Syntax-2: [expression for variable in iterable if test]
Syntax-2 is used for filter data/creating list based on condition

```
# cerate a list with all the numbers from 1 to 50 which are divisible with 7
# without creating comprehension
list1=[]
for num in range(1,51):
    if num%7==0:
        list1.append(num)
print(list1)

# with comprehension
list2=[num for num in range(1,51) if num%7==0]
print(list2)
```

Output:
[7, 14, 21, 28, 35, 42, 49]
[7, 14, 21, 28, 35, 42, 49]

Example:

```
grades_list=[['naresh','A'],
             ['suresh','B'],
             ['kishore','C'],
             ['ramesh','A'],
             ['rajesh','B']]
grade_listA=[stud for stud in grades_list if stud[1]=='A']
grade_listB=[stud for stud in grades_list if stud[1]=='B']
grade_listC=[stud for stud in grades_list if stud[1]=='C']
print(grades_list)
print(grade_listA)
print(grade_listB)
print(grade_listC)
```

Output:

```
[['naresh', 'A'], ['suresh', 'B'], ['kishore', 'C'], ['ramesh', 'A'], ['rajesh', 'B']]
[['naresh', 'A'], ['ramesh', 'A']]
[['suresh', 'B'], ['rajesh', 'B']]
[['kishore', 'C']]
```

https://www.hackerrank.com/challenges/list-comprehensions/problem?isFullScreen=true

```
x = int(input()) # 1
y = int(input()) # 1
z = int(input()) # 1
n = int(input()) # 2
list1=[[i,j,k] for i in range(x+1)  for j in range(y+1)  for k in range(z+1) if i+j+k!=n]
print(list1)
```

tuple

Tuples are immutable sequences, typically used to store collections of heterogeneous data (such as the 2-tuples produced by the enumerate() built-in). Tuples are also used for cases where an immutable sequence of homogeneous data is needed (such as allowing storage in a set or dict instance).

Tuples may be constructed in a number of ways:

- Using a pair of parentheses to denote the empty tuple: ()
- Using a trailing comma for a singleton tuple: a, or (a,)
- Separating items with commas: a, b, c or (a, b, c)
- Using the tuple() built-in: tuple() or tuple(iterable)

```
>>> a=()
>>> print(type(a))
<class 'tuple'>
>>> a
()
>>> b=(10)
>>> print(type(b))
<class 'int'>
>>> c=(10,)
>>> print(c)
(10,)
>>> type(c)
<class 'tuple'>
>>> x=(10+20)
>>> print(x)
30
>>> type(x)
<class 'int'>
>>> y=(10+20,)
```

```
>>> print(y)
(30,)
>>> a=10,
>>> print(a)
(10,)
>>> print(type(a))
<class 'tuple'>
>>> b=10,20,30,40,50
>>> b
(10, 20, 30, 40, 50)
>>> print(type(b))
<class 'tuple'>
>>>
>>> t1=tuple()
>>> print(type(t1))
<class 'tuple'>
>>> t1.append(10)
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    t1.append(10)
AttributeError: 'tuple' object has no attribute 'append'
>>> t2=tuple(range(10,110,10))
>>> print(t2)
(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
>>> t2[0]=70
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    t2[0]=70
TypeError: 'tuple' object does not support item assignment
>>>
>>> t3=tuple([10,20,30,40,50])
>>> t4=tuple("PYTHON")
>>> t5=tuple((10,20,30,40,50))
>>> print(t3,t4,t5,sep="\n")
(10, 20, 30, 40, 50)
('P', 'Y', 'T', 'H', 'O', 'N')
(10, 20, 30, 40, 50)
>>>
```

What is difference between list and tuple?

List	Tuple
List is a mutable sequence type	Tuple is immutable sequence type
Created using []	Created using ()
Comprehension is allowed	Comprehension not allowed
It is not hashable and cannot used as objects inside set and dictionaries	It is hashable and can be used as objects inside set and dictionaries
“list” class represents list object	“tuple” class represents tuple object

https://csiplearninghub.com/important-practice-questions-of-list-in-python/

Set
set is unordered collection, where insertion order is not preserved