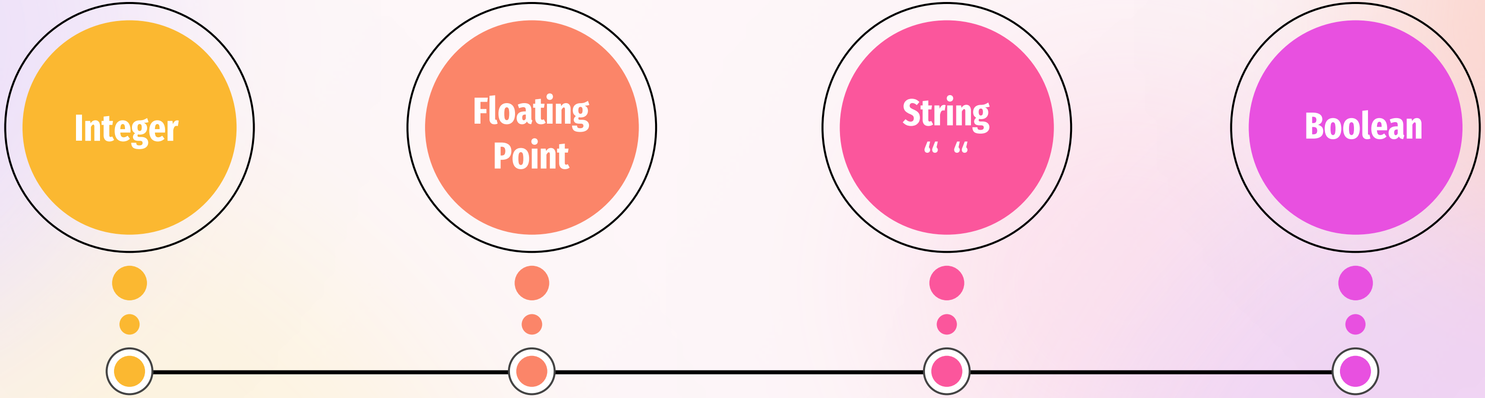
Introduction of Python - 1

Marina Hermaningsih

Data Type



Integer

Floating Point

String
“ “

Boolean

Whole numbers

Class = int
Slicing & Indexing are
not allowed

Decimal Numbers

Class = float
Slicing & Indexing are
not allowed

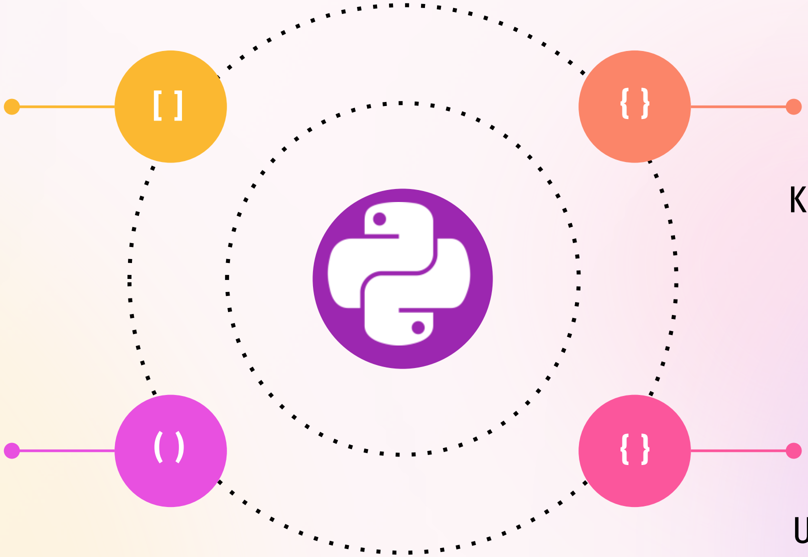
”Characters”

Class = str
Allow Slicing
& Indexing . Iterable

True or False

Class = bool
Slicing & Indexing are
not allowed

Data Type



List [ ]

Class : list

Ordered, mutable, iterable, allow duplicate, indexing, slicing,

Tuple ( )

Class : tuple

Ordered, immutable, iterable, allow duplicate, indexing, slicing

Dictionary {key : value }

Class : dict

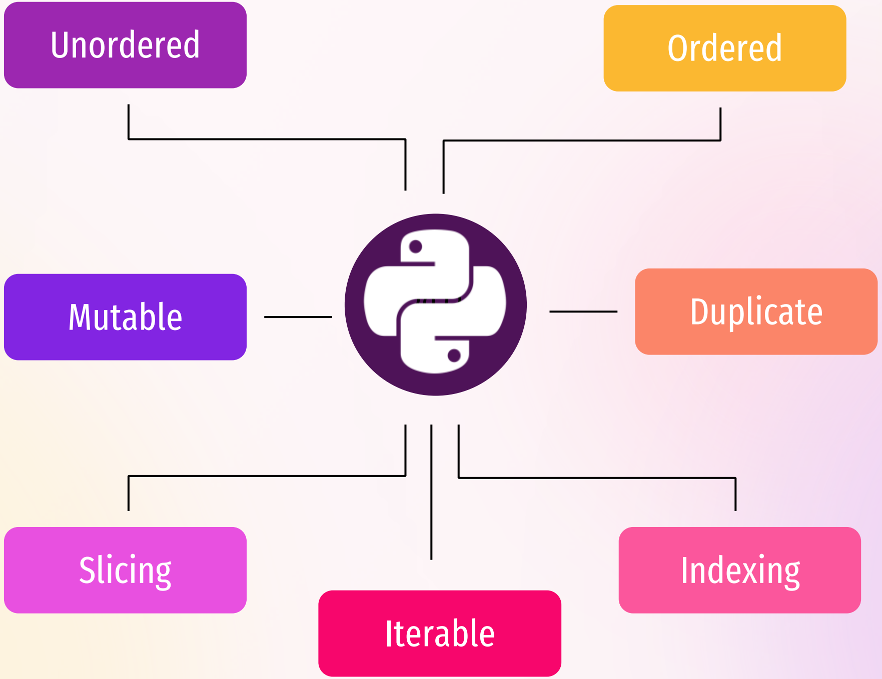
Keys are immutable, iterable Indxing and slicing are not allowed

Set { }

Class : set

Unordered, mutable,iterable duplicate, indexing and slicing are not allowed

Insight



Unordered

Ordered

Mutable

Duplicate

Slicing

Iterable

Indexing

Elements of data don’t have a defined order, that order will change after running

Elements of data have a defined order, that order will not change after running

Mutable means we can change the elements /items after it has been created

Allow duplicate means data can have two items with the same value

Return sequence of items/elements

Positif index starts 0 from left side, while negative index starts-1 from right index

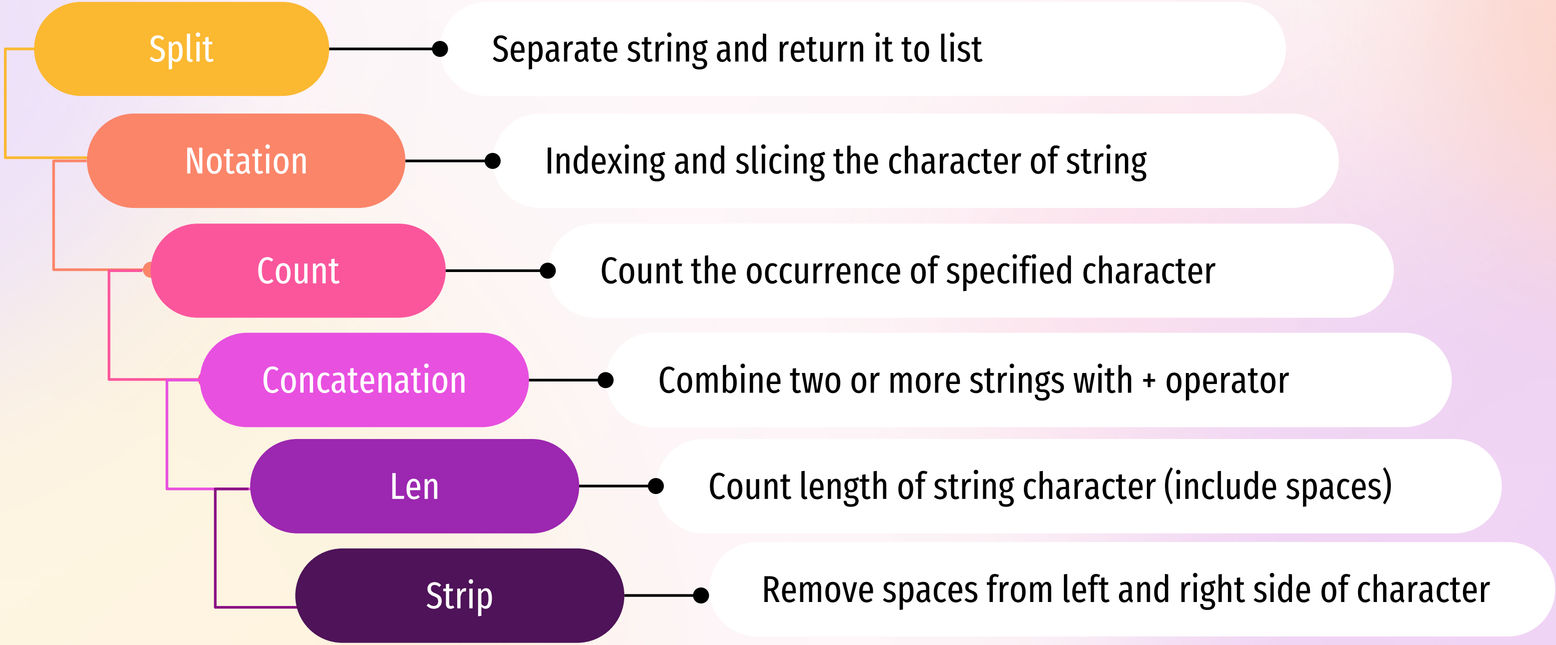
Iterable = Can be iterated by loop method

Python Operation

<table>
<tr><th>Arithmetic</th><th>Assignment</th><th>Comparison</th><th>Mathematical Set</th></tr>
<tr><td>x + y</td><td>x += y</td><td>x > y</td><td>Union
x | y</td></tr>
<tr><td>x - y</td><td>x -= y</td><td>x < y</td><td>Intersection
x & y</td></tr>
<tr><td>x * y</td><td>x *= y</td><td>x >= y</td><td>Difference
x – y != y – x</td></tr>
<tr><td>x ** y</td><td>x **= y</td><td>x <= y</td><td>Symmetric Difference
x ^ y</td></tr>
<tr><td>x / y</td><td>x /= y</td><td>x == y</td><td></td></tr>
<tr><td>x // y</td><td>x //= y</td><td>x != y</td><td></td></tr>
<tr><td>x % y</td><td>x %= y</td><td></td><td></td></tr>
</table>

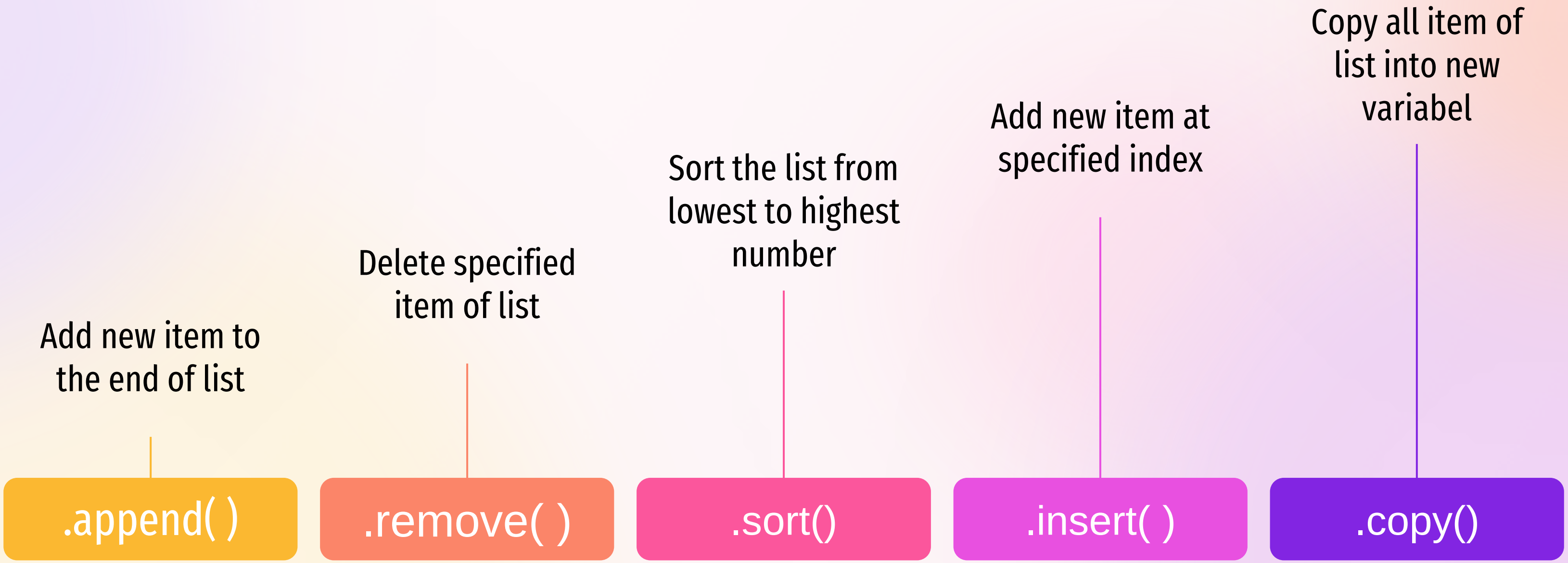
Assignment operation can also be applied for mathematical set operation

String Method



Split	Separate string and return it to list
Notation	Indexing and slicing the character of string
Count	Count the occurrence of specified character
Concatenation	Combine two or more strings with + operator
Len	Count length of string character (include spaces)
Strip	Remove spaces from left and right side of character

List Method



.append( )

Add new item to the end of list

.remove( )

Delete specified item of list

.sort()

Sort the list from lowest to highest number

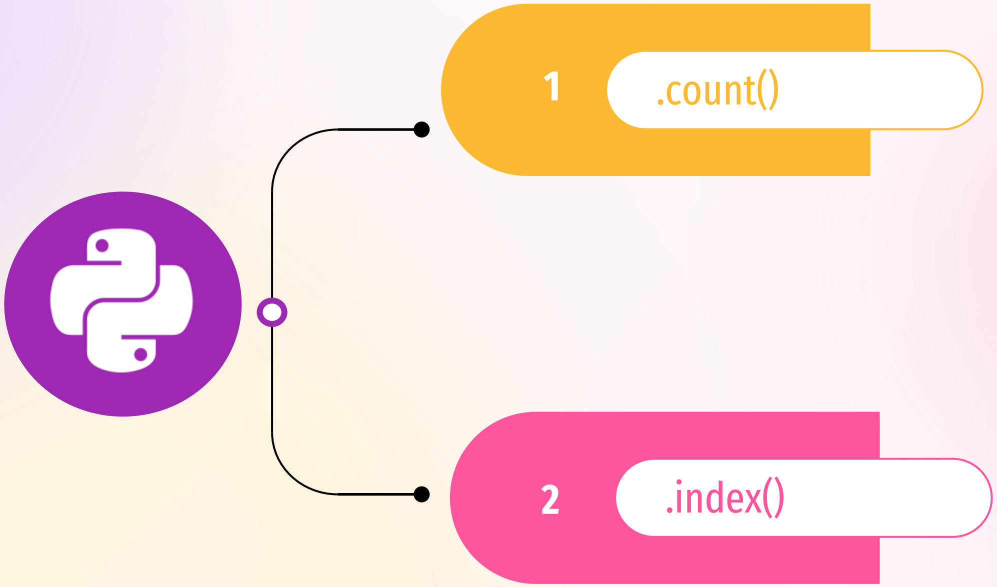
.insert( )

Add new item at specified index

.copy()

Copy all item of list into new variabel

Tuple Method



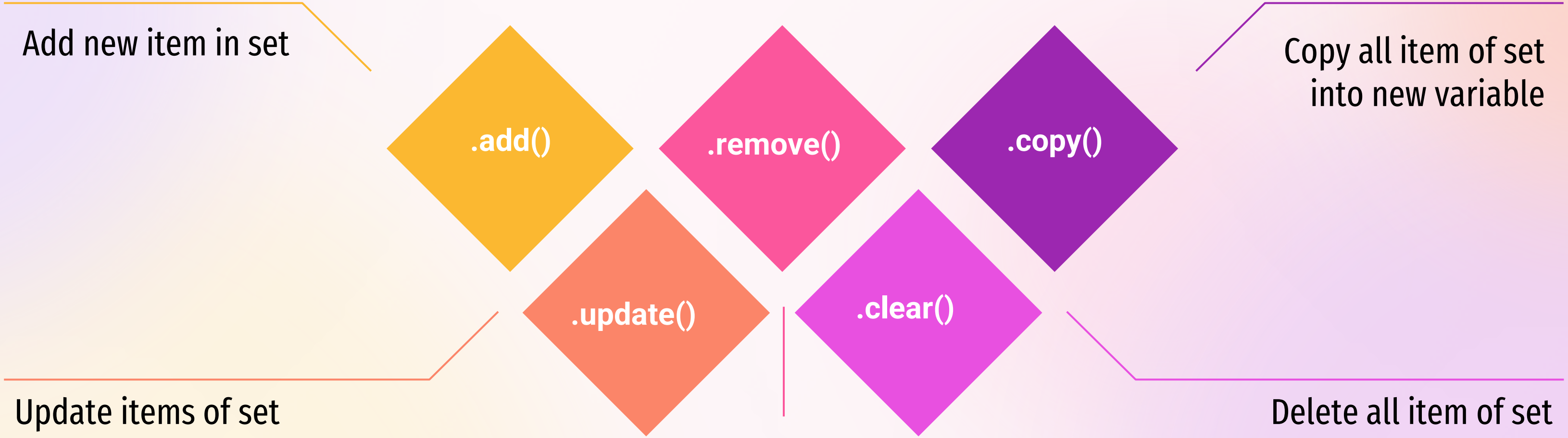
1 .count()

Count the number of occurrence of specified item

2 .index()

Search index of specified item

Set Method



Add new item in set

Copy all item of set
into new variable

Update items of set

Delete all item of set

Delete specified item
of set

Dictionary Method

Notation []

1. Update value of an existing key-value pair
2. Add new key-value pair

del[] / .pop()

Remove key value pair

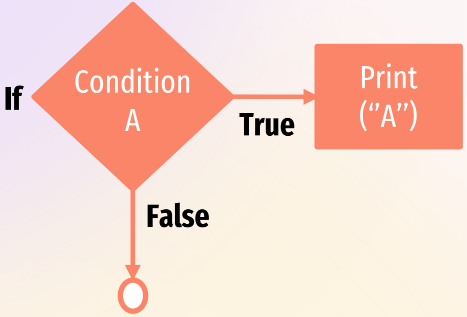
.get()

Access non existing key

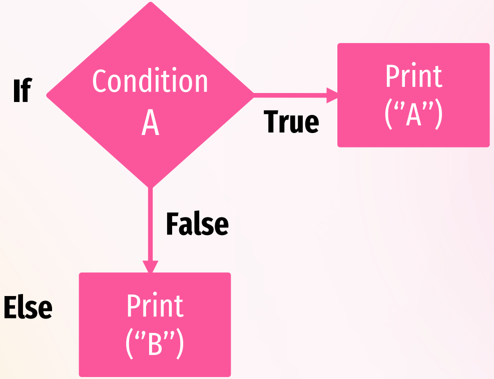
You can check **Python - Dictionary Methods (w3schools.com)** for another Dictionary Method

If Statement Concept

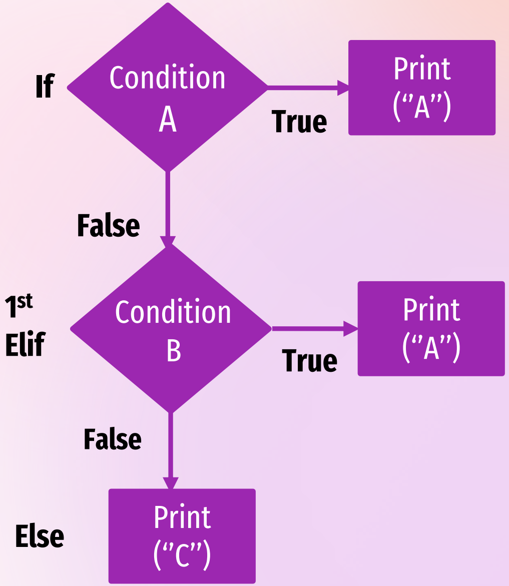
1 If



2 If Else



1 If Elif



Example of Indexing and Slicing

a = “Hay, everyone!”

-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	a	y	,		e	v	e	r	y	o	n	e	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

1. Positive Index

In [ ] = a[0]

Out[ ] ='H'

2. Negative Index

In [ ] = a[-1]

Out[ ] = '!'

3. Slicing with Starting and Ending Index

#Positive Index

In [ ] = a[0:3]

Out[ ] = 'Hay'

#Negative Index

In [ ] = a[-14:-11]

Out[ ] = 'Hay'

4. Slicing with Ending Index

#Positive Index

In [ ] = a[:3]

Out[ ] = 'Hay'

#Negative Index

In [ ] = a[:-11]

Out[ ] = 'Hay'

Example of Indexing and Slicing

a = “Hay, everyone!”

-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	a	y	,		e	v	e	r	y	o	n	e	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

5. Slicing with Starting Index

```
#Positive Index
 In [ ]   = a[5:]
 Out[ ] = ‘everyone!’

#Negative Index
 In [ ]   = a[-9:]
 Out[ ] = ‘everyone!’
```

6. Slicing with Step

```
#Positive Index
        In [ ]   = a[::2]
        Out[ ] = ‘Hy vroe’

 #Negative Index
        In [ ]   = a[::-2]
        Out[ ] = ‘!nyee,a’
```

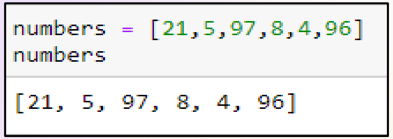
7. Slicing with Step, Starting and Ending Index

```
#Positive Index
 In [ ]   = a[0:13:2]
 Out[ ] = ‘Hy vroe’

#Negative Index
 In [ ]   = a[-1:-14:-2]
 Out[ ] = ‘’!nyee,a’
```

Unpacking List

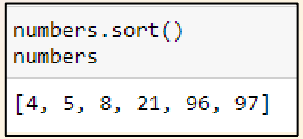
Variable Known :



```
numbers = [21,5,97,8,4,96]
numbers
```

```
[21, 5, 97, 8, 4, 96]
```

The sequence of element list will not change after running, thus the list is ordered

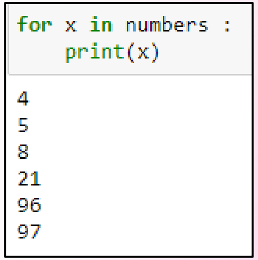


```
numbers.sort()
numbers
```

```
[4, 5, 8, 21, 96, 97]
```

When we use method “sort”, the sequence of element list has changed

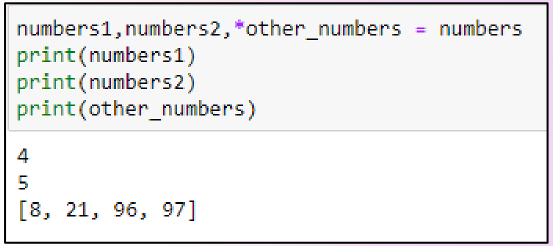
1. Unpacking List without Index



```
for x in numbers :
    print(x)
```

```
4
5
8
21
96
97
```

You can unpack all element of list



```
numbers1,numbers2,*other_numbers = numbers
print(numbers1)
print(numbers2)
print(other_numbers)
```

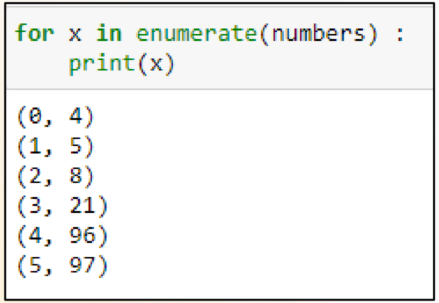
```
4
5
[8, 21, 96, 97]
```

Or unpack only some element of list

Unpacking List

2. Unpacking List with Index

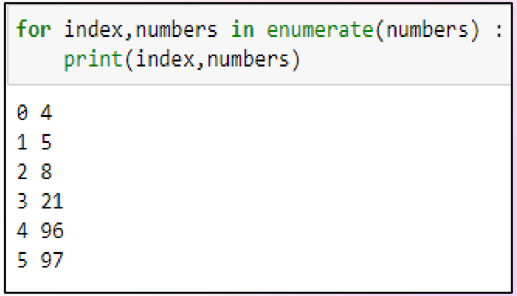
Use method “enumerate”



```
for x in enumerate(numbers) :
    print(x)
```

```
(0, 4)
(1, 5)
(2, 8)
(3, 21)
(4, 96)
(5, 97)
```

Result is tuple



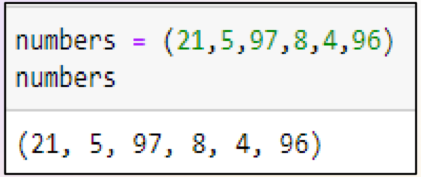
```
for index,numbers in enumerate(numbers) :
    print(index,numbers)
```

```
0 4
1 5
2 8
3 21
4 96
5 97
```

Result is not tuple

Unpacking Tuple

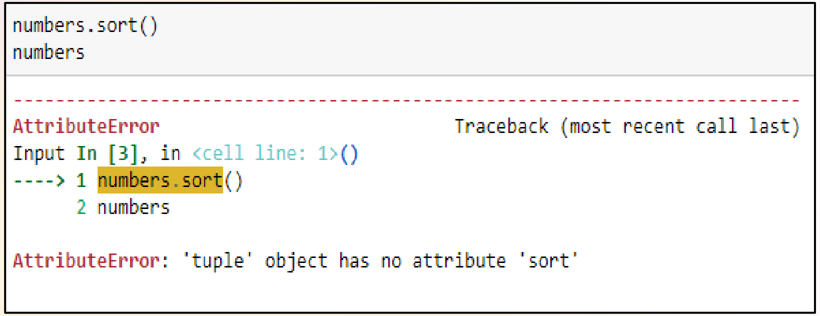
Variable Known :



```
numbers = (21,5,97,8,4,96)
numbers
```

```
(21, 5, 97, 8, 4, 96)
```

The sequence of element tuple will not change after running, thus the tuple is ordered



```
numbers.sort()
numbers
```

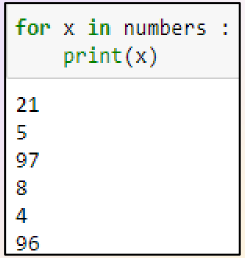
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [3], in <cell line: 1>()
----> 1 numbers.sort()
      2 numbers

AttributeError: 'tuple' object has no attribute 'sort'
```

When we use method “sort”, the sequence of element tuple has not change. It means tuple is mutable

Unpacking Tuple

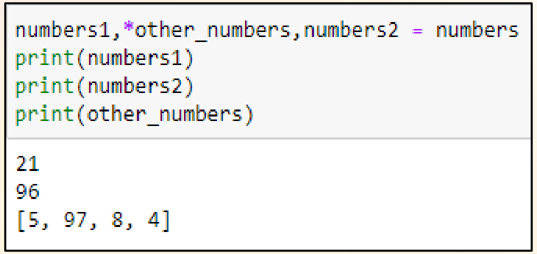
1. Unpacking Tuple without Index



```
for x in numbers :
    print(x)
```

```
21
5
97
8
4
96
```

You can unpack all element of tuple

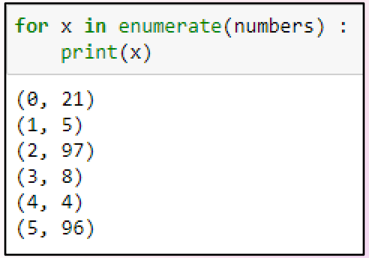


```
numbers1,*other_numbers,numbers2 = numbers
print(numbers1)
print(numbers2)
print(other_numbers)
```

```
21
96
[5, 97, 8, 4]
```

Or unpack only some element of tuple

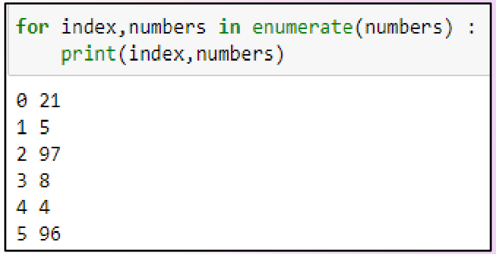
2. Unpacking Tuple with Index



```
for x in enumerate(numbers) :
    print(x)
```

```
(0, 21)
(1, 5)
(2, 97)
(3, 8)
(4, 4)
(5, 96)
```

Result is tuple



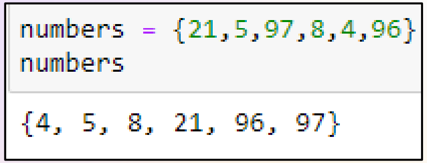
```
for index,numbers in enumerate(numbers) :
    print(index,numbers)
```

```
0 21
1 5
2 97
3 8
4 4
5 96
```

Result is not tuple

Unpacking Set

Variable Known :

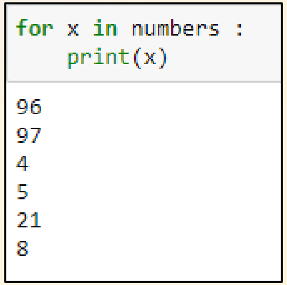


```
numbers = {21,5,97,8,4,96}
numbers
```

```
{4, 5, 8, 21, 96, 97}
```

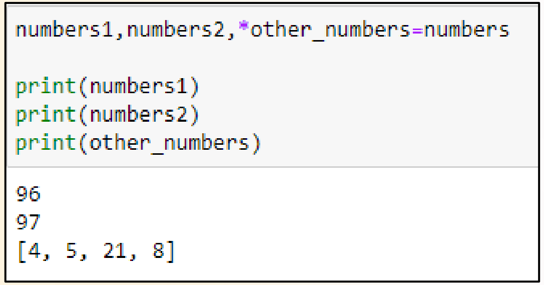
The sequence of element set will change after running,
thus the list is unordered

Set does not support indexing. So when we unpacking set, the result is random order



```
for x in numbers :
    print(x)
```

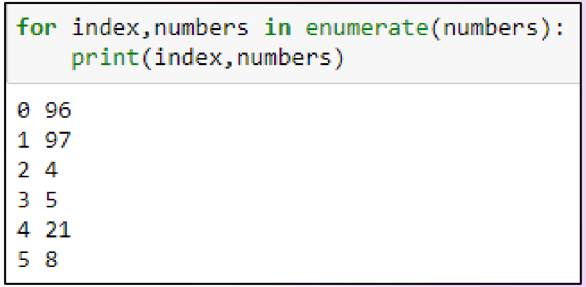
```
96
97
4
5
21
8
```



```
numbers1,numbers2,*other_numbers=numbers

print(numbers1)
print(numbers2)
print(other_numbers)
```

```
96
97
[4, 5, 21, 8]
```

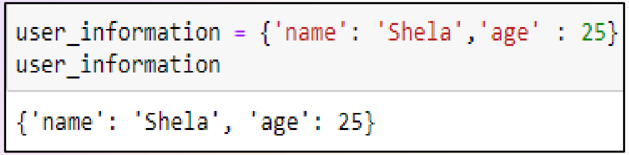


```
for index,numbers in enumerate(numbers):
    print(index,numbers)
```

```
0 96
1 97
2 4
3 5
4 21
5 8
```

Unpacking Dictionary

Variable Known :

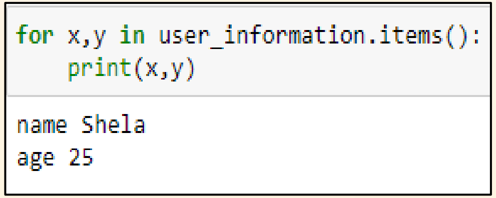


```
user_information = {'name': 'Shela','age' : 25}
user_information
```

```
{'name': 'Shela', 'age': 25}
```

Dictionary consists of key (name, age) and value (Shela, 25)

1. Unpacking with “items” Method

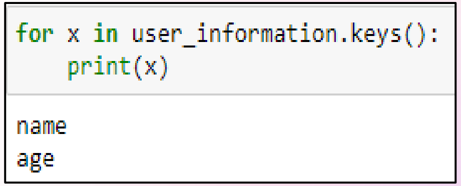


```
for x,y in user_information.items():
    print(x,y)
```

```
name Shela
age 25
```

Result is key and value

2. Unpacking with “keys” Method

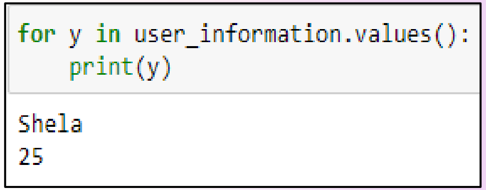


```
for x in user_information.keys():
    print(x)
```

```
name
age
```

Result is key

3. Unpacking with “values” Method



```
for y in user_information.values():
    print(y)
```

```
Shela
25
```

Result is values

THANK YOU!