```
# "FOR" LOOPS
# This is one of the most useful tools in programming
# In most languages, it is known as the DO-LOOP

print("--------")
print("FOR LOOP")
print("--------")

# ----------------------------------------
# Print a sentence many times
for n in range(5):
    print("I can be repetitive sometimes.")
print("Done!")



# ----------------------------------------
# Print the first 5 integers, including zero.
# NOTE: Python starts from ZERO
# and stops one number BEFORE the one in RANGE
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
print()
print("Remember that Python starts counting from ZERO")
for n in range(5):
    print(n)
print("Done!")

print()
print("Remember that Python starts counting from ZERO")
for n in range(0,5):
    print(n)
print("Done!")



# ----------------------------------------
# Print the integers from 1 to 5
print()
print("Remember that Python starts counting from ZERO")
for n in range(1,6):
    print(n)
print("Done!")



# ----------------------------------------
# Print the integers from 2 to 17 in steps of 3
print()
print("Steps of 3")
for n in range(2,20,3):
    print(n)
```

```
print("Done!")



# ------------------------------------------
# Print the integers from 20 to 5 in steps -3
print()
print("Decreasing in steps of -3")
for n in range(20,2,-3):
    print(n)
print("Done!")



# ------------------------------------------------
# Sum the odd numbers from 1 to 11 INCLUDING 1 and 11
print()
print("Sum of odd numbers in (1,11)")
SumOdd=0
for n in range(1,12,2):
    print(n, sumOdd) # print here just to check
    SumOdd = SumOdd + n
print("The sum is", SumOdd)



# ------------------------------------------
# Create an ARRAY with the 5 lowest even numbers including zero
# Use floating numbers
print()
print("First I create an array and then correct its elements")
from numpy import zeros
vec = zeros(5,float)
for n in range(0,5):
#    print(n) # use print for checks when writing the code
    vec[n]=2.*float(n)
print(vec)



# ------------------------------------------
# Create an array with the 5 lowest even numbers NOT including zero
print()
print("Starting from n+1, since the first is n=0")
vec = zeros(5, float)
for n in range(5):
    vec[n]=2.*float(n+1)
print(vec)


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

```
### STUDENTS TRY IN CLASS
################################################


# ----------------------------------------
# Create the vector {2., 5.,  8., 11., 14., 17.}
# The elements go from 2 to 17 in steps of 3
# Use floating numbers
print()
print("Vector: {2, 5,  8, 11, 14, 17}")
vec = zeros(6, float)
for n in range(0,6):
    vec[n] = 2. + 3*n
print(vec)

# CAREFUL!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# What is WRONG WRONG WRONG below?????!!!!
vec = zeros(6, float)
for n in range(2,18,3):
    print(n)
#    vec[n] = n
print(vec)

# Instead, you could
# have another variable for the index
print("Vector: {2, 5,  8, 11, 14, 17}")
vec = zeros(6, float)
index = 0
for n in range(2,18,3):
    print(n)
    vec[index] = n
    index = index + 1
print(vec)



# ----------------------------------------
# Create a 7x7 diagonal matrix where
# all elements are zero, EXCEPT for the
# diagonal elements,
# which are equal to the sum of the PYTHON indices.
# Remember that the first element in Python is (0,0)
# so the first element will be 0+0=0;
# the 2nd element of the diagonal 1+1=2, etc
print()
print("Diagonal matrix with elements i+i")
mat = zeros([7,7],float)
for n in range(7):
    mat[n,n] = n + n
print(mat)
```

```
# ----------------------------------------
# Construct a 5 X 5 upper triangular matrix of 1s
# with 0s below the main diagonal.
print()
print("Upper triangular of 1's")
mat = zeros([5,5],float)
for n in range(5):
    for m in range(n,5):
        mat[n,m]=1
#        print(n,m,mat[n,m])
print(mat)


# ----------------------------------------
# Suppose that the dot product from numpy did not exist
# Compute the dot product between
# the vectors {6., 5., 4.} and {1., 2., 3.}
# manually using FOR-LOOP
# Compare it with the result from the numpy dot product
print()
print("Manual dot product")

from numpy import array,dot
vecA = array([6, 5, 4], float)
vecB = array([1, 2, 3],float)

vecDP=dot(vecA,vecB)
print("From the dot product:",vecDP)

dpVV=0.
for n in range(3):
    dpVV = dpVV + vecA[n]*vecB[n]
print("From manual dot product:", dpVV)


# ----------------------------------------
# Example 2.7 from the book
# Emission lines of the Hydrogen atom are given by the
# Rydberg formula
#      1/lambda = R (1/m^2 - 1/n^2)
# where
# R is a constant equal to 1.097x10^(-2) nm^(-1)
# lambda is the wavelength and
# 1/lambda is the wavenumber
# m and n are integers with n>m.
# Each m determines a series, and
# within each series, each n determines a line.
# Print the first 5 lines of the first 3 series.
```

```
# CAREFUL! n and m CANNOT be ZERO!!!!!!!!!!!!

# CAREFUL! We want ALL 5 lines where n>m
# For m=1, this means n=2,3,4,5,6
# For m=2, this means n=3,4,5,6,7
print()
print("-----------")
print("Example 2.7")
print("-----------")

Ryd = 1.097e-2

for m in range(1,4):
    print()
    print("Serie:", m)
    for n in range(m+1,m+6):
        invlam = Ryd*(1/m**2 - 1/n**2)
        print("n=", n, "lambda=", 1./invlam, "nm")


# ----------------------------------------
# Make a 3x5 matrix with the values from the problem above
print()
print("------------------------------------------------------------")
print("Make a 3x5 matrix with the values from the problem above")

Ryd = 1.097e-2

# lam is the matrix we want to create
lam = zeros([3,5],float)

# ind1 = index for the rows of my matrix
# ind2 = index for the columns of my matrix
ind1=-1

for m in range(1,4):
    ind1 = ind1 + 1
    ind2=-1
    for n in range(m+1,m+6):
        ind2 = ind2 + 1
        invlam = Ryd*(1/m**2 - 1/n**2)
# The print line below is used for testing, while writing the code.
#        print(ind1,ind2,1./invlam)
        lam[ind1,ind2]=1./invlam
print(lam)


# ----------------------------------------
# In problem 2.7, within each serie,
# which line is closest to an even number?
# (You need to find out which line modulus 2 gives the
```

```
# smallest remainder)
# After you find the line, print its
# m, n, and the remainder

# Do NOT used the stored values of "lam" above.
# Start the problem from scratch, by finding lambda
# for each serie and each line
print()
print("-----------------------------------")
print("Find a specific line in each serie")
print("-----------------------------------")

Ryd = 1.097e-2

for m in range(1,4):
    remainSmall = 10000000.0
    for n in range(m+1,m+6):
        invlam = Ryd*(1/m**2 - 1/n**2)
        lamb=1./invlam
        remains = lamb%2
# The print line below is for testing
#        print("Serie:", m, "n=", n, "remain=", remains)
        if remains<remainSmall:
            remainSmall = remains
            thisline = n
# Now I print which line that is
    print("m =",m,"n =",thisline,"remainder =",remainSmall)
```