In [1]:
```
#Exercise 1.1

def to_celsius(f):
    celcius= (5/9)*(f-32)
    return celcius

to_celsius(78)
```
Out[1]:
```
25.555555555555557
```

In [2]:
```
#Exercise 1.2

def compare(a,b):
    if a==b:
        print('They are equal')
    elif a>b:
        print(a,'is bigger than',b)
    else:
        print(a,'is smaller than',b)

compare(99,98)
```
```
99 is bigger than 98
```

In [3]:
```
#Exercise 1.3

def divisible(a,b):
    if a%b==0:
        return True
    else:
        return False


divisible(5,6)
```
Out[3]:
```
False
```

In [4]:
```
#Exercise 1.4

for i in [-5,5,15,25]:
    if i<=20:
        if i<=10:
            if i<0:
                print('negative')
            else:
                print('small')
        else:
            print('medium')
    else:
        print('big')
```

```
negative
small
medium
big
```

In [5]:
```
#Exercise 1.5

import math
def solve2(a,b,c):
    delta= b**2-4*a*c
    if delta<0:
        return []
    else:
        x_1= int((-b - math.sqrt(delta))/(2*a))
        x_2= int((-b + math.sqrt(delta))/(2*a))
        return x_1,x_2
solve2(1,-2,1)
```

Out[5]:
```
(1, 1)
```

In [6]:
```
#Exercise 2.1 , without list comprehension

def function(ls):
    n=len(ls)
    ans=[0]*n #we need ans to has 'n' columns
    for i in range (n):
        ans[i]=ls[i]**3 -1
    return ans

#with list comprehension

def function1(s):
    m=[]
    for i in range(len(s)):
        m.append(s[i]**3-1)
    return m
function([1,2])

#Explaination on using [0]*n, we need ans to has 'n' columns
#(n=3)
#(ans=[0]*n) the result will be [0, 0, 0]
```

Out[6]:
```
[0, 7]
```

In [97]:
```
#Exercise 2.2, without list comprehension
def square(m,n):
    for i in range(m,n+1):
        for k in range(m,n+1):
            if i==k**2:
                print(k)


#Exercise 2.2, with list comprehension

def square2(x,y):
    ls=[]
    for i in range(x,y+1):
        for j in range(x,y+1):
            if i==j**2:
                ls.append(j)
    return ls

square2(1,9)
```

Out[97]:
```
[1, 2, 3]
```

```
In [8]: #Exercise 2.3, with list comprehension

        def power(m,n,k):
            for i in range(m,n+1):
                for j in range(m,n+1):
                    if i==j**k:
                        print(j)

        #Without list comprehension
        def power1(m,n,k):
            ls=[]
            for i in range(m,n+1):
                for j in range(m,n+1):
                    if i==j**k:
                        ls.append(j)
            return ls

        power1(1,27,3)

Out[8]: [1, 2, 3]
```

```
In [10]: #Exercise 2.4, find the index of number a in a given list b

         def index_of(a,b):
             m=len(b)
             for i in range(m+1):
                 if b[i]==a:
                     return i
             return None

         index_of(3,[1,2,4,3,4,6])

Out[10]: 3
```

```
In [12]: #Exercise 2.5

         def indices_of(a,b):
             indices=[]
             for i in range(len(b)):
                 if b[i]==a:
                     indices.append(i)
             return indices

         indices_of(3,[4,3,1,2,3])

Out[12]: [1, 4]
```

In [14]:
```
#Exercise 2.6

def substitute(a,b,c):
    for i in range(len(a)):
        if a[i]==b:
            a[i]=c
    return a

substitute([1,2,3,4,2],2,'a')
```
Out[14]:
```
[1, 'a', 3, 4, 'a']
```

In [15]:
```
#Exercise 2.7

def substitute1(a,b,c):
    for i in range(len(a)):
        if a[i]==b:
            a[i]=c
mylist = [1,2,3,4,2]
substitute1(mylist,2,'a')
mylist
```
Out[15]:
```
[1, 'a', 3, 4, 'a']
```

In [16]:
```
#Exercise 2.8

def divisibles_by(a,b):
    result=[]
    for i in a:
        if i%b==0:
            result.append(i)
    return result


divisibles_by([6,5,6,8,9],3)
```
Out[16]:
```
[6, 6, 9]
```

In [17]:
```
#Exercise 2.9


def proper_divisor(a):
    prop=[]
    for i in range(1,a):
        if a%i==0:
            prop.append(i)
    return prop

proper_divisor(108)
```
Out[17]:
```
[1, 2, 3, 4, 6, 9, 12, 18, 27, 36, 54]
```

In [18]:
```
#Exercise 2.10


def sigma(n):
    ans = 0
    for i in range(1, n+1):
        if n % i == 0:
            ans += i
    return ans

sigma(78)
```

Out[18]:
```
168
```

In [19]:
```
#Exercise 2.11:
def is_perfect(n): #we use sigma function from the previous problem
    s=sigma(n)-n
    return n==s

is_perfect(6)
```

Out[19]:
```
True
```

In [20]:
```
#Exercise 2.12
#with the helps of proper_divisor from previous problem

def proper_divisor(a):
    prop=[]
    for i in range(1,a):
        if a%i==0:
            prop.append(i)
    return prop
def divisors_ival(m,n):
    for i in range(m, n+1):
        print(i, " -> ", proper_divisor(i))

divisors_ival(30,35)
```

```
30  ->  [1, 2, 3, 5, 6, 10, 15]
31  ->  [1]
32  ->  [1, 2, 4, 8, 16]
33  ->  [1, 3, 11]
34  ->  [1, 2, 17]
35  ->  [1, 5, 7]
```

In [22]:
```
#another way for Exercise 2.12

def divisors_ival1(m,n):
    for i in range(m, n+1):
        prop=[]
        for j in range(1,i):
            if i%j==0:
                prop.append(j)
        print(i, " -> ", prop)

divisors_ival1(30,35)
```

```
30  ->  [1, 2, 3, 5, 6, 10, 15]
31  ->  [1]
32  ->  [1, 2, 4, 8, 16]
33  ->  [1, 3, 11]
34  ->  [1, 2, 17]
35  ->  [1, 5, 7]
```

In [30]:
```
#Exercise 2.13
def is_prime(n):
    for i in range(2,n-1):
        if (n%i)==0:
            return False
        else:
            return True

is_prime(3)
```

In [29]:
```
#Exercise 2.14
def is_prime(n):
    for i in range(3,n-1):
        if n==3:
            return True
        elif (n%i)==0:
            return False
        else:
            return True

def primes_between(m, n):
    ans = []
    for i in range(m, n+1) :
        if is_prime(i):
            ans.append(i)
    return ans

primes_between(1,17)
```

Out[29]:
```
[5, 7, 8, 10, 11, 13, 14, 16, 17]
```

In [32]:
```
#Exercise 2.16


def max_exp(m,n):
    k=0
    while n % pow(m,k)==0:
        k += 1
    return k

max_exp(3,4)
```

Out[32]:
```
1
```

In [34]:
```
#Exercise 2.17:

def isPrime(a):
    return not ( a < 2 or any(a % i == 0 for i in range(2, int(a **

def primes_between(m, n):
    ans = []
    for i in range(m, n+1) :
        if isPrime(i):
            ans.append(i)
    return ans

def prime_decomp(n):
    primes = primes_between(2,n)
    ans=[]
    for prime in primes:
        cnt=0
        while n>0 and n%prime==0:
            n /= prime
            cnt+=1
        if cnt:
            ans.append([prime,cnt])
    return ans

prime_decomp(13)
```

Out[34]:
```
[[13, 1]]
```

In [35]:
```
#Exercise 2.19:

def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b,a%b)


gcd(16,20)
```

Out[35]:
```
4
```

In [36]:
```
#Exercise 2.20:


def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b,a%b)
def phi(n):
    return sum(1 for i in range(1,n) if gcd(n,i)==1)

phi(17)
```

Out[36]:
```
16
```

In [37]:
```
#Exercise 2.21:

def separate(l):
    neg = []
    pos = []
    for num in l:
        if num < 0:
            neg.append(num)
        else:
            pos.append(num)
    return neg, pos


separate([-1,2,3-8,9,-9])
```

Out[37]:
```
([-1, -5, -9], [2, 9])
```

In [38]:
```
#Exercise 2.22:

def is_sorted(l):
    return l==sorted(l)

is_sorted([1,2])
```

Out[38]:
```
True
```

In [39]:
```
#Exercise 2.23:

def my_min(l):
    minimum = l[0]
    for num in l:
        if num < minimum:
            minimum = num
    return minimum

my_min([1,2,3,4,-4])
```

Out[39]:
```
-4
```

In [40]:
```
#Exercise 2.23 another way:

def my_min1(l):
    minimum = min(l)
    return minimum
my_min1([1,2,3,4,-4])
```

Out[40]:
```
-4
```

In [41]:
```
#Exercise 2.24:

def min_index(l):
    minimum= min(l)
    for i in range(len(l)):
        if l[i]==minimum:
            return i

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

Out[41]:
```
4
```

In [43]:
```
#Exercise 2.25:
#Write a function min_indices() which returns the list of the indic
#the occurrences of the minimal member of the nonempty list of numb
#is given as its only argument.

def min_indices(l):
    indices=[]
    minimum= min(l)
    for i in range(len(l)):
        if l[i]==minimum:
            indices.append(i)
    return indices

min_indices([1,3,4,2,1,3,1,2])
```

Out[43]:
```
[0, 4, 6]
```

In [45]:
```
#Exercise 2.26:

def nearest_to_average(lst):
    avg=sum(lst)/len(lst)
    return min(range(len(lst)), key=lambda i: abs(lst[i]-avg))

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

Out[45]:
```
2
```

In [48]:
```
#Exercise 2.26

def nearest_to_avg(ls):
    if len(ls)==0:
        return None
    avg = sum(ls)/len(ls)
    dif = abs(ls[0]-avg)
    ans = ls[0]
    for num in ls:
        d = abs (num - avg)
        if d<dif:
            ans = num
            d = dif
    return ans

nearest_to_avg([1,2,8,4,5])
```

Out[48]:
```
5
```

In [49]:
```
#Exercise 2.27:

def has_duplicates(l):
    count=0
    for i in range(len(l)):
        for j in range(i+1, len(l)):
            if l[i]==l[j]:
                return True
    return False

has_duplicates([1,2,8,4,5])
```

Out[49]:
```
False
```

In [51]:
```
#Exercise 2.28

def longest_run(ls):
    ans = min(0,len(ls))
    cnt = 1
    for i in range(1,len(ls)):
        if ls[i]==ls[i-1]:
            cnt +=1
        else:
            cnt=1
        ans=max(ans,cnt)
    return ans

longest_run([1,2,3,3,4])
```

Out[51]:
```
2
```

In [52]:
```
longest_run([1,2,2,2,1,2,3,3,4])
```

Out[52]:
```
3
```

In [55]:
```
#Exercise 2.29

def multiplication_table(n):
    for i in range(1,n+1):
        print(i,end =': ')
        for j in range(1,10):
            print(i*j , end =' ')
        print()

multiplication_table(9)
```

```
1: 1 2 3 4 5 6 7 8 9
2: 2 4 6 8 10 12 14 16 18
3: 3 6 9 12 15 18 21 24 27
4: 4 8 12 16 20 24 28 32 36
5: 5 10 15 20 25 30 35 40 45
6: 6 12 18 24 30 36 42 48 54
7: 7 14 21 28 35 42 49 56 63
8: 8 16 24 32 40 48 56 64 72
9: 9 18 27 36 45 54 63 72 81
```

In [57]:
```
#Exercise 2.30:

def permutation(n):
    for i in range(n):
        for j in range (n):
            if i!=j:
                for k in range(n):
                    if i!=k and k!=j:
                        print(i,j,k)

permutation(3)
```

```
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
2 1 0
```

In [58]:
```
#Exercise 2.31

#version1 with recursion

def factorial(n):
    if n<=1:
        return 1
    else:
        return n*factorial(n-1)



#Version2 without recursion

def factorial1(m):
    result=1
    if m<=1:
        return 1
    for i in range(1,m+1):
        result=result*i
    return result



factorial(4)
```

Out[58]:
```
24
```

In [85]:
```
#Exercise 2.32

def lucas(n):
    if n==0:
        return 2
    if n==1:
        return 1

    return lucas(n-2)+lucas(n-1)

lucas(4)
```

Out[85]:
```
7
```

In [72]:
```
#Exercise 2.33
```

In [73]:
```
#Exercise 2.34
```

In [67]:
```
#Exercise 2.35
#Pascal Triangle

def pascal(n):
    ans=[[1]]
    for i in range (1,n):
        ls = [1]*(i+1)
        for j in range(1,i):
            ls[j]=ans[i-1][j-1]+ans[i-1][j]   # wtf is this
        ans.append(ls)
    return ans

pascal(5)
```

Out[67]:
```
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]
```

In [68]:
```
#Exercise 2.36
```

In [91]:
```
#Exercise 2.37

def insert_sort(ls):
    n=len(ls)
    for i in range(n-1):
        j=i+1
        while j>0 and ls[j]<ls[j-1]:
            ls[j],ls[j-1]=ls[j-1],ls[j]  # what does it mean when w
            j-=1
    return ls

insert_sort([1,4,3,2,6,8])
```

Out[91]:
```
[1, 2, 3, 4, 6, 8]
```

In [92]:

```
#Exercise 3.1

count=0
sum=0

while True:
    number = input("Enter a number: ")
    if number != "":
        sum = sum + int(number)
        count +=1
    else:
        break

print("The average is",sum/count)
```

```
Enter a number: 3
Enter a number: 3
Enter a number: 4
Enter a number: 5
Enter a number: 
The average is 3.75
```

In [109]:

```
# Exercise 3.1 another way

def fl():
    ls=[]
    while True:
        try:
            num=((input("Enter a number: ")))
            ls.append(int(num))
        except:
            break
    ans=sum(ls)/float(len(ls))
    return "The average is "+ str(ans)
```

In [102]:
```
#Exercise 3.2
numbers= input("Enter numbers seperated by space: ").split(" ")

numbers_list=[]
for number in numbers:
    try:
        numbers_list.append(int(number))
    except:
        pass

sum=0
for number in numbers_list:
    sum += number

print("The average is: {}".format(sum/len(numbers_list)))
```

```
Enter numbers seperated by space: 3
The average is: 3.0
```

In [ ]:
```
# Exercise 3.2 another way

def fl():
    ls=[]
    while True:
        try:
            num=int((intput("Enter a numb")))
```

In [90]:
```
#Exercise 3.3
def read_first_lines(filename,n):
    with open(filename,'r') as input_file:
        for line_number, line in enumerate(input_file):
            if line_number >= n:
                break
            print(line.rstrip())

read_first_lines('blah blah.txt',1)
```

```
Sadfkjsal;dgjals;dgjaslkdgjalksdgjkasldjgjasdlfljasdsjkl
```

In [98]:
```
#Exercise 3.4

def copy_first_lines(in_file,out_file,n):
    in_f=open(in_file,'r')
    out_f=open(out_file,'w')
    cnt=0
    for line in in_f:
        if cnt==n:
            break
        cnt+=1
        out_f.write(line)
    in_f.close()
    out_f.close()
```

In [99]:
```
#Exercise 3.5

def count_lines(name):
    f=open(name,'r')
    n=sum(line for line in f)
    f.close()
    return n
```

In [100]:
```
#Exercise 3.6

def read_to_string(name):
    f=open(name,'r')
    ls=[line.rstrip() for line in f]
    f.close()
    a=" ".join(ls)
    return a
```

In [101]:
```
#Exercise 4.1

def list_diff(a,b):
    for i in range(len(b)):
        for j in range(len(b)):
            if a[i]==b[j]:
                a.remove(a[i])
    return a

#Exercise 4.1 the right answer

def list_diff1(lis1,lis2):
    newlist=[]
    for x in lis1:
        if x not in lis2:
            newlist.append(x)
    return newlist


list_diff1(list(range(10)),list(range(0,15,3)))
```

Out[101]:
```
[1, 2, 4, 5, 7, 8]
```

In [102]:
```
#Exercise 4.2    Print the following pattern:


for i in range(10):
    for j in range(i):
        print(i,end='')

    print("")
```

```

1
22
333
4444
55555
666666
7777777
88888888
999999999
```

In [103]:
```
# Or another way

def num_pattern():
    for j in range(1,10):
        print(str(j)*j)

num_pattern()
```

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

In [106]:
```
#Exercise 4.3 wave(a,b)

def wave(a,b):

    for m in range(1,b+1):
        for i in range(1,a+1):
            print(str('o')*i)
        for i in range(0,a):
            print(str('o')*(a-i))

wave(5,3)
```

```
o
oo
ooo
oooo
ooooo
ooooo
oooo
ooo
oo
o
o
oo
ooo
oooo
ooooo
ooooo
oooo
ooo
oo
o
o
oo
ooo
oooo
ooooo
ooooo
oooo
ooo
oo
o
```

In [107]:
```
#Exercise 4.3 new_wave(a,b)

def new_wave(a,b):

    for m in range(1,b+1):
        for i in range(1,a+1):
            print(str('o')*i)
        for i in range(1,a+1):
            print(str('o')*(a-i))

new_wave(5,1)
```

```
o
oo
ooo
oooo
ooooo
oooo
ooo
oo
o
```

In [109]:
```
#Exercise 4.4

def merge(l1,l2):
    new=[]
    if len(l1)==len(l2):
        for i,j in zip(l1,l2):
            new+= [i,j]
    else:
        new = [item for item in zip(l1[:len(l2)], l2)]
        new.extend(l1[len(l2):])
    return new

def merge2(ls1, ls2):
    a=len(ls1)
    b=len(ls2)
    d=min(a,b)
    ans=[]
    for i in range(d):
        ans.append(ls1[i])
        ans.append(ls2[i])
    if a>=b:
        ans=ans+ls1[d:]
    else:
        ans=ans+ls2[d:]
    return ans

merge2([1,2,3,4],[5,6,7,8,9,10,11])
```

Out[109]:
```
[1, 5, 2, 6, 3, 7, 4, 8, 9, 10, 11]
```

In [2]:
```
#Exercise 4.5

def cat(some_file):
    with open(some_file) as file:
        for line in file:
            print(line.upper())

cat('some_file.txt')   #YEAHHHHH I FINALLY DID IT MYSELFFFFFF LOLLL
```

```
WRITE SOME FILE LOL
```

In [111]:
```
#Exercise 4.6

def cat2(x):
    count=0
    sum=0
    with open(x) as file:
        for line in file:

            line1=float(line)
            sum+=line1
            count +=1
        print(sum/count)

cat2('some_file2.txt')
```

```
4.0
```

In [112]:
```
#Exercise 4.7

def transpose(m):
    for i in range(len(m)):
        for j in range(i+1,len(m)):
            m[i][j],m[j][i]=m[j][i],m[i][j]
    return m

transpose([[1,2,3],[4,5,6],[7,8,9]])
```

Out[112]:
```
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

In [113]:
```
#Exercise 4.8

def add_matrices(m1,m2):
    ans=[]
    for i in range(len(m1)):
        new=[]
        for j in range(len(m1[0])):
            new.append(m1[i][j]+m2[i][j])
        ans.append(new)
    return ans


add_matrices([[1,0,0],[0,1,0],[0,0,1]],[[1,2,3],[4,5,6],[7,8,9]])
```

Out[113]:
```
[[2, 2, 3], [4, 6, 6], [7, 8, 10]]
```

In [114]:
```
#Exercise 4.9

def matrix_mult (m1, m2):
    C = [[0 for row in range(len(m1))] for col in range(len(m2[0]))]
    for i in range(len(m1)):
        for j in range(len(m2[0])):
            for k in range(len(m2)):
                C[i][j] += m1[i][k]*m2[k][j]
    return C

matrix_mult([[1,2],[3,4],[5,6]],[[1,0,0],[0,1,0]])
```

Out[114]:
```
[[1, 2, 0], [3, 4, 0], [5, 6, 0]]
```

In [115]:
```
#Exercise 4.9 again

def matrixmult(m1,m2):
    ans=[]
    for x in m1:
        ans.append([0]*len(m2[0]))
    for k in range(len(m1)):
        for i in range(len(m2[0])):
            for j in range(len(m2)):
                ans[k][j]+=m1[k][j]*m2[j][i]
    return ans

matrixmult([[1,2],[3,4],[5,6]],[[1,0,0],[0,1,0]])
```

Out[115]:
```
[[1, 2, 0], [3, 4, 0], [5, 6, 0]]
```

In [116]:
```
#Exercise 4.10

def myzip(ls,st):
    ans=[(i,j) for i, j in zip(ls,st)]
    return ans

myzip('abcdefg', list(range(4)))
```
Out[116]:
```
[('a', 0), ('b', 1), ('c', 2), ('d', 3)]
```

In [124]:
```
#Exercise 4.10 again

def myzip2(ls1,ls2):
    ans1=[]
    for ans in ((ls1[i], ls2[i]) for i in range(min(len(ls1), len(l
        ans1.append(ans)
    return ans1
myzip2('abcdefg', list(range(4)))
```
Out[124]:
```
[('a', 0), ('b', 1), ('c', 2), ('d', 3)]
```

In [125]:
```
# Exercise 4.11
#Write a function lindex(string, substring) that returns the smalle
#where substring is found, or -1 if substring is not found.
#This is almost what the .index() method does, so don’t use that!

def lindex(string, substring):
    if substring not in string:
        return -1
    else:
        result = string.index(substring)
    return result

lindex("At the turn of the century", "them")
```
Out[125]:
```
-1
```

In [126]:
```
#Exercise 4.11 agian

def lindex1(string,sub):
    if sub not in string:
        return -1
    else:
        return string.find(sub)

lindex1("At the turn of the century", "the")
```
Out[126]:
```
3
```

In [127]:
```
# Exercise 4.11 again

def lindex2(s, sub):
    index = 0
    if sub in s:
        for ch in s:
            if ch == sub[0]:
                if s[index:index+len(sub)] == sub:
                    return index
            index += 1
    return -1

lindex2("At the turn of the century", "the")
```

Out[127]:
```
3
```

In [128]:
```
# Exercise 4.11 again

def lindex3(s,char):

    for index in range(len(s)-len(char)-1):
        if s[index:index+len(char)] == char:
            return index
    return -1

lindex3("At the turn of the century", "the")
```

Out[128]:
```
3
```

In [129]:
```
# Exercise 4.12

def count_occurrences(s,ss):
    if ss not in s:
        return 0
    else:
        print(s.count(ss))

count_occurrences("At the turn of the century", "the")
```

```
2
```

In [130]:
```
# Exercise 4.12 again

def count(string, sub_string):

    ans = 0

    for i in range(len(string)-(len(sub_string)-1)):
        if sub_string == string[i:len(sub_string)+i]:
            ans += 1
    return ans

    if sub_string!= string[i:len(sub_string)+i]:
        return 0

count("At the turn of the century the", "them")
```

Out[130]:
```
0
```

In [131]:
```
# Exercise 4.12
def count(string, sub_string):
    ans = 0
    for i in range(len(string)-(len(sub_string)-1)):
        if sub_string == string[i:len(sub_string)+i]:
            ans += 1
    return ans

    if sub_string!= string[i:len(sub_string)+i]:
        return 0

count("At the turn of the century the", "them")
```

Out[131]:
```
0
```

In [159]:
```
s=[1,2,3,4,5,7,9,4]
s[2:4]  #this is what slicing does s[i:j]
```

Out[159]:
```
[3, 4]
```

In [3]:
```
# Exercise 4.13

import csv
with open('ball.csv','r') as file:  # use f-string and 'r' stands f
    reader = csv.reader(file)
    for line in reader:
        print(f'Good:{line[0]}\nAmount:{line[1]}\nUnit price:{line[
```

```
Good:ball
Amount:570
Unit price:0.13
Good:table
Amount:3
Unit price:2000
Good:racket
Amount:12
Unit price:185
Good:net
Amount:17
Unit price:23
```

In [7]:
```
# Exercise 4.14

import csv
with open('ball.csv','r') as file:  # use f-string
    reader = csv.reader(file)
    for line in reader:
        print(f'Good:{line[0]}\nTotal price:{float(line[1])*float(l
```

```
Good:ball
Total price:74.10000000000001
Good:table
Total price:6000.0
Good:racket
Total price:2220.0
Good:net
Total price:391.0
```

In [166]:
```
# Exercise 4.15
print(f'{" ":4}{"1":4}{"2":4}{"3":4}{"4":4}{"5":4}{"6":4}{"7":4}{"8

#{"8":4}{"9":4}\n{" ":4}{"+"*33}') continue from the above row

for row in range(1, 10):
    print(row,end=":")
    print(*(f'{row*col:3}' for col in range(1, 10)))  #  this {a:b}
```

```
      1   2   3   4   5   6   7   8   9
      +++++++++++++++++++++++++++++++++
1:    1   2   3   4   5   6   7   8   9
2:    2   4   6   8  10  12  14  16  18
3:    3   6   9  12  15  18  21  24  27
4:    4   8  12  16  20  24  28  32  36
5:    5  10  15  20  25  30  35  40  45
6:    6  12  18  24  30  36  42  48  54
7:    7  14  21  28  35  42  49  56  63
8:    8  16  24  32  40  48  56  64  72
9:    9  18  27  36  45  54  63  72  81
```

In [187]:
```python
# Exercise 4.16

def lost_by_point(x):

    print(f'{" ":12}{"Club":20}{"GP":10}{"Pts":10}{"-Pts":10}')
    with open('pl.csv') as file:
        reader = csv.reader(file)
        for i,line in enumerate(reader,1):
            print(f'{i:11} {line[0]:20}{line[1]:10} {line[2]:10} {i

lost_by_point('pl.csv')
```

```
            Club                GP        Pts       -Pts
          1 Man City            23         57         12
          2 Liverpool           22         48         18
          3 Chelsea             24         47         25
          4 Man Utd             22         38         28
          5 West Ham            23         37         32
          6 Arsenal             21         36         27
          7 Tottenham           20         36         24
          8 Wolverhampton       21         34         29
          9 Brighton            22         30         36
         10 Leicester           20         26         34
         11 Aston Villa         21         26         37
         12 Southampton         22         25         41
         13 Crystal Palace      22         24         42
         14 Brentford           23         23         46
         15 Leeds               21         22         41
         16 Everton             20         19         41
         17 Norwich             22         16         50
         18 Newcastle           21         15         48
         19 Watford             20         14         46
         20 Burnley             18         12         42
```

In [199]:
```python
# Exercise 4.28

def even_odd(ls1, ls2):
    ans1=[x for x in ls1 if x%2==0]
    ans2=[y for y in ls2 if y%2==1]
    return ans1+ans2

even_odd([1,2,3,4,5],[8,6,4,5,2])
```

Out[199]:
```
[2, 4, 5]
```

In [201]:
```
# Exercise 4.29

def same_mod(ls1, ls2, m):
    ans = [(i,j) for i in ls1 for j in ls2 if (i-j)%m==0]
    return ans
same_mod(list(range(4)), list(range(2,8)),3)
```

Out[201]:
```
[(0, 3), (0, 6), (1, 4), (1, 7), (2, 2), (2, 5), (3, 3), (3, 6)]
```

In [205]:
```
# Exercise 4.30

def same_mod1(ls1,ls2,m):
    ans = [(i,j) for i in ls1 for j in ls2 if (i-j)%m==0 and i%m!=0
    return ans
same_mod1(list(range(4)), list(range(2,8)),3)
```

Out[205]:
```
[(1, 4), (1, 7), (2, 2), (2, 5)]
```

In [208]:
```
# Exercise 4.31

def squares(lst, n):
    ans = [[j*j for j in list(range(i,i+n))] for i in lst]
    return ans

squares(list(range(1,8,3)),3)
```

Out[208]:
```
[[1, 4, 9], [16, 25, 36], [49, 64, 81]]
```

In [190]:
```
# Exercise 4.32

def concatenate(L):
    new_list=[]
    for item in L:
        if isinstance(item,list):
            for num in item:
                new_list.append(num)
    return new_list

concatenate([list(range(3)),list(range(3,6)),list(range(6,8))])
```

Out[190]:
```
[0, 1, 2, 3, 4, 5, 6, 7]
```

In [241]:
```
# Exercise another 4.32

def concatenate1(ls):
    return [s for x in ls for s in x]


concatenate1([list(range(3)),list(range(3,6)),list(range(6,8))])
```

Out[241]:
```
[0, 1, 2, 3, 4, 5, 6, 7]
```

In [233]:
```
# Exercise 4.33

def matrix_add(m1,m2):

    result = [[x + y for x, y in zip(a, b)] for a, b in zip(m1,m2)]

    return result

matrix_add([[1,2],[3,4]],[[1,2],[3,4]])
```

Out[233]:
```
[[2, 4], [6, 8]]
```

In [236]:
```
# Exercise 4.33 another way

def matrix_Add(m1,m2):
    result = []
    for a, b in zip(m1, m2):
        current_list = []
        for x, y in zip(a, b):
            current_list.append(x + y)
        result.append(current_list)
    return result

matrix_Add([[1,2],[3,4]],[[1,2],[3,4]])
```

Out[236]:
```
[[2, 4], [6, 8]]
```

In [227]:
```
# Exercise 5.1

def sumtree(l):
    s = 0
    for i in l:
        if isinstance(i, list):
            s += sumtree(i)
        else:
            s += i
    return s

sumtree([0, [0, 1, 2], [0, [0, 1, 2], 1, [0, 1, 2], 2], 2])
```

Out[227]:
```
14
```

In [192]:
```
# Exercise 5.1 another way

def sumtree1(tree):
    ans=0
    for i in tree:
        if type(i) is int:
            ans+= i
        else:
            ans += sumtree(i)
    return ans

sumtree1([0, [0, 1, 2], [0, [0, 1, 2], 1, [0, 1, 2], 2], 2])
```

Out[192]:
```
14
```

In [193]:
```
# Exercise 5.2

def flatten(L):
    if len(L)==1:
        if type(L[0])==list:
            result=flatten(L[0])
        else:
            result=L
    elif type(L[0])==list:
            result=flatten(L[0])+flatten(L[1:])
    else:
        result= [L[0]]+flatten(L[1:])
    return result


flatten([1, 2, [3, 4]])
```

Out[193]:
```
[1, 2, 3, 4]
```

In [194]:
```
M=[1,2,3,4,5,6,7,8,8,1,2,3]  # example of slicing list
M[1:]
```

Out[194]:
```
[2, 3, 4, 5, 6, 7, 8, 8, 1, 2, 3]
```

In [195]:
```
# 5.2 another method

def flatten1(tree,ans=[]):
    for item in tree:
        if type(item) is int:
            ans.append(item)
        else:
            flatten1(item,ans)
    return ans

flatten1([1, 2, [3, 4]])
```

Out[195]:
```
[1, 2, 3, 4]
```

In [196]:
```
# Exercise 5.3

def sublists(ls):
    ans=[[]]
    for num in ls:
        n=int(len(ans))
        for i in range(n):
            t=list(ans[i])
            t.append(num)
            ans.append(t)
    return ans

sublists([1,2])
```

Out[196]:
```
[[], [1], [2], [1, 2]]
```

In [210]:
```
list(range(1,8,3))  #some example on list
```

Out[210]:
```
[1, 4, 7]
```

In [216]:
```
# Exercise 5.4


def lookup(key,ls):
    for a,b in ls:
        if a==key:
            return b
    return None
```

In [245]:
```
# Exercise 5.5 Concatenates from problem 4.32

def new_concatenate(*lists):

    return [num for ls in lists for num in ls]

new_concatenate(list(range(3)),list(range(3,6)),list(range(6,8)))
```

Out[245]:
```
[0, 1, 2, 3, 4, 5, 6, 7]
```

In [255]:
```
# Exercise 5.6

def myzip(*l):
    ans=[]
    for i in l:
        if type(i)!=list:
            i=list(i)
    n = min (len(i) for i in l)
    for j in range(n):
        ls=[]
        for s in l:
            ls.append(s[j])
        ans.append(tuple(ls))
    return ans

myzip('abdsdf',range(1,6),range(8,12))
```

Out[255]:
```
[('a', 1, 8), ('b', 2, 9), ('d', 3, 10), ('s', 4, 11)]
```

In [256]:
```
# Exercise 5.7

def transpose(mat):
    n=len(mat)
    m=len(mat[0])
    ans=[]
    for j in range(m):
        row=[mat[i][j] for i in range(n)]
        ans.append(row)
    return ans

transpose([[1,2],[3,4],[5,6]])
```

Out[256]:
```
[[1, 3, 5], [2, 4, 6]]
```

In [ ]: