***Python***

from z1 import \*  
import sys # za argumente komandne linije  
import random

import string  
  
def f(x):  
 sum = 0  
 i = 1  
 while sum < int(x):  
 sum += i  
 i+=1  
 return sum, i  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # text = input("prompt")  
  
 x = sys.argv[1]  
 text = input("Unos: ")  
 if text.isdigit():  
 print("Is a number")  
  
 print(random.randint(0,9))

res = ''.join(random.choices(string.ascii\_uppercase + string.digits, k=5))

l10 = []

for i in range(10):  
 l10.append(random.randint(1, 50))

l10.reverse() # invert list

**Lista**

l = [] # kreiranje prazne liste

l = [1, 2, ‘a’, “b”]  
l.append(x) # dodavanje novog elementa na kraj liste  
print(len(l)) # vraca duzinu liste  
l.insert(x, i) # dodaje element x na index i  
l.remove(x) # brise element  
del l[i] # brise element sa indeksom i  
clear() # brise sve elemente iz liste

**Recnik (Dictionary)**

d = {} # kreiranje praznog recnika  
d = {"key1" : 1, "key2": 2}  
d["key2"] = 0 # menjanje vrednosti elementa  
d["key3"] = 3 # dodavanje novog elementa  
del d["key1"] # brisanje elementa  
list(d) # vraca listu svih kljuceva u recniku

**Rad sa datotekama**

import sys  
  
fin = open("input\_file\_name", "r")  
fout = open("output\_file\_name", "w")  
  
for line in fin:  
 fout.write(line)  
  
fin.close()  
fout.close()

**Torka (Tuple)**

t = ((),) # definisanje torke (tuple)  
t = ((1, 2.3, "Jedan"), (2, 3.4, "Dva"), (3, 4.5, "Tri"))  
print(t[0]) # ispisivanje

t += (1233, “nnsasd”)

**Time**

import time  
  
# Measure execution time:  
# float value in seconds: time.perf\_counter()  
# integer value in nanoseconds: time.perf\_counter\_ns()  
  
start\_time = time.perf\_counter()  
# do some processing...  
end\_time = time.perf\_counter()  
  
print('Execution time is', end\_time-start\_time)

**Sort**

def insertion\_sort(L):  
 for j in range(1, len(L)):  
 key = L[j]  
 i = j - 1  
 while i >= 0 and key < L[i]:  
 L[i+1] = L[i]  
 i -= 1  
 L[i+1] = key  
  
def bubble\_sort(L):  
 for i in range(len(L)-1):  
 for j in range(0, len(L)-i-1):  
 if L[j] > L[j+1]:  
 L[j+1], L[j] = L[j], L[j+1]

def linear\_search(ip\_arr, x):  
 for i in range(len(ip\_arr)):  
 if ip\_arr[i] == x:  
 return True  
 return False

def partition(arr, l, h):  
 x = arr[h]  
 i = l - 1  
 for j in range(l, h):  
 if arr[j] <= x:  
 i = i + 1  
 (arr[i], arr[j]) = (arr[j], arr[i])  
 (arr[i+1], arr[h]) = (arr[h], arr[i+1])  
 return i + 1  
  
def quick\_sort(arr, l, h):  
 if l < h:  
 pi = partition(arr, l, h)  
 quick\_sort(arr, l, pi-1)  
 quick\_sort(arr, pi+1, h)

def countingSort(array):  
 size = len(array)  
 output = [0] \* size  
  
 # Initialize count array  
 count = [0] \* 10  
  
 # Store the count of each elements in count array  
 for i in range(0, size):  
 count[array[i]] += 1  
  
 # Store the cummulative count  
 for i in range(1, 10):  
 count[i] += count[i - 1]  
  
 # Find the index of each element of the original array in count array  
 # place the elements in output array  
 i = size - 1  
 while i >= 0:  
 output[count[array[i]] - 1] = array[i]  
 count[array[i]] -= 1  
 i -= 1  
  
 # Copy the sorted elements into original array  
 for i in range(0, size):  
 array[i] = output[i]

def bucketSort(array):  
 bucket = []  
  
 # Create empty buckets  
 for i in range(len(array)):  
 bucket.append([])  
  
 # Insert elements into their respective buckets  
 for j in array:  
 index\_b = int(10 \* j)  
 bucket[index\_b].append(j)  
  
 # Sort the elements of each bucket  
 for i in range(len(array)):  
 bucket[i] = sorted(bucket[i])  
  
 # Get the sorted elements  
 k = 0  
 for i in range(len(array)):  
 for j in range(len(bucket[i])):  
 array[k] = bucket[i][j]  
 k += 1  
 return array