Syntax2: {key:value for variable in iterable if test}

Example:

```
grades_dict={1:['naresh','A'],
             2:['suresh','B'],
             3:['kiran','A'],
             4:['kishore','C'],
             5:['ramesh','B']}

print(grades_dict)
grade_dictA={rno:l  for rno,l in grades_dict.items() if l[1]=='A'}
grade_dictB={rno:l for rno,l in grades_dict.items() if l[1]=='B'}
grade_dictC={rno:l for rno,l in grades_dict.items() if l[1]=='C'}
print(grade_dictA)
print(grade_dictB)
print(grade_dictC)
```

Output:

```
{1: ['naresh', 'A'], 2: ['suresh', 'B'], 3: ['kiran', 'A'], 4: ['kishore', 'C'], 5:
['ramesh', 'B']}
{1: ['naresh', 'A'], 3: ['kiran', 'A']}
{2: ['suresh', 'B'], 5: ['ramesh', 'B']}
{4: ['kishore', 'C']}
>>>
```

Example:

```
sales_dict={2000:150000,
            2001:250000,
            2002:120000,
            2003:170000,
            2004:270000,
            2005:180000,
            2006:220000,
            2007:290000,
            2009:120000}

print(sales_dict)
sales_dict1={year:sales for year,sales in sales_dict.items() if
sales<200000}
print(sales_dict1)
```

```
sales_dict2={year:sales for year,sales in sales_dict.items() if
sales>=200000}
print(sales_dict2)
```

Output:

```
{2000: 150000, 2001: 250000, 2002: 120000, 2003: 170000, 2004: 270000,
2005: 180000, 2006: 220000, 2007: 290000, 2009: 120000}
{2000: 150000, 2002: 120000, 2003: 170000, 2005: 180000, 2009:
120000}
{2001: 250000, 2004: 270000, 2006: 220000, 2007: 290000}
>>>
```

Nested dictionary

We cannot write a dictionary inside dictionary because dictionary is having key and value pair

Inner dictionary can be represented as value

Example:

```
>>> student_data={1:{'name':'naresh','course':'python'},
            2:{'name':'kishore','course':'c'},
            3:{'name':'ramesh','course':'cpp'}}
>>> print(student_data)
{1: {'name': 'naresh', 'course': 'python'}, 2: {'name': 'kishore', 'course': 'c'}, 3:
{'name': 'ramesh', 'course': 'cpp'}}
>>> print(student_data[1])
{'name': 'naresh', 'course': 'python'}
>>> print(student_data[1]['name'])
naresh
>>> print(student_data[1]['course'])
python
>>> print(student_data[2]['name'])
kishore
>>> print(student_data[2]['course'])
c
>>>
```

What is difference between list, set and dictionary?

List	**Set**	**Dictionary/mapping**
List is a sequence	Set is non sequence	Dictionary is mapping
List is index based	Set is non index based	Dictionary is key based
List allows duplicates	Set does not allows	Dictionary does not

	duplicates	allows duplicate keys but allows duplicate values
Support indexing and slicing	Does not support indexing and slicing	Does not support indexing and slicing
Any objects can be added inside list	Only hashable objects are allowed	Keys are hashables and values can be any type
[] is used for creating list	{ele,} used for creating set	{key:value} used for creating dictionary
“list” class represents list object	“set” class represents set object	“dict” class represents dictionary object
In application development list is used to represent group individual object, where duplicates are allowed and accessing is done in seq and random	In application development set is used to represent group of individual objects where duplicates are not allowed and perform math set operations	In application developed dictionary is used represent data as key and value pair

Strings

What is string?

String is a collection of characters

String is a sequence data type/String is an immutable sequence data type.

Once the string is created we cannot do changes

String is sequence data type, which supports indexing and slicing

How to create string?

String is created in different ways

1. String is created by representing in single quotes
2. String is created by representing in double quotes
3. String is created by representing in triple quotes

The string which consist of only alphabets is called alphabetic string

The string which consist of alphabets and digits is called alphanumeric string

Within single quotes we can represent single line string

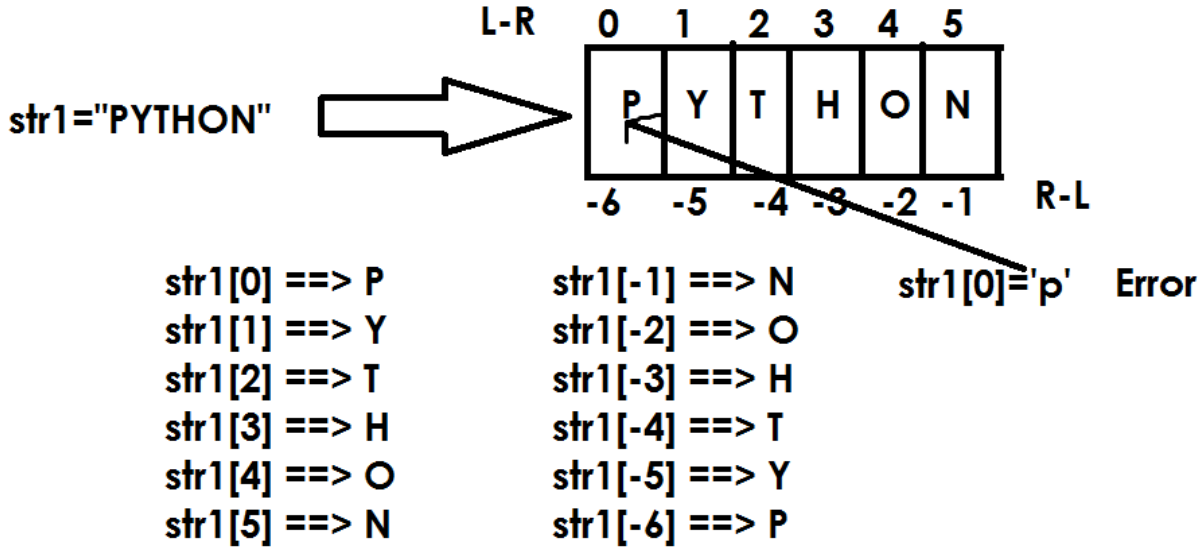
Within double quotes we can represent single line string

Within triple quotes we can represent multiline string/doc string/document string

```
>>> str1='python language'
>>> str2="python language"
>>> print(str1)
python language
>>> print(str2)
python language
>>> str3='python is "high level" language'
>>> str4="python is 'high level' language"
>>> print(str3)
python is "high level" language
>>> print(str4)
python is 'high level' language
>>> str4='''python is
high level,
object oriented,
programming
language'''
>>> print(str4)
python is
high level,
object oriented,
programming
language
>>>
```

String is sequence data type, we can read individual characters from string using,

1. Index
2. Slicing
3. Iterator
4. Enumerate
5. For loop




```
>>> str1="PYTHON"
>>> str1[0]='p'
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    str1[0]='p'
TypeError: 'str' object does not support item assignment
>>>
```

Example:

 # write a program find length of string without using predefined function

```
str1=input("enter any string") # python
c=0
for ch in str1:
    c+=1

print(f'length of string is {c}')
```

Output:

```
enter any stringpython
length of string is 6
>>>
```

Example:

```
# write a program to find input string is pal or not
str1=input("enter any string")
```

```
if str1==str1[::-1]:
    print(f'{str1} is pal')
else:
    print(f'{str1} is not pal')
```

Output:

```
enter any stringaba
aba is pal
>>>
```

You are given a string and your task is to *swap cases*. In other words, convert all lowercase letters to uppercase letters and vice versa.

```
str1=input("enter any string")
str2=''
for ch in str1:
    if ch>='A' and ch<='Z':
        ch=chr(ord(ch)+32)
        str2=str2+ch
    elif ch>='a' and ch<='z':
        ch=chr(ord(ch)-32)
        str2=str2+ch
    else:
        str2=str2+ch

print(str1)
print(str2)
```

Output:

```
enter any stringPyThoN
PyThoN
pYtHOn
>>>
```