```
# Some functions are already available in Python,
# such as log or sqrt.
# but we can also defined our own functions.
# This is useful when we need to perform a particular
# calculation many times.
# For example: factorial, binomial, etc


########################################
# FACTORIAL
# With the FOR-loop we computed the factorial of a number
# in one of the exercises of Lec06.
# If we had to do this for many different numbers at different
# parts of our code, it might be handy to have a
# "factorial function defined"
print()
print("-----------")
print(" FACTORIAL ")
print("-----------")

print()
print("Factorial of 8:")
fac=1.;
for n in range(8):
    fac=fac*(n+1)
print(fac)


print()
print("This also works")
fac=1.;
for n in range(8):
    fac*=(n+1)
print(fac)

print()
print("Or even better, by defining a function")
def factorial(x):
    fac = 1.0
    for n in range(x):
        fac = fac*(n+1)
    return fac

a = factorial(8)
print(a)
# CAREFUL!! Do NOT use a floating number x in factorial(x),
# because x is inside range(x), so it needs to be an integer!!
# NOTE also that "fac" and "n" are LOCAL variables. Their values
# are NOT kept in the computer's memory.

print()
```

```
print("With the function, we can get factorials whenever needed")
print("4! = ",factorial(4), ", 7! = ", factorial(7))



########################################
# CYLINDRICAL COORDINATES
# We are given the cylindrical coordinates
# r, theta, z of a point and we want to
# compute the distance "d" between this point
# and the origin. For that we need to use the following equations
# x = r cos(theta)
# y = r sin(theta)
# d = sqrt(x^2 + y^2 + z^2)
print()
print()
print("-------------------------")
print(" CYLINDRICAL COORDINATES ")
print("-------------------------")

from math import *
def distance(r,theta,z):
    x = r*cos(theta)
    y = r*sin(theta)
    dist = sqrt(x**2 + y**2 + z**2)
    return dist
print("For r=2, theta=0.1, z=-1.5, the distance is ",distance(2.,0.1,-1.5))



########################################
# POLAR to CARTESIAN
# Given the values of r and theta, create
# an array with the values of x and y
print()
print()
print("-------------------------------")
print(" POLAR to CARTESIAN: using array ")
print("-------------------------------")

from numpy import array
def PolarToCartesian(r,theta):
    x = r*cos(theta)
    y = r*sin(theta)
    newCart = array([x,y], float)
    return newCart
myArray = PolarToCartesian(1.,pi)
print("[x,y] = ", myArray)
```

```
########################################
# POLAR to CARTESIAN
# Given the values of r and theta, create
# an array with the values of x and y
print()
print()
print("-------------------------------------")
print(" POLAR to CARTESIAN: multiple values ")
print("-------------------------------------")

from numpy import array
def PolarToCartesian(r,theta):
    x = r*cos(theta)
    y = r*sin(theta)
    return x,y
myX,myY = PolarToCartesian(1.,pi)
print("x = ",myX)
print("y = ",myY)




########################################
# POLAR to CARTESIAN
# Given the values of r and theta, create
# an array with the values of x and y
print()
print()
print("--------------------------------")
print(" POLAR to CARTESIAN: using *np* for numpy ")
print("--------------------------------")

import math
import numpy as np

def PolarToCartesian(r,theta):
    x = r*math.cos(theta)
    y = r*math.sin(theta)
    newCart = np.array([x,y], float)
    return newCart

myArray = PolarToCartesian(1.,math.pi)
print("[x,y] = ", myArray)




########################################
```

```
# EXAMPLE 2.8
# Given an integer "n"
#
# (i) Define a function that finds the prime factors
# of a number n, that is, the prime numbers that can be multiplied
# to give the original number "n".
# This can be obtained by dividing "n" repeatedly by
# all integers from 2 up to "n" and checking to see if
# the remainder after the division is zero.
#
# (ii) Use (i) to find out if "n" is prime.
#
#
print()
print()
print("--------------------------------")
print(" PRIME FACTORS and PRIME NUMBER ")
print("--------------------------------")

def factors(x):
# let us start with an empty LIST, where we will add the
# prime factors
    primeFactors=[]
# k is the number for the operation x/k, which starts at 2
    k = 2
# we will keep dividing x/k up to when k=x ("while k<=x)
    while k<= x:
# if the division gives remainder=0, then k is a prime factor.
# Since this prime factor can appear more than once,
# such as in 12 -> [2, 2, 3],
# we keep appending "while" it works (i.e. "while" the remainder=0)
        while x%k == 0:
# if satisfied, it gets appended to the LIST
            primeFactors.append(k)
# and we now repeat the process to a new INTEGER x
            # note // instead of /
            x = x//k
# Once, we are done with one k, we go to the next k:
        k = k+1
# Once all k's up to k=x have been tested, we get the output
    return primeFactors

listFactors = factors(17556)
print("Factors of 17556 are ", listFactors)


print()
print()
print("--------------")
print(" PRIME NUMBER ")
print("--------------")
```

```
thisNumb=37
isitP = factors(thisNumb)
if len(isitP) == 1:
    print(thisNumb," is prime")

print()
print("Let us print the prime numbers between 1 and 49")
for n in range(50):
    thisNumb = n+1
    isitP = factors(thisNumb)
    if len(isitP) == 1:
        print(thisNumb," is prime")
```