#1.py

Let's just use `randrange` and `random`

```python
from random import *

def main():
    # generate 10 random numbers between 1 and 100
    # these numbers will be "uniformly distributed" between 1 and 100
    # (which means each number has the same probability of occurring)

    print("Uniformly random numbers in a given range \n")

    for i in range(10):
        number = randrange(1,101)
        print("{0:2}         {1:3}".format(i+1,number))

    # now generate 10 numbers that are uniformly distributed between 0 and 1.

    print("\n\nUniformly random numbers between 0 and 1 \n")

    for i in range(10):
        number = random()
        print("{0:2}         {1:0.6f}".format(i+1,number))

main()
```

#2.py

Want to toss a coin and find out how many tosses are required for it to land "H" (heads).

Let the user input `p`, where `p = probability that on a any toss the coin comes up H`.

The number of tosses until the coin comes up H is called a GEOMETRIC random variable.
#Since the probability is p, it is called a geometric(p) random variable.

```python
from random import *

def toss(p):
    if (random() <= p):
        return("H")
    else:
        return("T")

def main():
    p = eval(input("Probability of H on any toss for your coin? "))
    n = eval(input("How many coin-tossing experiments? " ))
    for i in range(n):
        seq =[]
        while(1):
            result = toss(p)
            seq.append(result)
            if (result == "H"): break

        print(seq)
    print()

main()

...

Let X = number of tossed to get H on any given experiment.

Then P(X = k ) = probability that the number of tries to get H is k (i.e., that's the meaning)

\[
\begin{align*}
    P(X = k) &= (1-p)^{k-1} \cdot p \\
    &= (1-p)^{k-1} \cdot p \\
\end{align*}
\]

for k = 1,2,3,4,5,....... 

This tells you the exact probability that k tries are required...
```
#3.py

# Just to remind you that we can pass a list to a function and change it

def emptyout(list):
    while(len(list)>0):
        list.pop()  # unless you specify an index, it removes the first item

def main():

    items=["1","5","22","91","7"]

    print("Before function call\n")
    print(items)

    emptyout(items)

    print("\n\nAfter function call\n")
    print(items)

#4.py

# Let's put n distinct numbers in a list and "permute" the list k times.
# To "permute" is to make one arrangement. With n objects there are a total of
n! arrangements.

# We won't try to list all the arrangements. You can do this for homework,
# by extending the
# idea shown here just a bit.

# In our example we don't check that a permuted list has not been repeated.
# For your homework
# problem you can do this check, but it would be taxing. It's better to find
# lists in order (as if you were counting)

from random import *

def permute(a):
    n = len(a)
output = []

while(len(output) < n):
    index = randrange(1,n-len(output)+1)
    #       print(index, " out of ", n-len(output))      tells which elem of
    # current ist is selected next
    output.append(a.pop(index-1))

return(output)

def main():

    n = eval(input("How many distinct numbers in your list? "))
    list=[]
    print("Enter the ", n," numbers \n") # hit return after each number
    entered
    for i in range(n):
        next = eval(input())
        list.append(next)
    print("\n",list,\\"\n")
    m = eval(input("How many permutations do you want? "))
    for j in range(m):
        # copy list into dup list
        dup = []
        for k in range(len(list)):  dup.append(list[k])
        print(permute(dup))

main()

#_________________________________________________________________
#5.py
#This shows the inner-workings of a linear congruential (uniform) random
#number generator.

from math import *

def uniform(seed):

http://courses.cs.purdue.edu/
d2p31m = 2147483647
d2p31 = 2147483711

seed[0] = 16807*seed[0] - floor(16807*(seed[0])/d2p31m) * d2p31m

return(fabs((seed[0]/d2p31)))

def main():
    seed=[1234567]  #use any 7-10 digit number as the initial seed and
    #then do not
    #touch the seed. Uniform() will do the
    #result.

    for i in range(100):
        print(i,uniform(seed))