Tuples

Tuples

sequence of immutable Python objects

tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

empty tuple −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma −

```
a = (50)                    # an integer

tup1 = (50,);               # tuple containing an integer
```

tuple indices start at 0

```
print ("tup1[0]: ", tup1[0]) # print physics

print ("tup2[1:5]: ", tup2[1:5]) # print (2,3,4,5)
```

Tuples in Action

```
>>> (1, 2) + (3, 4)   # Concatenation
(1, 2, 3, 4)

>>> (1, 2) * 4                # Repetition
(1, 2, 1, 2, 1, 2, 1, 2)

>>> T = (1, 2, 3, 4)          # Indexing, slicing

>>> T[0], T[1:3]
(1, (2, 3))
```

Sorted method in Tuples

```
>>> tmp = ['aa', 'bb', 'cc', 'dd']

>>> T = tuple(tmp)

 # Make a tuple from the list's items

>>> T
('aa', 'bb', 'cc', 'dd')

>>> sorted(T)   #Return a list of items
# Since tuples are immutable

['aa', 'bb', 'cc', 'dd‘]
```

Index method can be used to find the position of particular value in the tuple.

```
>>> T = (1, 2, 3, 2, 4, 2)
>>> T.index(2)          # Offset of first appearance of 2
1

>>> T.count(2)          # How many 2s are there?
3
```

```
>>> bob = ('Bob', 40.5, ['dev', 'mgr'])
# Tuple record

>>> bob
('Bob', 40.5, ['dev', 'mgr'])

>>> bob[0], bob[2]                    # Access by position
('Bob', ['dev', 'mgr'])
```

Delete Tuple Elements

Removing individual tuple elements is not possible

But possible to remove an entire tuple

```
tup = ('physics', 'chemistry', 1997, 2000);

print (tup)
del tup;
print ("After deleting tup : ")
print (tup)
```

Basic Tuples Operations

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Indexing, Slicing

If L = ('spam', 'Spam', 'SPAM!')

Python Expression	Results	Description
L[2]	'SPAM!'	Offsets start at zero
L[-2]	'Spam'	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

Built-in Tuple Functions

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')

len(tuple1)
2
```

When we have numerical tuple:

```
t1 = (1,2,3,7,4)

max(t1)

min(t1)
```

Converts a list into a tuple

tuple(seq)

t2=tuple([2,4])

```
>>> t2
(2, 4)
```

For reading a tuple values using for loop

```
t=()
n=int(input())
for i in range(0,n):
    s = int(input())
    t = t+(s,)
print(t)
```

For reading a tuple values using for loop

```
n=int(input())
t=()
l=[]
for i in range(0,n):
    s=int(input())
    l.append(s)
t=tuple(l)
print(l)
print(t)
```

Problem:

Create a tuple T with N numbers.
Extract the numbers that are multiples of 5 from T and store it in another tuple K.

Read N values in a tuple

```
t=()


n=int(input("Enter the number of elements"))
print("Enter the numbers")
for i in range(1,n+1):
    t+=(int(input()),)

Print(t)
```

```
t=()
k=()
n=int(input("Enter the number of elements"))
print("Enter the numbers")
for i in range(1,n+1):
    t+=(int(input()),)

for x in t:
    if(x%5==0):
        k+=(x,)
print("Multiples of 5 \n ")
print(k)
```

```
l=[]
t=()
n=int(input("Enter the number of elements"))
print("Enter the numbers")
for i in range(1,n+1):
    t+=(int(input()),)

for x in t:
    if(x%5==0):
            l.append(x)
print("Multiples of 5 \n "+str(l))
```

Problem

A hospital has received a set of lab reports. Totally five tests are conducted in the lab and the report is prepared in such a way that the ‘n^{th}’ number correspond to value of $test_n$. If the test was not performed for the particular patient then ‘N’ is written in the corresponding place. Write a program to print if the test result is normal or not normal by using the values in Table 1. Since the value is sensitive, provide a mechanism so that the values do not get altered.

Name of the Test	Minimum Value	Maximum Value
Test1	20	30
Test2	35.5	40
Test3	12	15
Test4	120	150
Test5	80	120

```
READ lab_report
FOR i =0 to 5
        IF lab_report[i] != N
                IF lab_report[i] < min of test_i OR
                        lab_report[i] >max of test_i THEN
                        PRINT ‘abnormal’
                ELSE
                        PRINT ‘normal’
                END IF
        END IF
END FOR
```

```
report=()
minv=(20,35.5,12,120,80)
maxv=(30,40,15,150,120)
l=[]

print("Enter the values")
for i in range(0,5):
    report+=(input(),)

print(report)



print("Lab Report\n")

for i in range(0,5):
    if(report[i]=='N'):
        print("Test  "+str(i+1)+" is not performed")
        continue
    if((float(report[i])>=(minv[i])) and (float(report[i])<=(maxv[i]))):
        print("Test result "+str(i+1)+" is normal")
    else:
        print("Test result "+str(i+1)+" is abormal")
```

```
for i in range(0,5):
    if(report[i]=='N'):
        print("Test  "+str(i+1)+" is not performed")
        continue
    if((float(report[i])>=(minv[i])) and
(float(report[i])<=(maxv[i]))):
        print("Test result "+str(i+1)+" is normal")
    else:
        print("Test result "+str(i+1)+" is abormal")
```

Sets

an unordered collection of unique and immutable objects that supports operations corresponding to mathematical set theory

No duplicates

Sets are iterable, can grow and shrink on demand, and may contain a variety of object types

Does not support indexing

```
x = {1, 2, 3, 4}

y = {'apple','ball','cat'}

x1 = set('spam')  # Prepare set from a string
print (x1)

{'s', 'a', 'p', 'm'}

x1.add('alot')    # Add an element to the set
print (x1)

{'s', 'a', 'p', 'alot', 'm'}
```

Set Operations

Let S1 = {1, 2, 3, 4}

Union (|)

S2 = {1, 5, 3, 6} | S1

print(S2) # prints {1, 2, 3, 4, 5, 6}

Intersection (&)

S2 = S1 & {1, 3}

print(S2) # prints {1, 3}

Difference (-)

S2 = S1 - {1, 3, 4}

print(S2) # prints {2}

Super set (>)

S2 = S1 > {1, 3}

print(S2) # prints True

Empty sets must be created with the set built-in, and print the same way

S2 = S1 - {1, 2, 3, 4}
print(S2) # prints set() – Empty set

Empty curly braces represent empty dictionary but not set

In interactive mode – type({}) gives
<class 'dict'>

Reading values for SET

```
n=int(input("Enter number of elements in set1"))
s=set()
for i in range(n):
    val=int(input("Enter number"))
    s.add(val)
print(s)
print(len(s))
```

Reading values for SET

```
n=int(input("Enter number of elements in set1"))
s=set()
for i in range(n):
    val=int(input("Enter number"))
    s=s|{val}
print(s)
print(len(s))
```

Find the sum of unique elements present in the given list

```
L=[10,20,30,10,20,45,56,30]
s=set(L)
total=0
for x in s:
    total+=x
print(L)
print(s)
print(total)
```

```
n=int(input("Enter number of elements in set1"))
s1=set()
for i in range(n):
    val=int(input("Enter number"))
    s1.add(val)
print(s1)
print(len(s1))
n=int(input("Enter number of elements in set2"))
s2=set()
for i in range(n):
    val=int(input("Enter number"))
    s2.add(val)
print(s2)
print(len(s2))

print(s1|s2)
print(s1&s2)
print(s1-s2)
print(s2-s1)
```

Sum of the squares

```
n=int(input("Enter number of elements in set1"))
s1=set()
for i in range(n):
    val=int(input("Enter number"))
    s1.add(val)
print(s1)
print(len(s1))

total=0
for v in s1:
    total=total+v*v
print(total)
```

Reading values for SET and find the sum of squares of all numbers in set

```
n=int(input("Enter number of elements in set1"))
s=set()
for i in range(n):
    val=int(input("Enter number"))
    s.add(val)
print(s)
print(len(s))

summ=0
for i in s:
    summ=summ+i*i
print(summ)
```

Immutable constraints and frozen sets

Can only contain immutable object types

lists and dictionaries cannot be embedded in sets, but tuples can if you need to store compound values.

Tuples compare by their full values when used in set operations:

```
>>> S
{1.23}

>>> S.add([1, 2, 3])
TypeError
```

clear()

All elements will removed from a set.

```
>>> cities = {"Stuttgart", "Konstanz", "Freiburg"}
>>> cities.clear()
>>> cities
set() # empty
>>>
```

Copy

Creates a shallow copy, which is returned.

```
>>> more_cities = {"Winterthur","Schaffhausen","St. Gallen"}
>>> cities_backup = more_cities.copy()
>>> more_cities.clear()
>>> cities_backup  # copied value is still available
{'St. Gallen', 'Winterthur', 'Schaffhausen‘}
```

Just in case, you might think, an assignment might be enough:

```
>>> more_cities = {"Winterthur","Schaffhausen","St. Gallen"}
>>> cities_backup = more_cities #creates alias name
>>> more_cities.clear()
>>> cities_backup
set()
>>>
```

The assignment "cities_backup = more_cities" just creates a pointer, i.e. another name, to the same data structure.

To create frozenset:

```
cities = frozenset(["Frankfurt", "Basel","Freiburg"])

cities.add("Strasbourg")   #cannot modify

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
```

Set comprehensions

run a loop and collect the result of an expression on each iteration

result is a new set you create by running the code, with all the normal set behavior

```
>>> {x ** 2 for x in [1, 2, 3, 4]}
{16, 1, 4, 9}

>>> {x for x in 'spam'}
{'m', 's', 'p', 'a'}
```

Problem:

An University has published the results of the term end examination conducted in April. List of failures in physics, mathematics, chemistry and computer science is available. Write a program to find the number of failures in the examination

```
phy=['20mis1212','20mis1213','20mis1214']
chem=['20mis1210','20mis1212']
maths=['20mis1210','20mis1345','20mis1210','20mis1213
']
cse=['20mis1212','20mis1210','20mis1289']

s=set(phy)|set(chem)|set(maths)|set(cse)
print(s)
print(len(s))
```

Exercise 1

While John got ready for the office a shopping list was given by his son and daughter. While returning home John decides to buy the items from a shop. He decides to buy the items common in both the list, then the items listed only by his son and at last the items listed only by his daughter. Write a program to help him in achieving the task.