            chgd = chgd+dna[i]
            continue

    return chgd


p53="""1 ttcccatcaa gccctagggc tcctcgtggc tgctgggagt tgtagtctga acgcttctat
61 cttggcgaga agcgcctacg ctccccctac cgagtcccgc ggtaattctt aaagcacctg
121 caccgccccc ccgccgcctg cagagggcgc agcaggtctt gcacctcttc tgcatctcat
181 tctccaggct tcagacctgt ctccctcatt caaaaaatat ttattatcga gctcttactt
241 gctacccagc actgatatag gcactcagga atacaacaat gaataagata gtagaaaaat
301 tctatatcct cataaggctt acgtttccat gtactgaaag caatgaacaa ataaatctta
361 tcagagtgat aagggttgtg aaggagatta aataagatgg tgtgatataa agtatctggg
421 agaaaacgtt agggtgtgat attacggaaa gccttcctaa aaaatgacat tttaactgat
481 gagaagaaag gatccagctg agagcaaacg caaaagcttt cttccttcca cccttcatat
541 ttgacacaat gcaggattcc tccaaaatga tttccaccaa ttctgccctc acagctctgg
601 cttgcagaat tttccacccc aaaatgttag tatctacggc accaggtcgg cgagaatcct
661 gactctgcac cctcctcccc aactccattt cctttgcttc ctccggcagg cggattactt
721 gcccttactt gtcatggcga ctgtccagct ttgtgccagg agcctcgcag gggttgatgg
781 gattggggtt ttcccctccc atgtgctcaa gactggcgct aaaagttttg agcttctcaa
"""
print(DNA(p53))
```

```
1 aagggtagtt cgggatcccg aggagcaccg acgaccctca acatcagact tgcgaagata
61 gaaccgctct tcgcggatgc gaggggggatg gctcagggcg ccattaagaa tttcgtggac
121 gtggcggggg ggcggcggac gtctcccgcg tcgtccagaa cgtggagaag acgtagagta
181 agaggtccga agtctggaca gagggagtaa gtttttttata aataatagct cgagaatgaa
241 cgatgggtcg tgactatatc cgtgagtcct tatgttgtta cttattctat catcttttta
301 agatatagga gtattccgaa tgcaaaggta catgactttc gttacttgtt tatttagaat
361 agtctcacta ttcccaacac ttcctctaat ttattctacc acactatatt tcatagaccc
421 tcttttgcaa tcccacacta taatgccttt cggaaggatt ttttactgta aaattgacta
481 ctcttctttc ctaggtcgac tctcgtttgc gttttcgaaa gaaggaaggt gggaagtata
541 aactgtgtta cgtcctaagg aggttttact aaaggtggtt aagacgggag tgtcgagacc
601 gaacgtctta aaaggtgggg ttttacaatc atagatgccg tggtccagcc gctcttagga
661 ctgagacgtg ggaggagggg ttgaggtaaa ggaaacgaag gaggccgtcc gcctaatgaa
721 cgggaatgaa cagtaccgct gacaggtcga aacacggtcc tcggagcgtc cccaactacc
781 ctaaccccaa aaggggaggg tacacgagtt ctgaccgcga ttttcaaaac tcgaagagtt
```

Problem 2

[10]:
```
def min_index(nums):
    min=nums[0]
    minx=0
    for i in range(0,len(nums)):
        if(nums[i]<min):
            min=nums[i]
            minx=i
    print("min : %d, min_index : %d"%(min,minx))

def max_index(nums):
    max=nums[0]
    maxx=0
    for i in range(0,len(nums)):
        if(nums[i]>max):
            max=nums[i]
            maxx=i
    print("max : %d, max_index : %d"%(max,maxx))


nums=[7,3,4,2,9,8,10,1,6,5]
min_index(nums)
max_index(nums)
```

```
min : 1, min_index : 7
max : 10, max_index : 6
```