PYTHON CODES - CANON LIST

Prepared by (LI): ganga-babu200495

List to Dict:

```
lst = [1,1,1,2,2,2]
lst = ['a',1,'b',2,'c',3]
it = iter(lst)
dct = dict(zip(it,it))
print(dct)
#note : here dict doesn't allow duplicate
```

Justify why tuple is immutable & list is mutable:

```
tuple_0 = (1,2,3)
print('ID of tuple : '+ str(id(tuple_0)))
print(tuple_0)
tuple_0 += (4,5)
print('ID of tuple : '+ str(id(tuple_0)))
print(tuple_0)

List_0 = [1,2,3]
print('ID of List_0 : '+ str(id(List_0)))
print(List_0)
List_0 += [4,5]
print('ID of List_0 : '+ str(id(List_0)))
print(List_0)
```

Fibonacci code:

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
print("Fibonacci sequence:")
while count < nterms:
   print(n1)
   nth = n1 + n2
   n1 = n2
   n2 = nth
   count += 1
```

Division Operator:

```
x,y = 7,5
print(x/y)
# True division o/p decimal quotient
print(x//y)
# Floor division o/p integer quotient
print(x%y)
# modulus operator o/p remainder
```

Quadratic Equation:

```
# x = -b +- sqrt(b**2 - 4ac)/2a
a,b,c = 1,4,2

# find value of discriminant
D = (b**2) - (4*a*c)
D = D**.5

sol1 = (-b+D)/(2*a)
sol2 = (-b-D)/(2*a)
print('Roots of the quadratic equations are', sol1,'&', sol2)
```

Swap numbers:

```
x,y=5,6
temp=x
x=y
y=temp
x,y
```

n is +ve or -ve:

```
num = int(input())
if num > 0:
  print('+ve')
elif num == 0:
  print('0')
else :
  print('-ve')
```

n is Odd or Even:

```
num = int(input())
if (num % 2) == 0:
   print("Even")
else:
   print("Odd")
```

Sum of Natural N:

```
n = 16
n= int(n*(n+1)/2)
print(n)
```

Divisible Numbers by n:

```
my_list = list(range(1,101))
n = int(input('assign a divisible number : '))
r = list(filter(lambda x: (x % n == 0), my_list))
print(r)
```

Check Leap Year:

```
year = int(input('Mention the year : '))
if (year % 400 ==0) and (year % 100 ==0):
  print('{0} is a leap year and a century
        year'.format(year))
elif (year % 4 ==0) and (year % 100 !=0):
  print('{0} is a leap year and not a century
        year'.format(year))
else:
  print('{0} is not a leap year'.format(year))
```

Largest Among Three Numbers:

```
n1,n2,n3 = 10,12,20
if n1>n2 and n1>n3:
  larget = n1
elif n2>n1 and n2 >n3:
  larget = n3
else:
  larget = n3
print(larget)
```

Check Prime Number:

```
n = int(input())
Flag = False
if n>1:
  for i in range(2,n):
    if (n % i ==0):
      Flag = True
      break
if Flag:
  print(n,'is not a prime number')
else:
  print(n,'is a prime number')
```

Factorial of a Number:

```
n=int(input())
f=1
if n<0:
  print('-ve assignment not allowed')
elif n==0:
  print('0 not allowed')
else:
  for i in range(1,n+1):
    f = f*i
  print('Factorial of ', i, 'is',f)
```

Find Armstrong Number in an Interval:

```
lower = 100
upper = 2000
for num in range(lower, upper+1):
  order, sum = len(str(num)),0
  temp = num
  while temp > 0:
    digit = temp % 10
    sum += digit**order
    temp //=10
  if num == sum:
    print(num)
```

Find LCM using GCD:

```
x,y = 54,24
def c_gcd(x,y):
  while(y):
    x,y = y, x%y
  return x
def lcm(x,y):
  lcm = (x*y)//c_gcd(x,y)
  return lcm
lcm(x,y)
```

Factors of a number:

Numbers that divide the original number evenly or exactly

```
n = int(input())
print('Factor of {0} are :'.format(n))
for i in range(1, n+1):
  if n % i == 0:
    print(i)
```

Fibonacci Sequence Using Recursion:

```
def fsr(n):
  if n <=1:
    return n
  else:
    return(fsr(n-1) + fsr(n-2))
l = 10
if l <=0:
  print('Enter a positive number ')
else:
  for i in range(l):
    print(fsr(i))
```

Convert Decimal to Binary Using Recursion:

```
def d_bc(n):
  if n>1:
    d_bc(n//2)
  print(n%2, end='')
d = int(input())
d_bc(d)
```

Add Two Matrices:

```
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

Transpose Matrices using Nested List Comprehension:

```
result = [[X[j][i] for j in range(len(X))] for i in
                   range(len(X[0]))]
for r in result:
  print(r)
```

Multiply Two Matrices using list comprehension:

```
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
result = [[sum(a*b for a,b in zip(X_row,Y_col))
              for Y_col in zip(*Y)] for X_row in X]
for r in result:
   print(r)
```

Check Whether a String is Palindrome or Not:

```
my_str = 'albohPhoBiA'
my_str = my_str.casefold()
rev = reversed(my_str)
if list(my_str)== list(rev):
  print('yes')
else:
  print('No')
```

Punctuations From a String:

```
import re
my_str = 'Python, is best : for ! \
        Programming. %^&**#!@#$%^&\
        *()+;{}|[]\;:,./<>?/-'

res = re.sub(r'[^\w\s]','',my_str)
print(res)
```

Sort Words in Alphabetic Order:

```
my_str = "Hello this Is an Example With\
          cased letters"
words = [W.lower() for W in my_str.split()]
words.sort()
for i in words:
  print(i)
```

Flatten a Nested List:

```
# Using List Comprehension:
L = [[1], [2, 3], [4, 5, 6, 7]]
r = [n for sublist in L for n in sublist]
print(r)

# Using itertools package:
import itertools
r= list(itertools.chain(*L))
print(r)

# Using sum() :
r = sum(my_list, [])
print(r)
```

Split a List Into Evenly Sized Chunks:

```
import numpy as np
my_list = [1,2,3,4,5,6,7,8,9]
print(np.array_split(my_list, 5))
```

Convert String to Datetime:

```
from datetime import datetime
my_date_string = "Mar 11 2011 11:31AM"
r = datetime.strptime(
   my_date_string, '%b %d %Y %I:%M%p')
print(r)
```

Count an Item in a List:

```
freq = ['a', 1, 'a', 4, 3, 2, 'a'].count('a')
print(freq)
```

Reverse a Number:

```
# using while Loop
n,rn = 1234,0
while n != 0:
   d = n % 10
   rn = rn * 10 + d
   n //= 10
print(rn)

# using slicing
n = 123456
r = int(str(n)[::-1])
print(r)
```

Count number of Digits:

```
c,n = 0,3452
while n != 0:
   n //= 10
   c += 1
print(c)
```

Remove Duplicates From a List:

```
list_1 = [1, 2, 1, 4, 6]
print(list(set(list_1)))
# between lists
list_2 = [7, 8, 2, 1]
print(list(set(list_1) ^ set(list_2)))
```

Check If Two Strings are Anagram:

```
str1, str2 = "Act", "Cat"
str1, str2 = str1.lower(), str2.lower()
if(len(str1) == len(str2)):
   sorted_str1 = sorted(str1)
   sorted_str2 = sorted(str2)
   if(sorted_str1 == sorted_str2):
      print("anagram.")
   else:
      print("not anagram.")
else:
   print("not anagram.")
```

Compute Permutations of a String:

```
from itertools import permutations as p
words = [''.join(i) for i in p('ABC')]
print(words)
```

Data frame Slicing:

```
import pandas as pd
df = pd.DataFrame({"A":[12, 4, 5, None, 1],
                   "B":[7, 2, 54, 3, None],
                   "C":[20, 16, 11, 3, 8],
                   "D":[14, 3, None, 2, 6]})
df.loc[:, ["A","C"]]
df.loc[:,'A':'C']
```

Lambda function:

```
a = lambda x,y : x+y
print(a(5, 6))
```

Add values to a python array:

```
a=[1.1 , 2.1 ,3.1]
a.append(3.4)
a.extend([4.5,6.3,6.8])
a.insert(2,3.8)
```

Remove values on python array:

```
a=[1.1 , 2.1 ,3.1]
print(a.pop())
print(a.pop(3))
a.remove(1.1)
print(a)
```

pyramid pattern:

```
def pyfunc(r):
 for x in range(r):
   print(' '*(r-x-1)+'*'*(2*x+1))
pyfunc(9)
```

Python Tricks for Competitive Coding: To find top 3 elements and their counts:

```
from collections import Counter
arr = [1, 3, 4, 1, 2, 1, 1, 3, 4, 3, 5, 1, 2, 5, 3, 4, 5]
counter = Counter(arr)
top_three = counter.most_common(3)
print(top_three)
```

Find 3 largest and 4 smallest elements of a list using heapq

```
import heapq
grades = [110, 25, 38, 49, 20, 95, 33, 87, 80, 90]
print(heapq.nlargest(3, grades))
print(heapq.nsmallest(4, grades))
```

Demonstrate use of zip

```
stocks = {
  'Goog' : 520.54,
  'FB' : 76.45,
  'yhoo' : 39.28,
  'AMZN' : 306.21,'APPL' : 99.76
  }
zipped_1 = zip(stocks.keys(), stocks.values())
print(sorted(zipped_1))
```

Apply map function on a list:

```
income = [10, 30, 75]
def double_money(dollars):
  return dollars * 2
new_income = list(map(double_money, income))
print(new_income)
```

Concatenation of list of strings

```
string = " "
lst = ["Geeks", "for", "Geeks"]
for i in lst:
    string += i
----------or----------
string = ' '.join(lst)
print(string)
```

PYTHON CODES - CANON LIST

Prepared by (LI): ganga-babu200495

List to Dict:

```
lst = [1,1,1,2,2,2]
lst = ['a',1,'b',2,'c',3]
it = iter(lst)
dct = dict(zip(it,it))
print(dct)
#note : here dict doesn't allow duplicate
```

Justify why tuple is immutable & list is mutable:

```
tuple_0 = (1,2,3)
print('ID of tuple : '+ str(id(tuple_0)))
print(tuple_0)
tuple_0 += (4,5)
print('ID of tuple : '+ str(id(tuple_0)))
print(tuple_0)

List_0 = [1,2,3]
print('ID of List_0 : '+ str(id(List_0)))
print(List_0)
List_0 += [4,5]
print('ID of List_0 : '+ str(id(List_0)))
print(List_0)
```

Fibonacci code:

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
print("Fibonacci sequence:")
while count < nterms:
   print(n1)
   nth = n1 + n2
   n1 = n2
   n2 = nth
   count += 1
```

Division Operator:

```
x,y = 7,5
print(x/y)
# True division o/p decimal quotient
print(x//y)
# Floor division o/p integer quotient
print(x%y)
# modulus operator o/p remainder
```

Quadratic Equation:

```
# x = -b +- sqrt(b**2 - 4ac)/2a
a,b,c = 1,4,2

# find value of discriminant
D = (b**2) - (4*a*c)
D = D**.5

sol1 = (-b+D)/(2*a)
sol2 = (-b-D)/(2*a)
print('Roots of the quadratic equations are', sol1,'&', sol2)
```

Swap numbers:

```
x,y=5,6
temp=x
x=y
y=temp
x,y
```

n is +ve or -ve:

```
num = int(input())
if num > 0:
  print('+ve')
elif num == 0:
  print('0')
else :
  print('-ve')
```

n is Odd or Even:

```
num = int(input())
if (num % 2) == 0:
   print("Even")
else:
   print("Odd")
```

Sum of Natural N:

```
n = 16
n= int(n*(n+1)/2)
print(n)
```

Divisible Numbers by n:

```
my_list = list(range(1,101))
n = int(input('assign a divisible number : '))
r = list(filter(lambda x: (x % n == 0), my_list))
print(r)
```

Check Leap Year:

```
year = int(input('Mention the year : '))
if (year % 400 ==0) and (year % 100 ==0):
  print('{0} is a leap year and a century
          year'.format(year))
elif (year % 4 ==0) and (year % 100 !=0):
  print('{0} is a leap year and not a century
          year'.format(year))
else:
  print('{0} is not a leap year'.format(year))
```

Largest Among Three Numbers:

```
n1,n2,n3 = 10,12,20
if n1>n2 and n1>n3:
  larget = n1
elif n2>n1 and n2 >n3:
  larget = n3
else:
  larget = n3
print(larget)
```

Check Prime Number:

```
n = int(input())
Flag = False
if n>1:
  for i in range(2,n):
    if (n % i ==0):
      Flag = True
      break
if Flag:
  print(n,'is not a prime number')
else:
  print(n,'is a prime number')
```

Factorial of a Number:

```
n=int(input())
f=1
if n<0:
  print('-ve assignment not allowed')
elif n==0:
  print('0 not allowed')
else:
  for i in range(1,n+1):
    f = f*i
  print('Factorial of ', i, 'is',f)
```

Find Armstrong Number in an Interval:

```
lower = 100
upper = 2000
for num in range(lower, upper+1):
  order, sum = len(str(num)),0
  temp = num
  while temp > 0:
    digit = temp % 10
    sum += digit**order
    temp //=10
  if num == sum:
    print(num)
```

Find LCM using GCD:

```
x,y = 54,24
def c_gcd(x,y):
  while(y):
    x,y = y, x%y
  return x
def lcm(x,y):
  lcm = (x*y)//c_gcd(x,y)
  return lcm
lcm(x,y)
```

Factors of a number:

Numbers that divide the original number evenly or exactly

```
n = int(input())
print('Factor of {0} are :'.format(n))
for i in range(1, n+1):
  if n % i == 0:
    print(i)
```

Fibonacci Sequence Using Recursion:

```
def fsr(n):
  if n <=1:
    return n
  else:
    return(fsr(n-1) + fsr(n-2))
l = 10
if l <=0:
  print('Enter a positive number ')
else:
  for i in range(l):
    print(fsr(i))
```

Convert Decimal to Binary Using Recursion:

```
def d_bc(n):
  if n>1:
    d_bc(n//2)
  print(n%2, end='')
d = int(input())
d_bc(d)
```

Add Two Matrices:

```
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]
result = [[0,0,0],
         [0,0,0],
         [0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

Transpose Matrices using Nested List Comprehension:

```
result = [[X[j][i] for j in range(len(X))] for i in
                range(len(X[0]))]
for r in result:
  print(r)
```

Multiply Two Matrices using list comprehension:

```
result = [[0,0,0],
         [0,0,0],
         [0,0,0]]
result = [[sum(a*b for a,b in zip(X_row,Y_col))
              for Y_col in zip(*Y)] for X_row in X]
for r in result:
   print(r)
```

Check Whether a String is Palindrome or Not:

```
my_str = 'albohPhoBiA'
my_str = my_str.casefold()
rev = reversed(my_str)
if list(my_str)== list(rev):
  print('yes')
else:
  print('No')
```

Punctuations From a String:

```
import re
my_str = 'Python, is best : for ! \
        Programming. %^&**#!@#$%^&\
        *()+;{}|[]\;:,./<>?/-'

res = re.sub(r'[^\w\s]','',my_str)
print(res)
```

Sort Words in Alphabetic Order:

```
my_str = "Hello this Is an Example With\
          cased letters"
words = [W.lower() for W in my_str.split()]
words.sort()
for i in words:
  print(i)
```

Flatten a Nested List:

```
# Using List Comprehension:
L = [[1], [2, 3], [4, 5, 6, 7]]
r = [n for sublist in L for n in sublist]
print(r)

# Using itertools package:
import itertools
r= list(itertools.chain(*L))
print(r)

# Using sum() :
r = sum(my_list, [])
print(r)
```

Split a List Into Evenly Sized Chunks:

```
import numpy as np
my_list = [1,2,3,4,5,6,7,8,9]
print(np.array_split(my_list, 5))
```

Convert String to Datetime:

```
from datetime import datetime
my_date_string = "Mar 11 2011 11:31AM"
r = datetime.strptime(
   my_date_string, '%b %d %Y %I:%M%p')
print(r)
```

Count an Item in a List:

```
freq = ['a', 1, 'a', 4, 3, 2, 'a'].count('a')
print(freq)
```

Reverse a Number:

```
# using while Loop
n,rn = 1234,0
while n != 0:
   d = n % 10
   rn = rn * 10 + d
   n //= 10
print(rn)

# using slicing
n = 123456
r = int(str(n)[::-1])
print(r)
```

Count number of Digits:

```
c,n = 0,3452
while n != 0:
   n //= 10
   c += 1
print(c)
```

Remove Duplicates From a List:

```
list_1 = [1, 2, 1, 4, 6]
print(list(set(list_1)))
# between lists
list_2 = [7, 8, 2, 1]
print(list(set(list_1) ^ set(list_2)))
```

Check If Two Strings are Anagram:

```
str1, str2 = "Act", "Cat"
str1, str2 = str1.lower(), str2.lower()
if(len(str1) == len(str2)):
   sorted_str1 = sorted(str1)
   sorted_str2 = sorted(str2)
   if(sorted_str1 == sorted_str2):
      print("anagram.")
   else:
      print("not anagram.")
else:
   print("not anagram.")
```

Compute Permutations of a String:

```
from itertools import permutations as p
words = [''.join(i) for i in p('ABC')]
print(words)
```

Data frame Slicing:

```
import pandas as pd
df = pd.DataFrame({"A":[12, 4, 5, None, 1],
                   "B":[7, 2, 54, 3, None],
                   "C":[20, 16, 11, 3, 8],
                   "D":[14, 3, None, 2, 6]})
df.loc[:, ["A","C"]]
df.loc[:,'A':'C']
```

Lambda function:

```
a = lambda x,y : x+y
print(a(5, 6))
```

Add values to a python array:

```
a=[1.1 , 2.1 ,3.1]
a.append(3.4)
a.extend([4.5,6.3,6.8])
a.insert(2,3.8)
```

Remove values on python array:

```
a=[1.1 , 2.1 ,3.1]
print(a.pop())
print(a.pop(3))
a.remove(1.1)
print(a)
```

pyramid pattern:

```
def pyfunc(r):
 for x in range(r):
   print(' '*(r-x-1)+'*'*(2*x+1))
pyfunc(9)
```

Python Tricks for Competitive Coding: To find top 3 elements and their counts:

```
from collections import Counter
arr = [1, 3, 4, 1, 2, 1, 1, 3, 4, 3, 5, 1, 2, 5, 3, 4, 5]
counter = Counter(arr)
top_three = counter.most_common(3)
print(top_three)
```

Find 3 largest and 4 smallest elements of a list using heapq

```
import heapq
grades = [110, 25, 38, 49, 20, 95, 33, 87, 80, 90]
print(heapq.nlargest(3, grades))
print(heapq.nsmallest(4, grades))
```

Demonstrate use of zip

```
stocks = {
  'Goog' : 520.54,
  'FB' : 76.45,
  'yhoo' : 39.28,
  'AMZN' : 306.21,'APPL' : 99.76
  }
zipped_1 = zip(stocks.keys(), stocks.values())
print(sorted(zipped_1))
```

Apply map function on a list:

```
income = [10, 30, 75]
def double_money(dollars):
  return dollars * 2
new_income = list(map(double_money, income))
print(new_income)
```

Concatenation of list of strings

```
string = " "
lst = ["Geeks", "for", "Geeks"]
for i in lst:
    string += i
----------or----------
string = ' '.join(lst)
print(string)
```