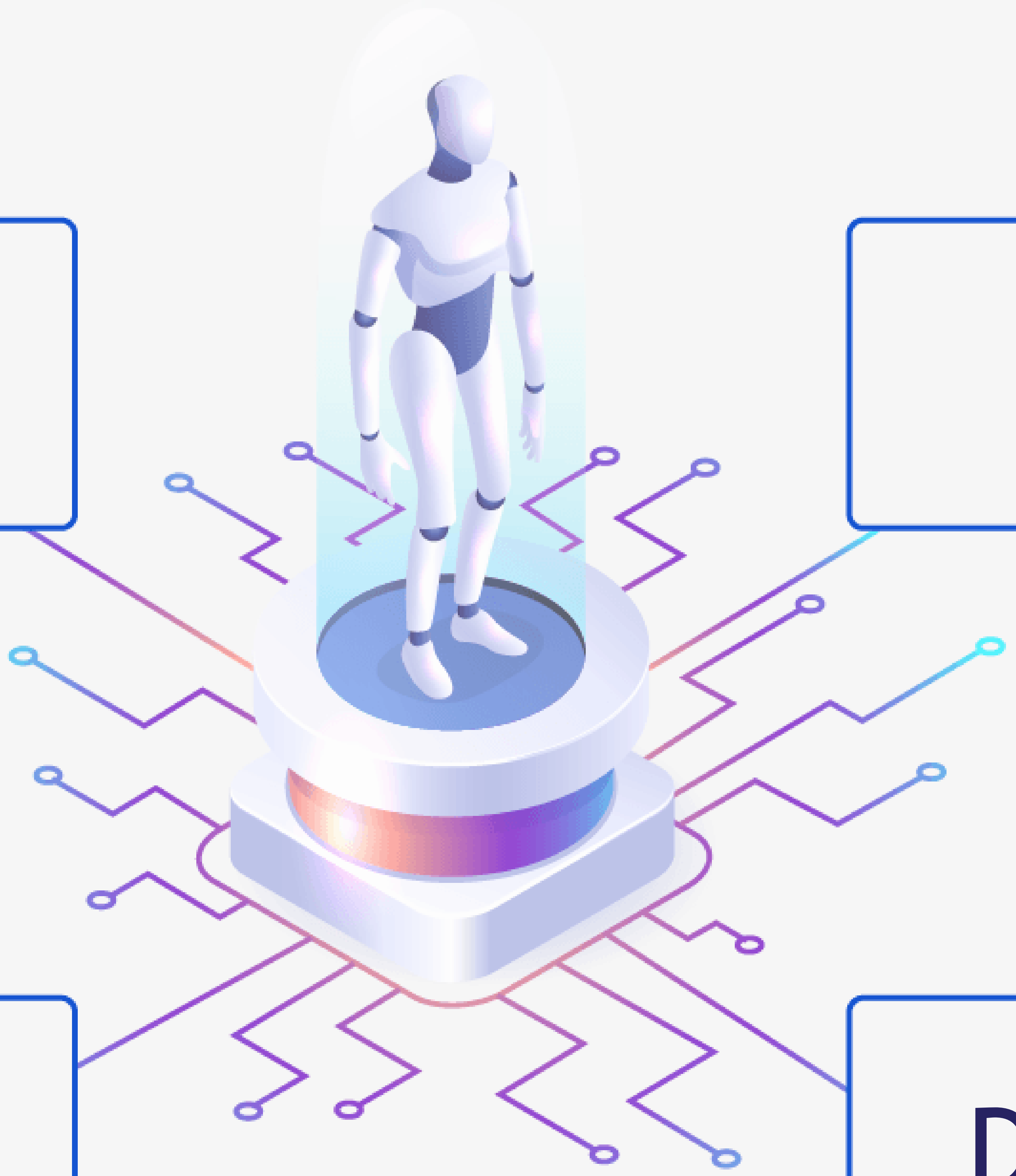
The Learning Hub

# Data Structure

Education and Training Solutions 2023

The Learning Hub



List

Tuple

Set

Dictionary

The Learning Hub

## Lists

# Introduction to Lists

- **Lists** are Python's most flexible ordered collection object type. Unlike strings, lists can contain any sort of object: numbers, strings, and even other lists.

- **Lists** are created using square brackets

  List =[item1 , item2, item3,...]

The Learning Hub

## Introduction to Lists

- **Lists:** are used to store multiple items in a single variable **(mutable).**

```python
mylist = ["ai", "hello", "be right back"]
```

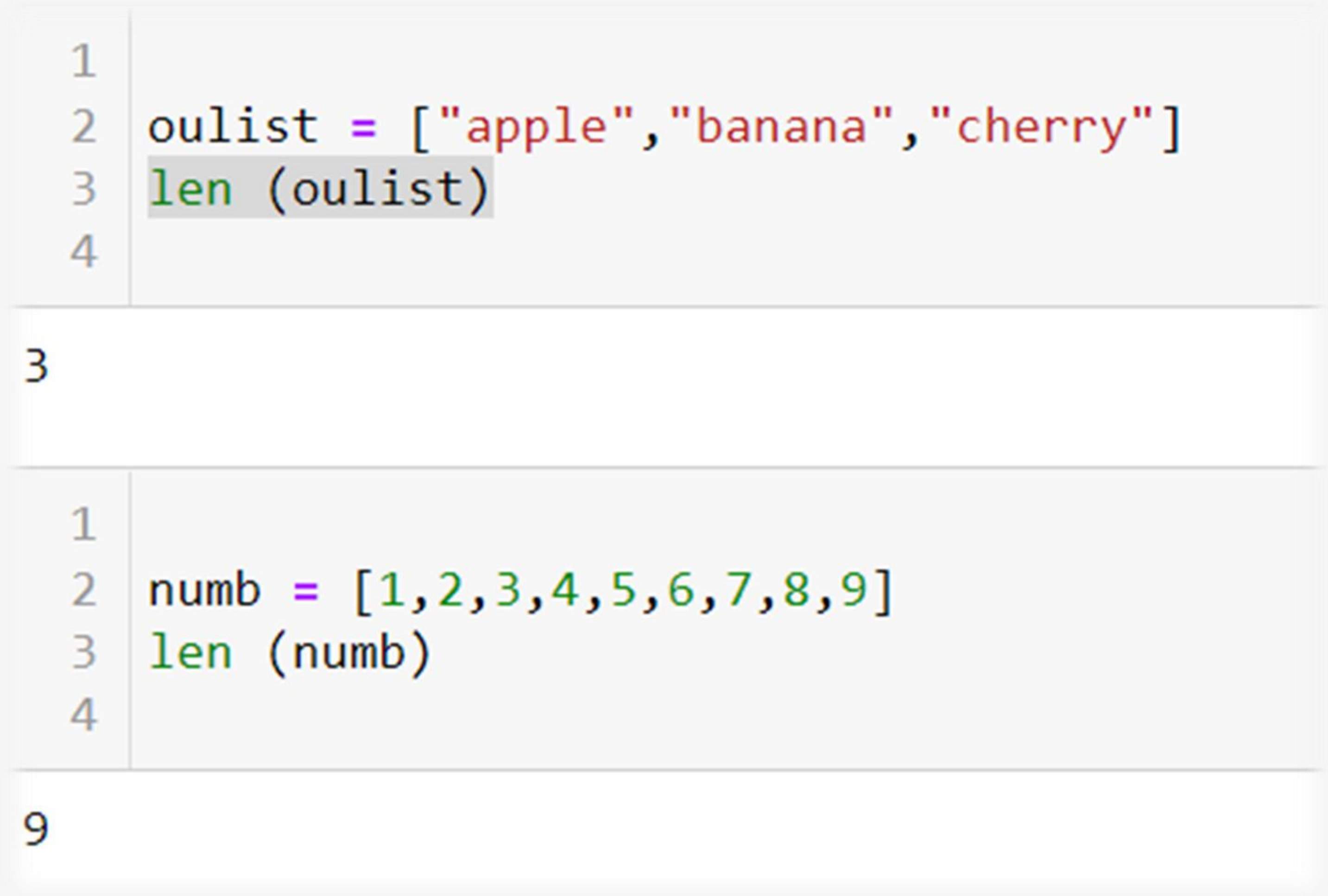
- **List Items:** ordered, changeable, and allow duplicate values.

```python
lis = ["program", "hi", "good", "nice", "good"]
```

The Learning Hub

# Introduction to Lists

- **List Length:** To determine how many items a list has, use the len() function.



```

oulist = ["apple","banana","cherry"]
len (oulist)

```

```
3
```

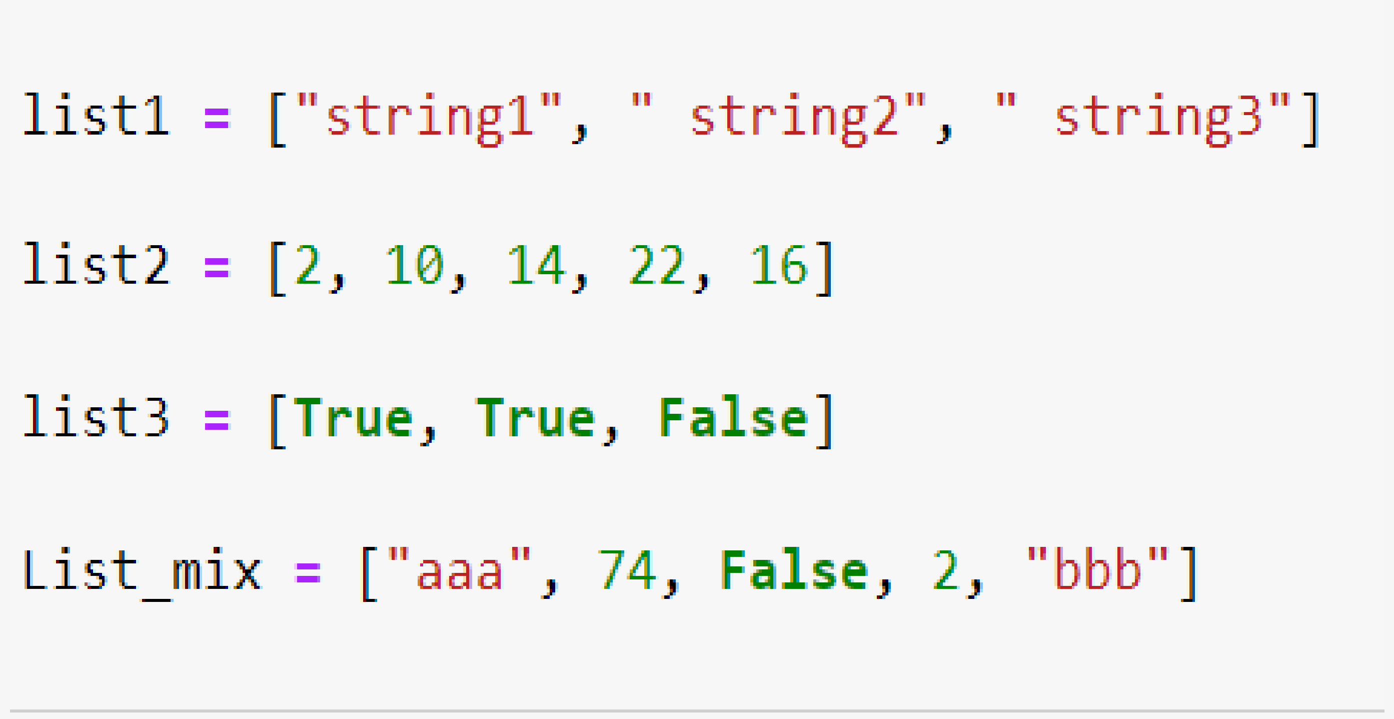
```

numb = [1,2,3,4,5,6,7,8,9]
len (numb)

```

```
9
```

The Learning Hub

## Introduction to Lists

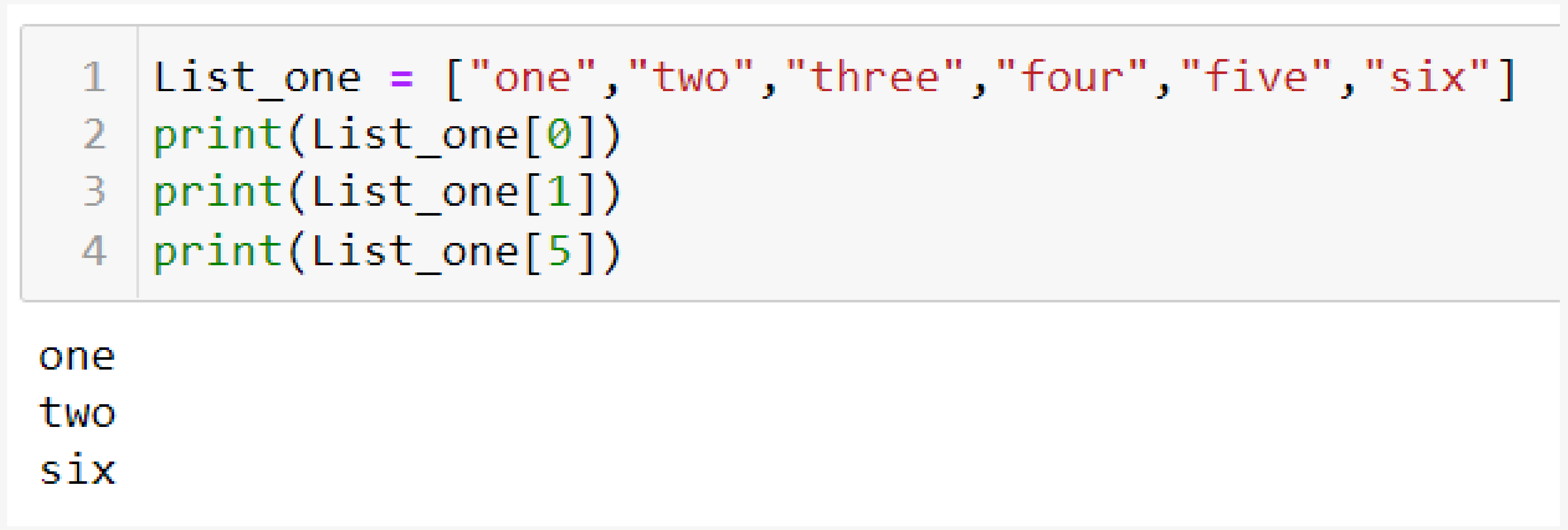
- **List Data Types:** items can be of any data type, and a list can contain different data types



```
list1 = ["string1", " string2", " string3"]

list2 = [2, 10, 14, 22, 16]

list3 = [True, True, False]

List_mix = ["aaa", 74, False, 2, "bbb"]
```

# Access List items

- **Index number:** List items are indexed, the first item in the list has the index [0], the second item has index [1]
- List items can be accessed by referring to the index number

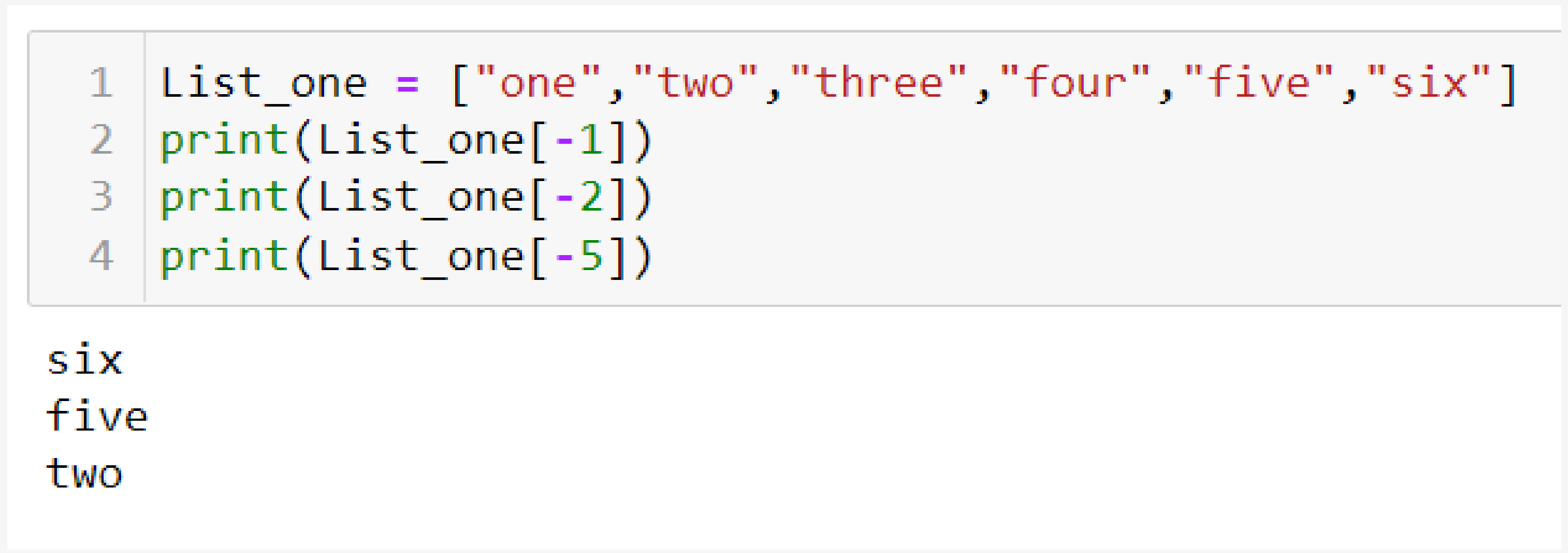


```
List_one = ["one","two","three","four","five","six"]
print(List_one[0])
print(List_one[1])
print(List_one[5])
```

```
one
two
six
```

## Access List items

- **Negative Indexing:** means start from the end

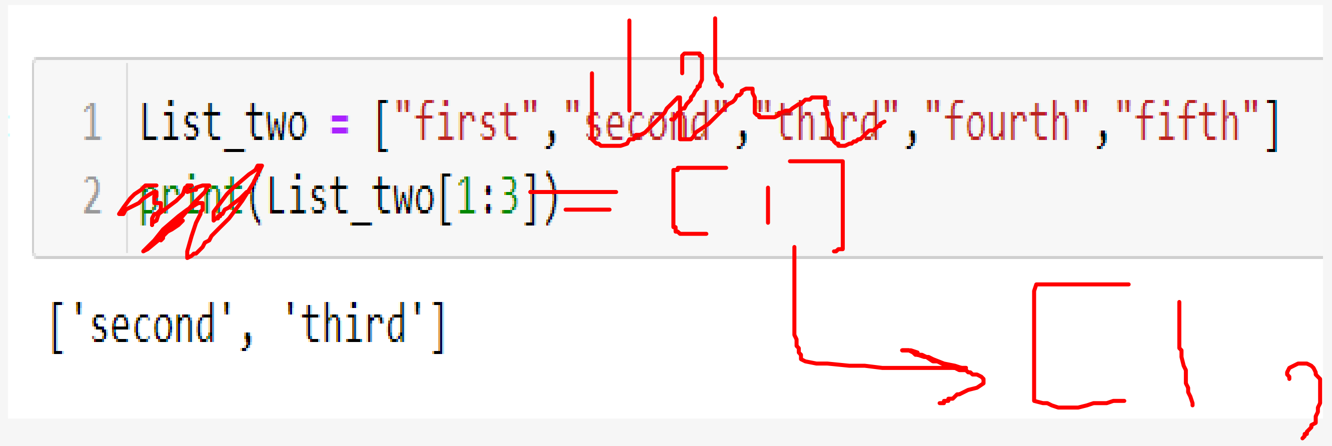


```
List_one = ["one","two","three","four","five","six"]
print(List_one[-1])
print(List_one[-2])
print(List_one[-5])
```

```
six
five
two
```

# Access List items

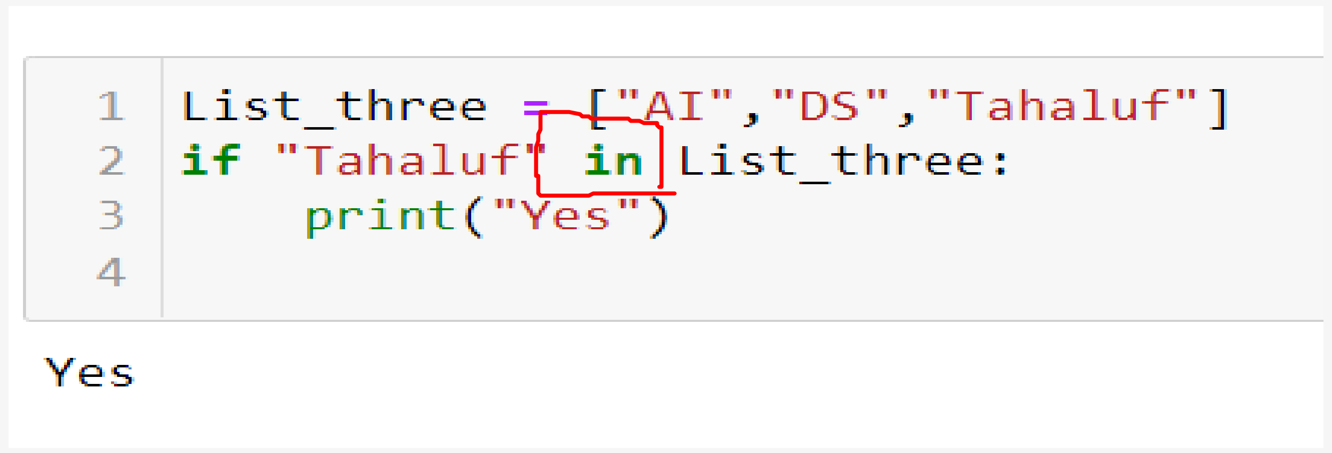
- **Range of Indexes:** You can specify a range of indexes by specifying the start and end points.



```
List_two = ["first","second","third","fourth","fifth"]
print(List_two[1:3])
```

```
['second', 'third']
```

- **Check if an Item Exists or not in the list**

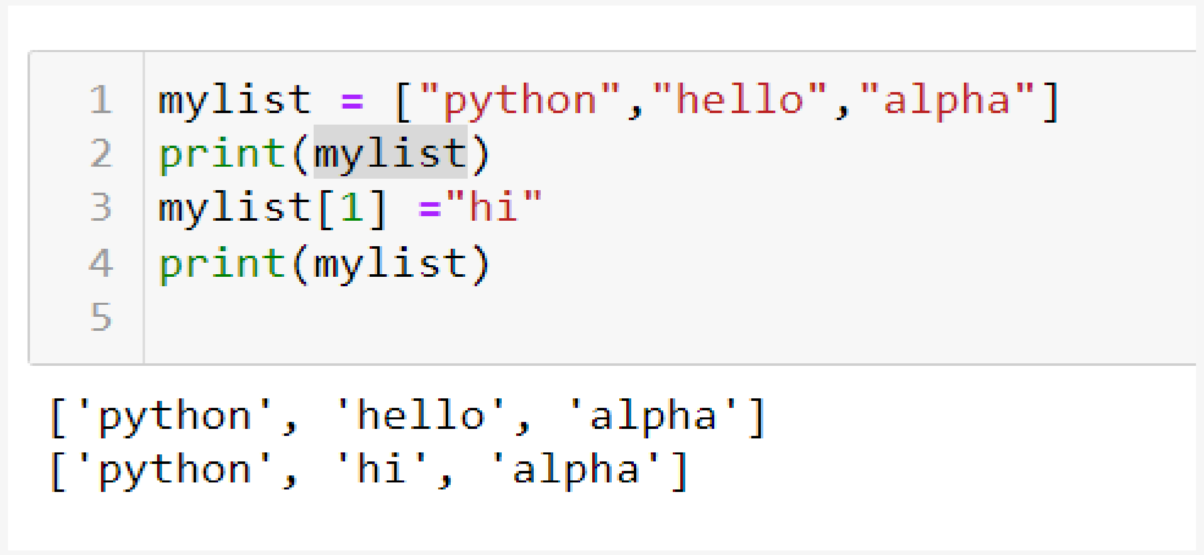


```
List_three = ["AI","DS","Tahaluf"]
if "Tahaluf" in List_three:
    print("Yes")
```

```
Yes
```

# Change List items

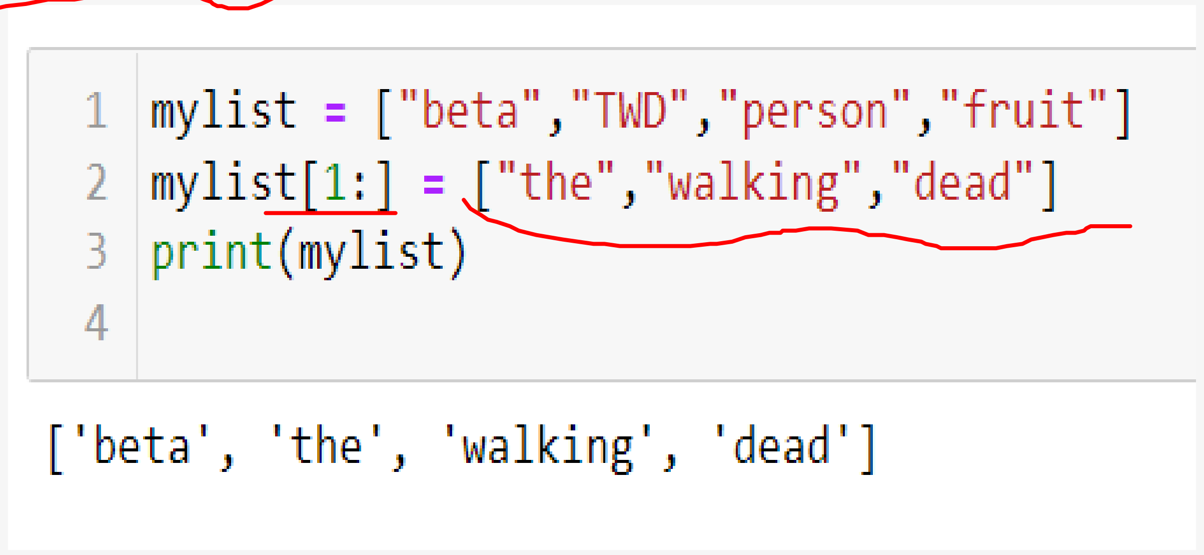
- **Change Item Value**



```python
mylist = ["python","hello","alpha"]
print(mylist)
mylist[1] ="hi"
print(mylist)

```

```
['python', 'hello', 'alpha']
['python', 'hi', 'alpha']
```

- **Change a Range of Item Values**



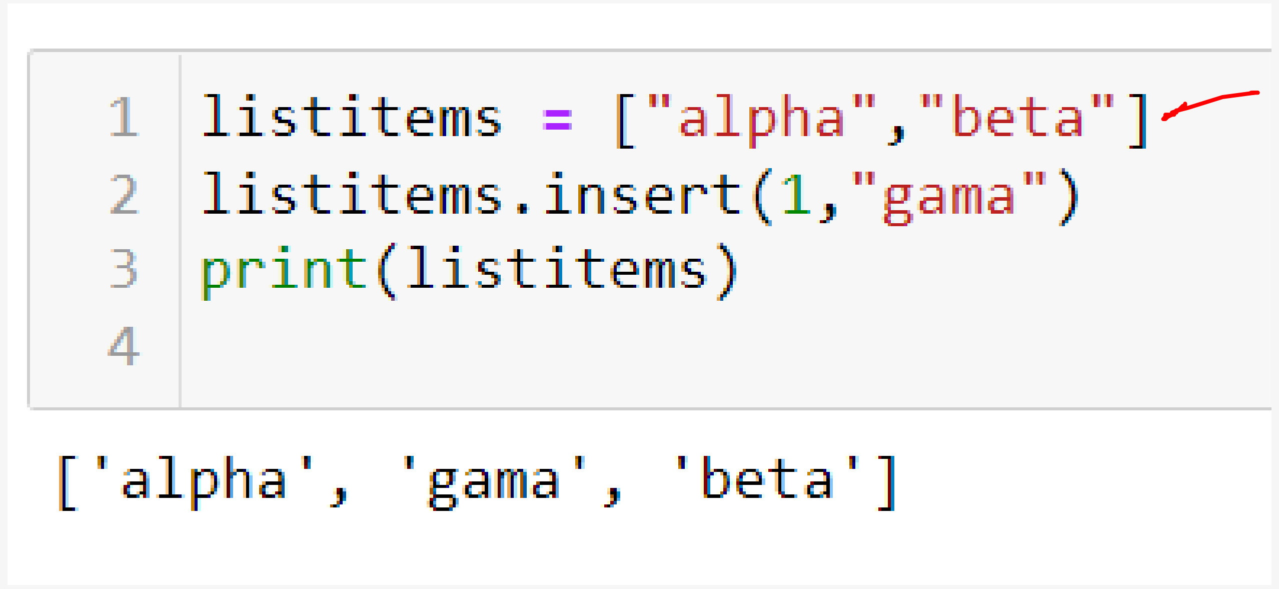
```python
mylist = ["beta","TWD","person","fruit"]
mylist[1:] = ["the","walking","dead"]
print(mylist)

```

```
['beta', 'the', 'walking', 'dead']
```

The Learning Hub

## Change List items

- **Insert Items**



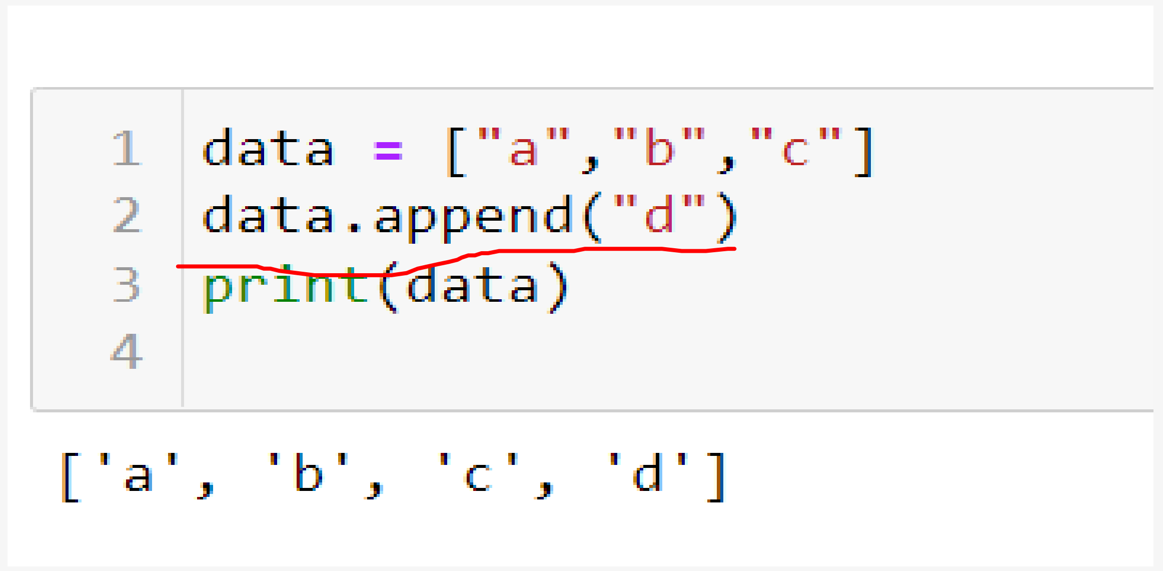
```
listitems = ["alpha","beta"]
listitems.insert(1,"gama")
print(listitems)

```

```
['alpha', 'gama', 'beta']
```

The Learning Hub

## Add List items

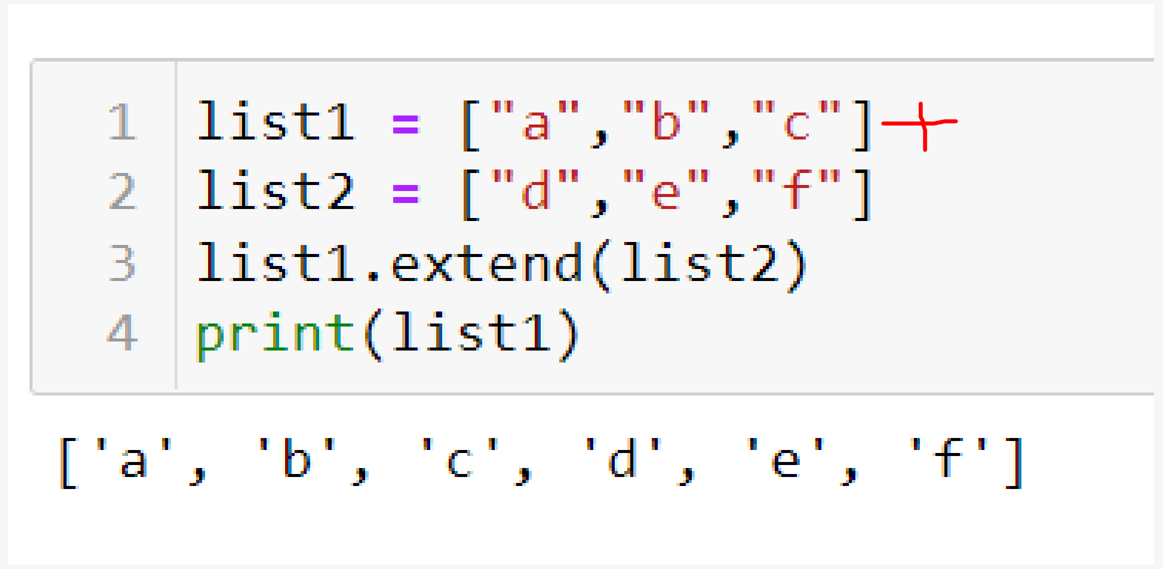
- Append Items



```
data = ["a","b","c"]
data.append("d")
print(data)

```

```
['a', 'b', 'c', 'd']
```

- Extend List



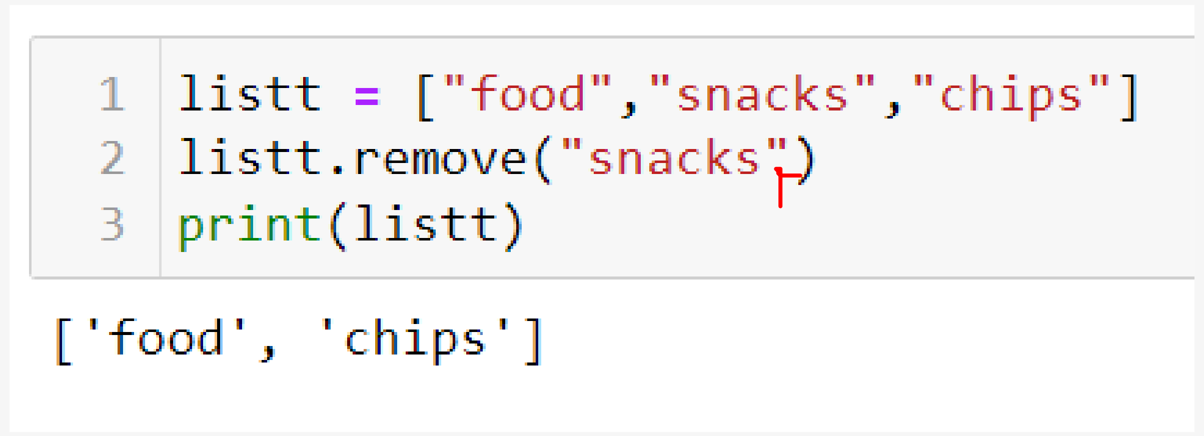
```
list1 = ["a","b","c"]
list2 = ["d","e","f"]
list1.extend(list2)
print(list1)
```

```
['a', 'b', 'c', 'd', 'e', 'f']
```

The Learning Hub

## Remove List items

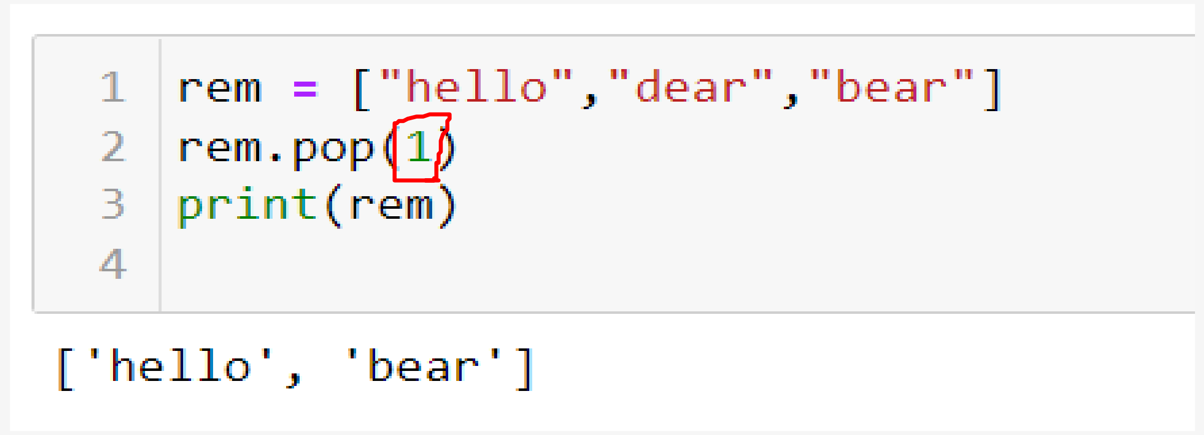
- **Remove Specified Item**



```
listt = ["food","snacks","chips"]
listt.remove("snacks")
print(listt)
```

```
['food', 'chips']
```

- **Remove Specified Index using pop**

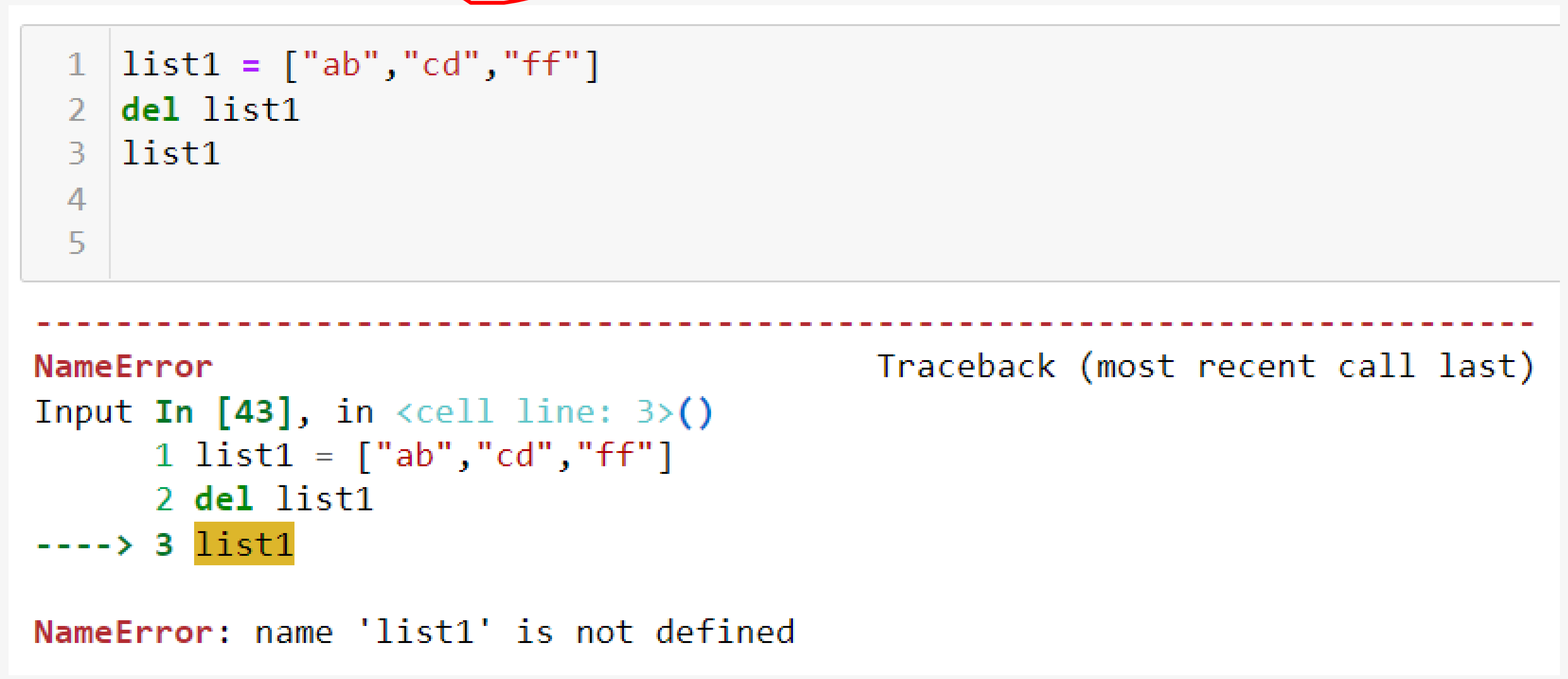


```
rem = ["hello","dear","bear"]
rem.pop(1)
print(rem)

```

```
['hello', 'bear']
```

# Remove List items

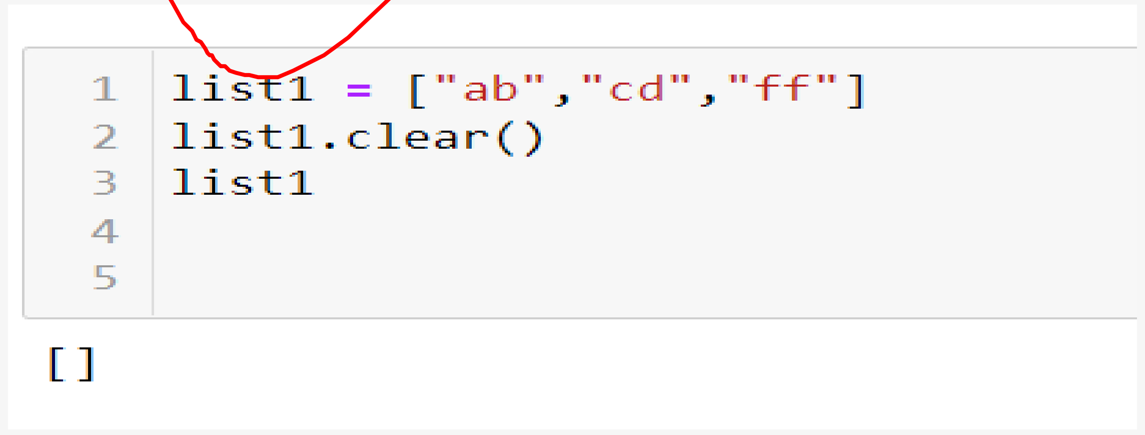
- **Remove Specified Index using del**



```
list1 = ["ab","cd","ff"]
del list1
list1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [43], in <cell line: 3>()
      1 list1 = ["ab","cd","ff"]
      2 del list1
----> 3 list1

NameError: name 'list1' is not defined
```

- **Remove Specified Index using clear**



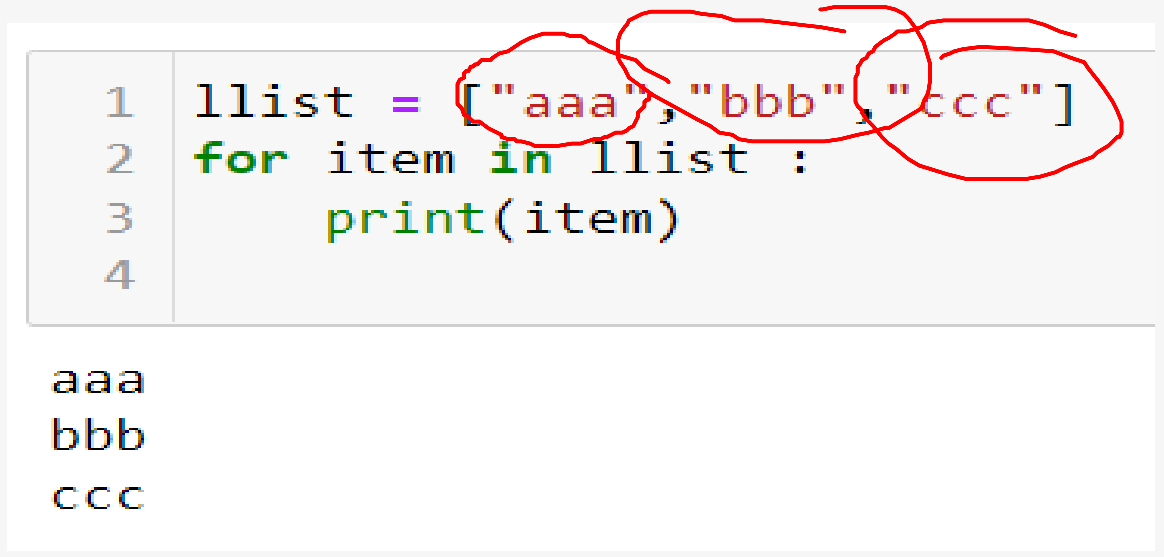
```
list1 = ["ab","cd","ff"]
list1.clear()
list1
```

```
[]
```

The Learning Hub

# Loop inside Lists

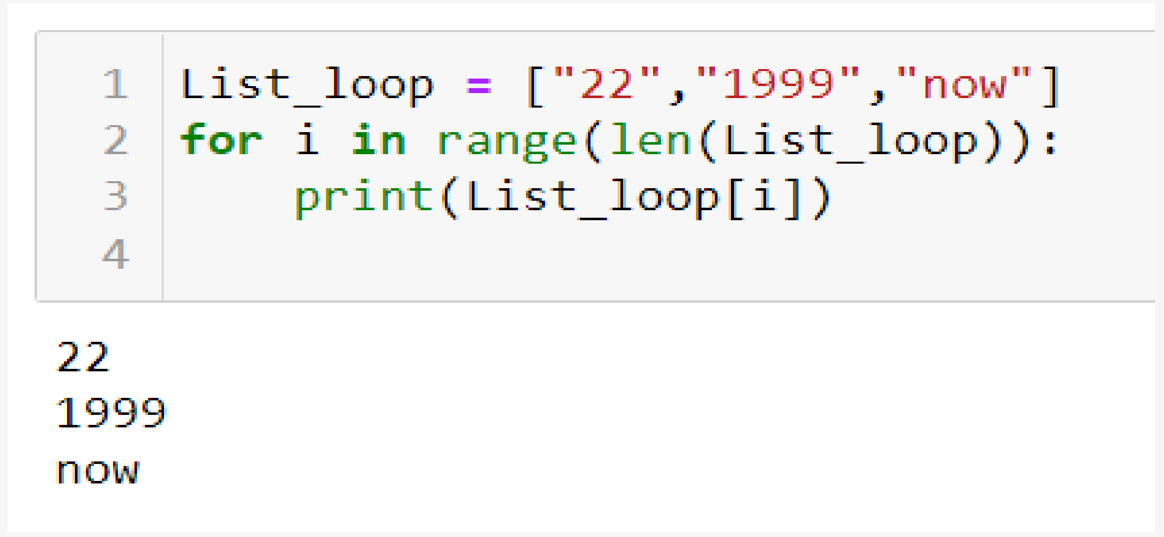
- **Loop in the list**



```
llist = ["aaa","bbb","ccc"]
for item in llist :
    print(item)
```

```
aaa
bbb
ccc
```

- **Loop Through the Index Numbers**



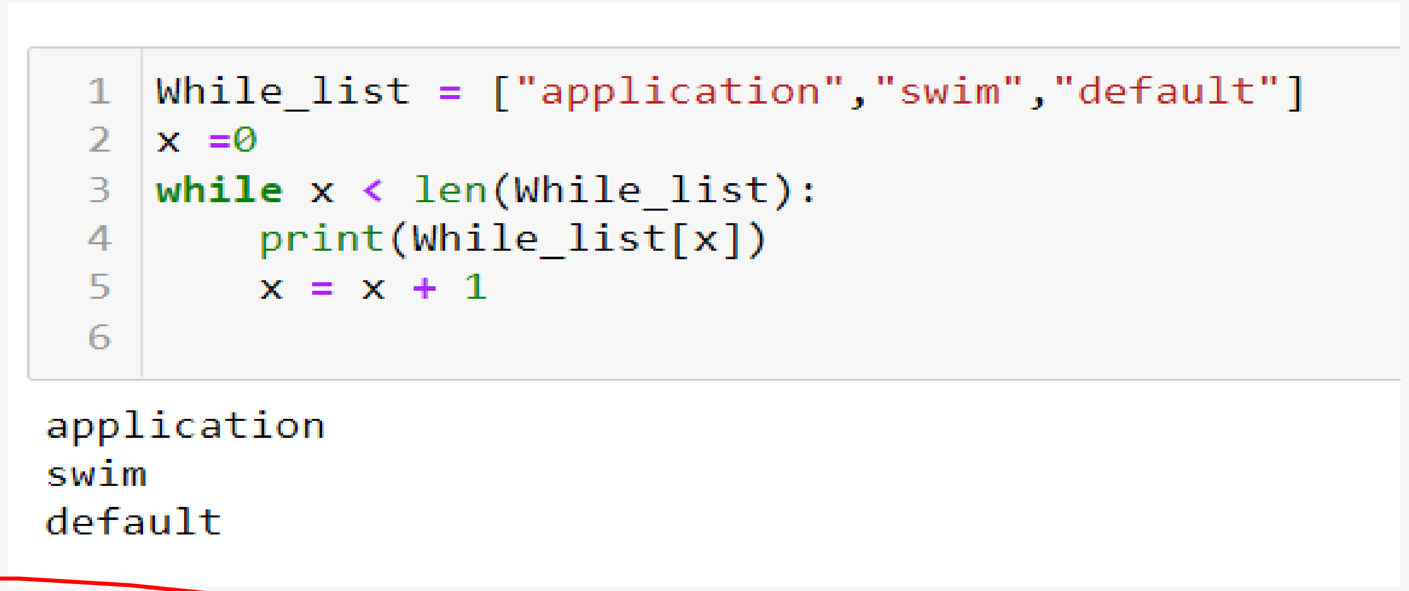
```
List_loop = ["22","1999","now"]
for i in range(len(List_loop)):
    print(List_loop[i])
```

```
22
1999
now
```

The Learning Hub

# Loop inside Lists

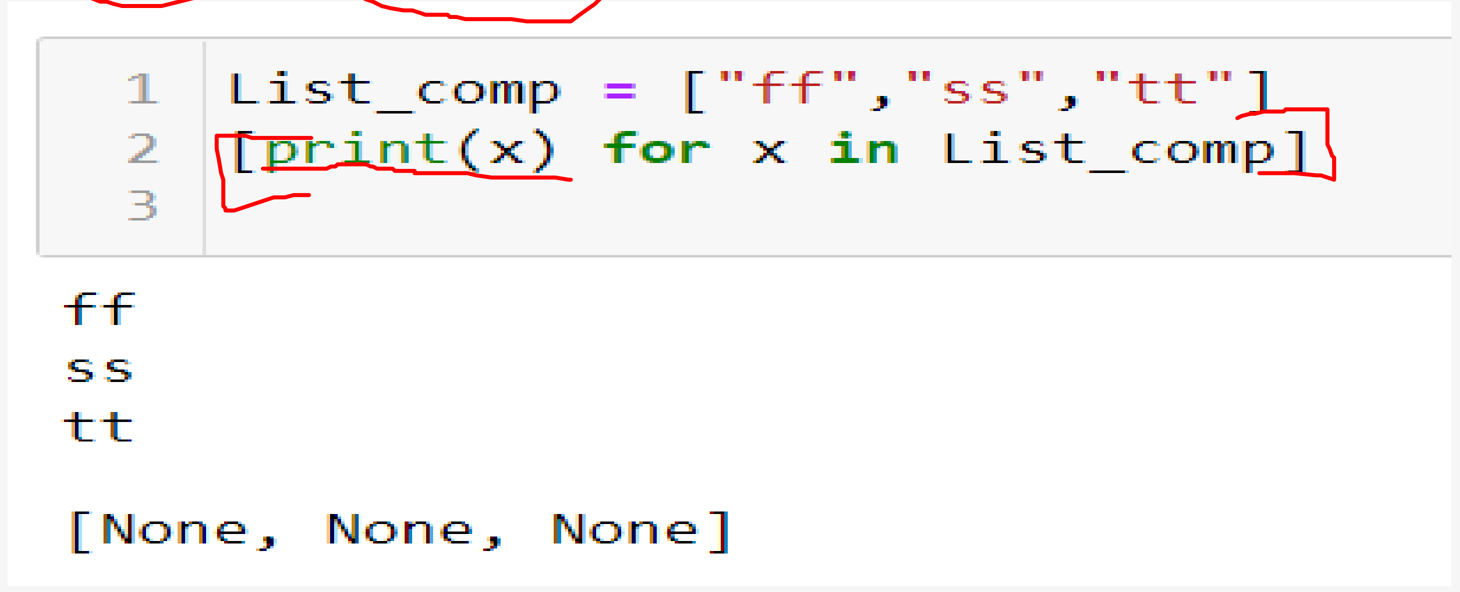
- Using a While Loop



```
While_list = ["application","swim","default"]
x =0
while x < len(While_list):
    print(While_list[x])
    x = x + 1

```

```
application
swim
default
```

- Looping Using List Comprehension



```
List_comp = ["ff","ss","tt"]
[print(x) for x in List_comp]

```

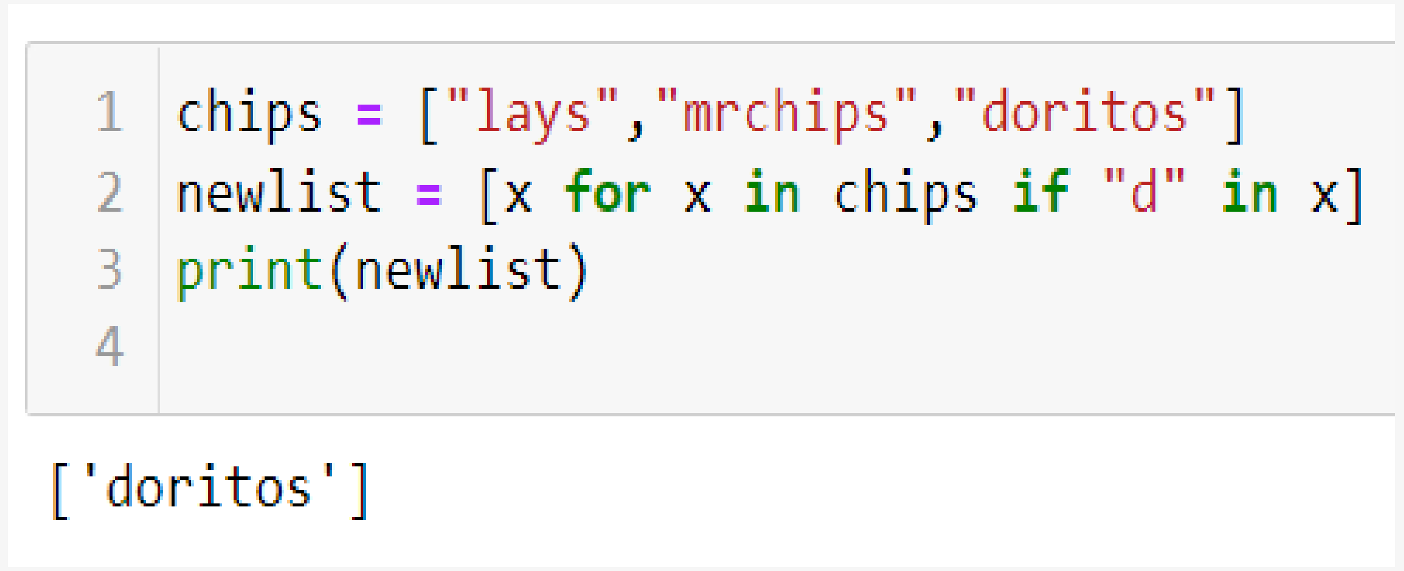
```
ff
ss
tt

[None, None, None]
```

# List Comprehension

- **List comprehension**: allows the creation of a new list based on the values of an existing list using shorter syntax.

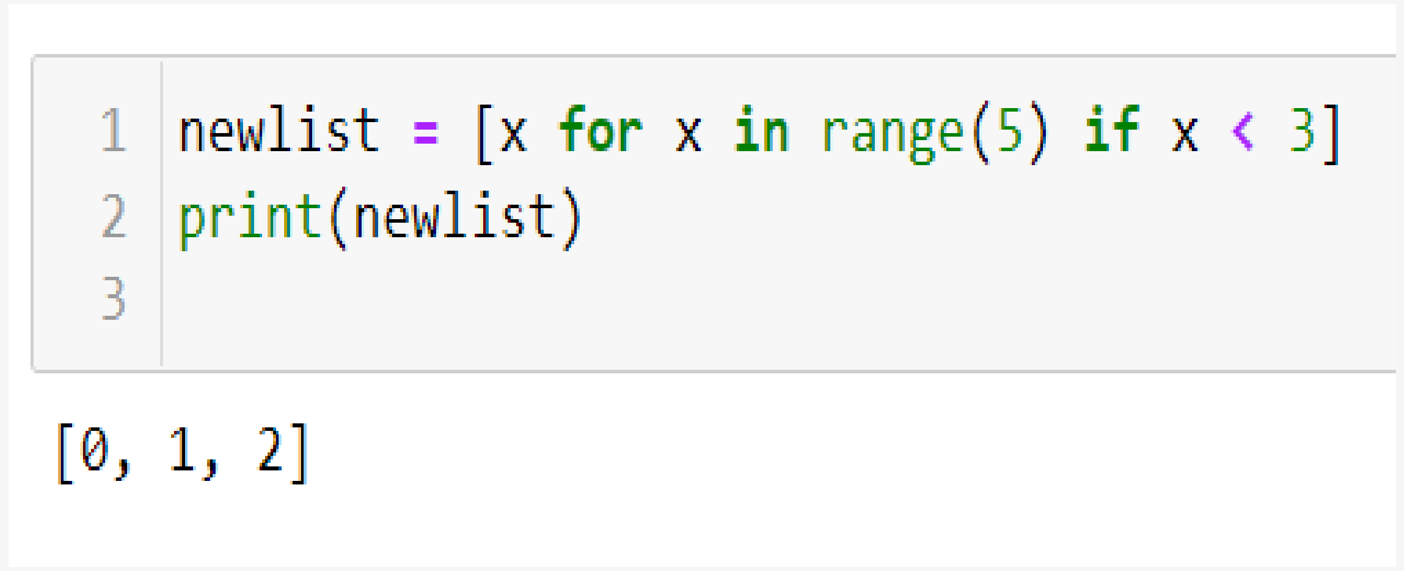
- **Syntax of list comprehension:**

List_syn : [expression for item in iterable if condition == True]



```
chips = ["lays","mrchips","doritos"]
newlist = [x for x in chips if "d" in x]
print(newlist)

```

```
['doritos']
```



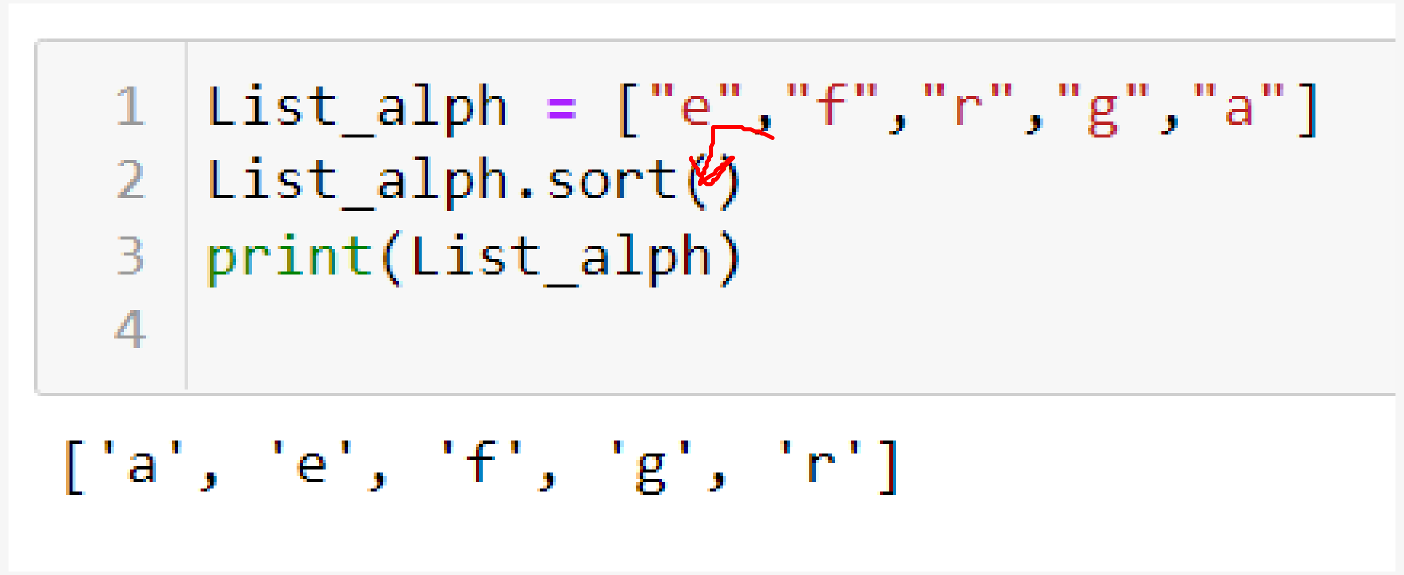
```
newlist = [x for x in range(5) if x < 3]
print(newlist)

```

```
[0, 1, 2]
```

The Learning Hub

# Sort Lists

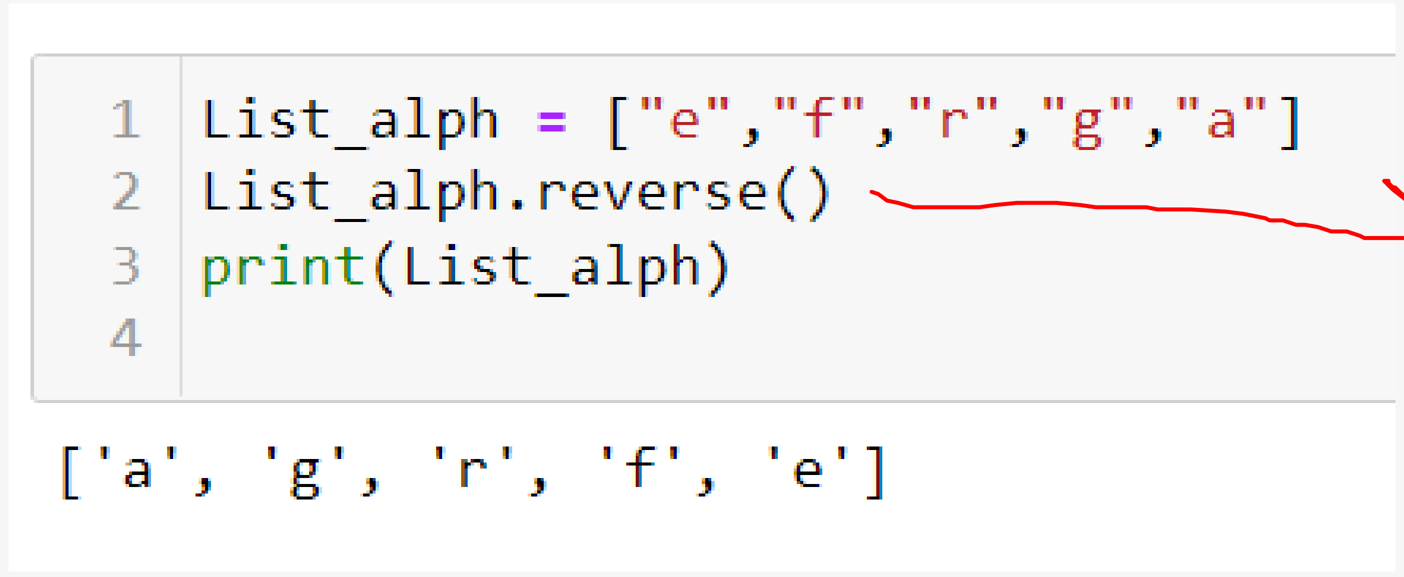
- Alphabetically



```python
List_alph = ["e","f","r","g","a"]
List_alph.sort()
print(List_alph)

```

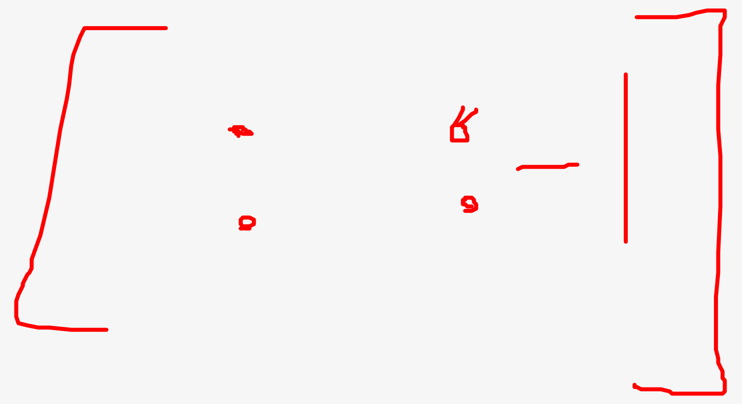
```
['a', 'e', 'f', 'g', 'r']
```

- Reverse



```python
List_alph = ["e","f","r","g","a"]
List_alph.reverse()
print(List_alph)

```

```
['a', 'g', 'r', 'f', 'e']
```



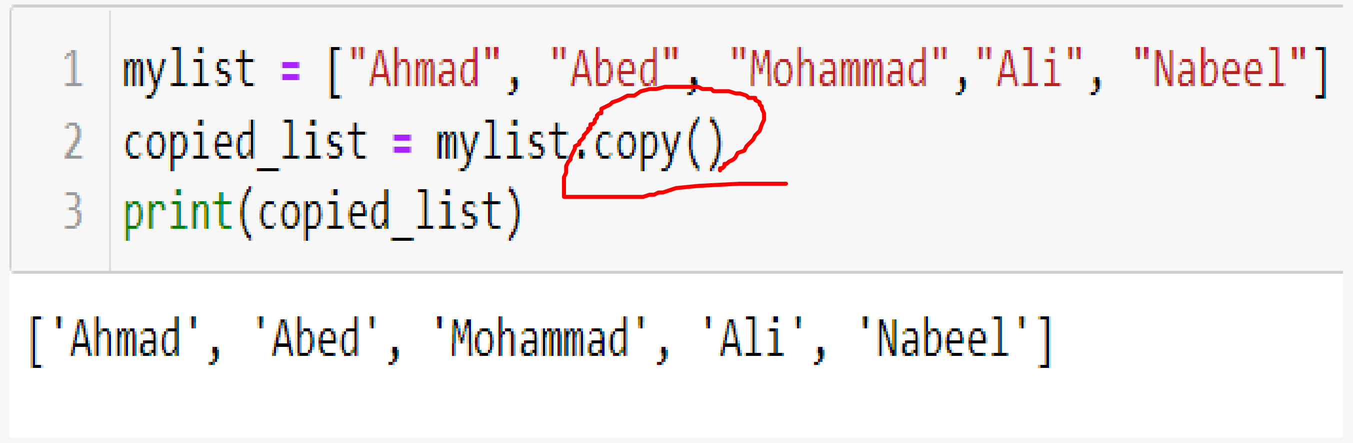
# Copy Lists

x = 5

y = x

y x = 2

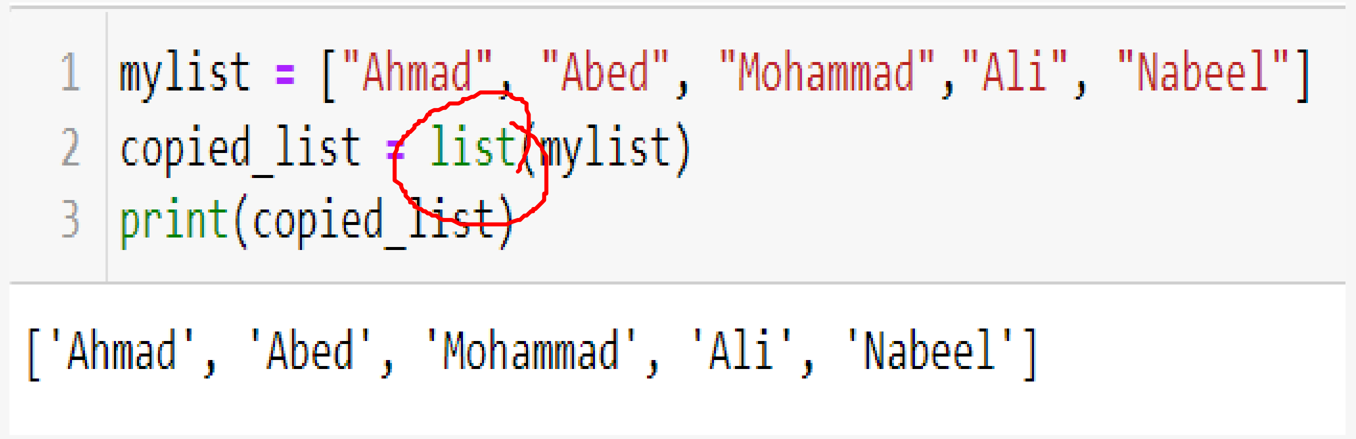
- There are many ways to make a copy of the list, one way is to use the built-in List method copy()



```
mylist = ["Ahmad", "Abed", "Mohammad","Ali", "Nabeel"]
copied_list = mylist.copy()
print(copied_list)
```

```
['Ahmad', 'Abed', 'Mohammad', 'Ali', 'Nabeel']
```

- Making a copy of a list using the built_in list() method



```
mylist = ["Ahmad", "Abed", "Mohammad","Ali", "Nabeel"]
copied_list = list(mylist)
print(copied_list)
```

```
['Ahmad', 'Abed', 'Mohammad', 'Ali', 'Nabeel']
```

# List Methods

| Method | Description |
|---|---|
| append() | Adds an element at the end of the list |
| clear() | Removes all the elements from the list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| extend() | Add the elements of a list, to the end of the current list |
| index() | Returns the index of the first element with the specified value |
| insert() | Adds an element at the specified position |
| pop() | Removes the element at the specified position |
| remove() | Removes the item with the specified value |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

The Learning Hub

## Tuples

The Learning Hub

# Introduction to Tuples

- **Tuples:** A tuple is a collection which is ordered and unchangeable **(Immutable).**

```python
tuplee = ("alter", "baloon", "at")
```

- **Allow Duplicates**

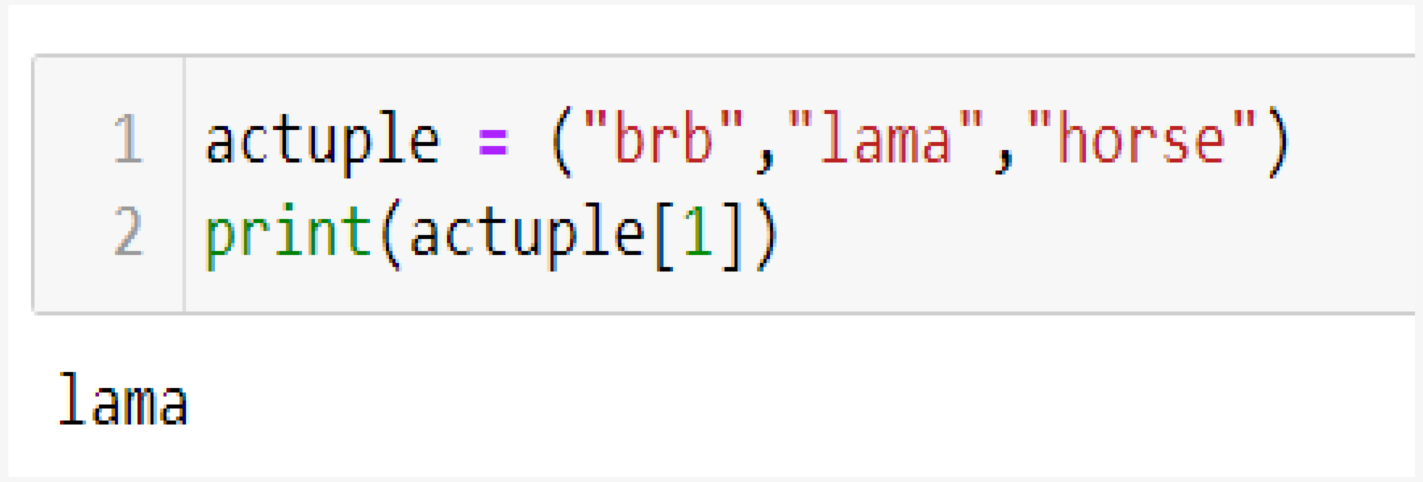
```python
duptuple = ("head", "veg", "head", "apple", "moon")
```

- **Tuple Data Types**

```python
Tuple_d = ("str", 9, False, 403)
```

# Access Tuple items

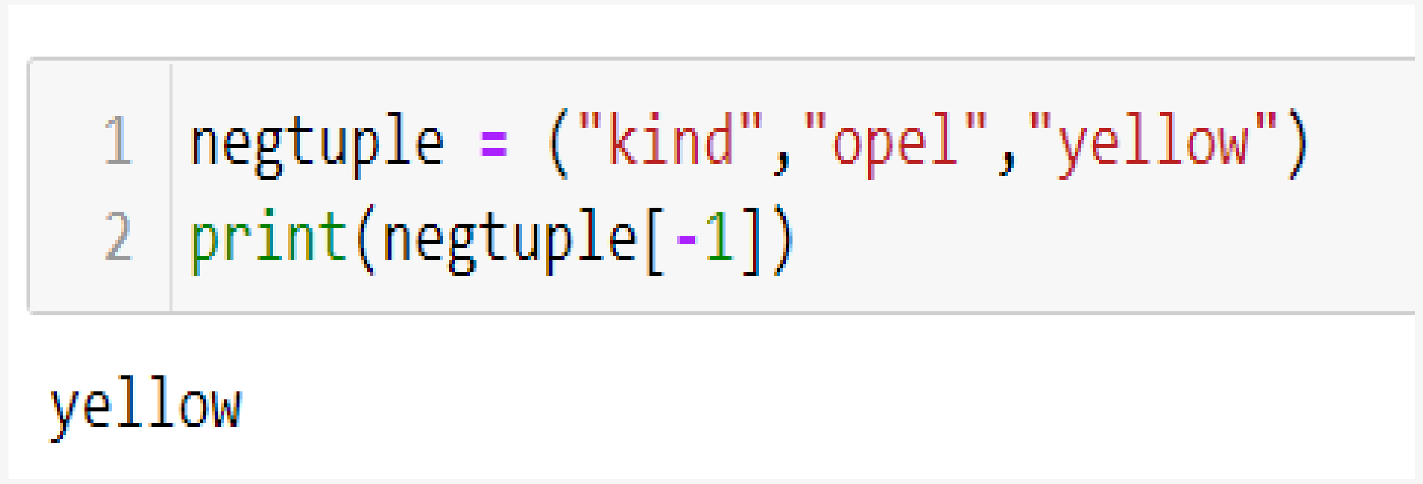
- Access Tuple Items



```
actuple = ("brb","lama","horse")
print(actuple[1])
```

```
lama
```

- Negative Indexing



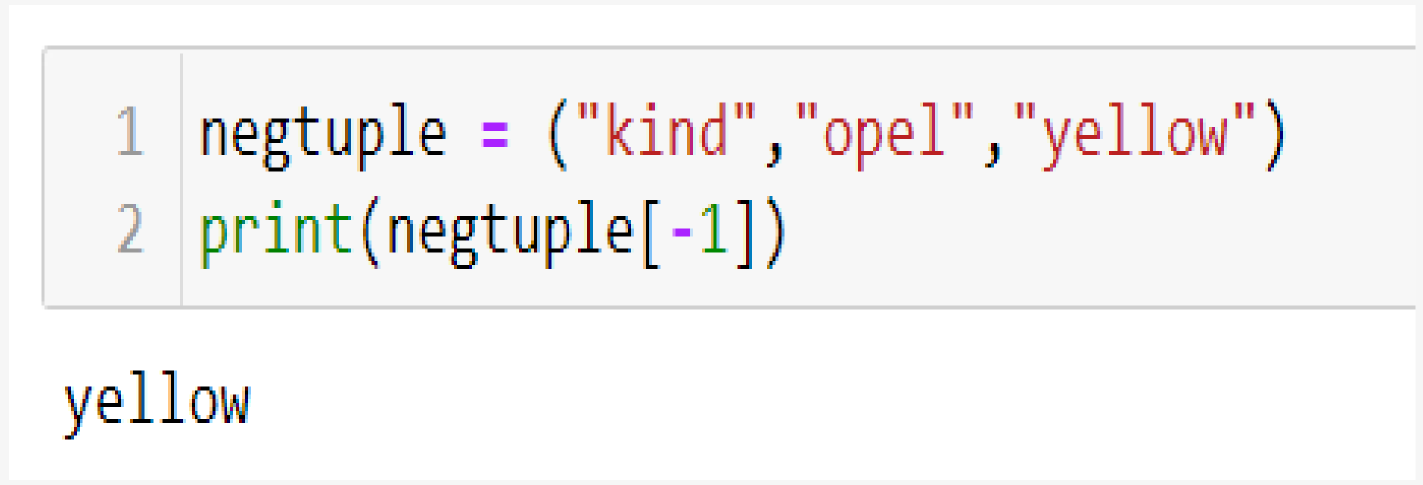
```
negtuple = ("kind","opel","yellow")
print(negtuple[-1])
```

```
yellow
```

The Learning Hub

## Access Tuple items

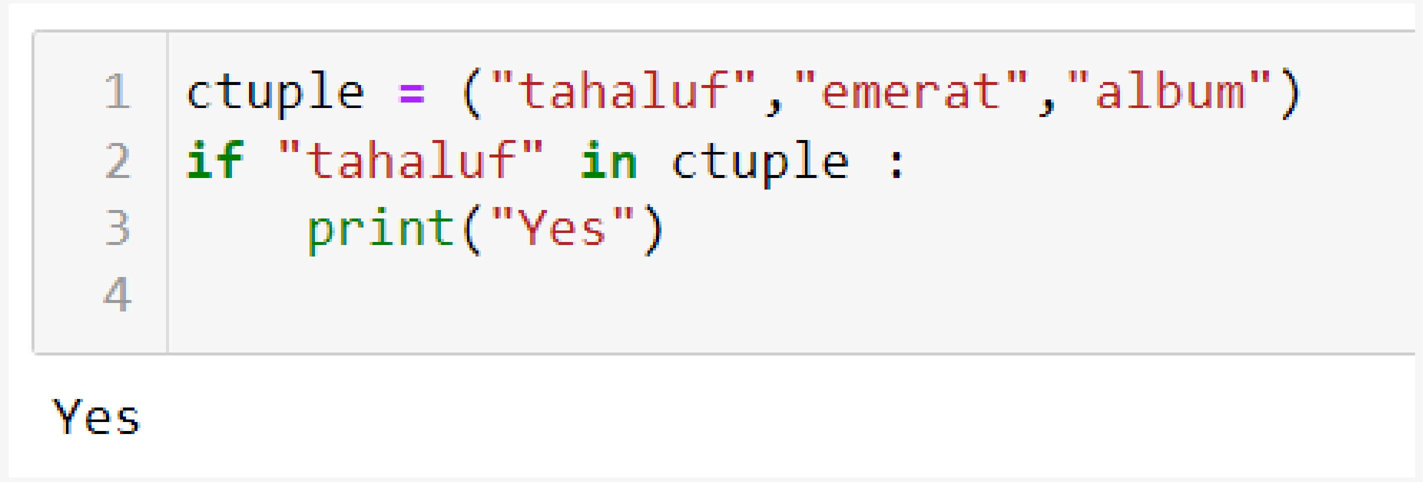
- **Range of Indexes**



```
negtuple = ("kind","opel","yellow")
print(negtuple[-1])
```

```
yellow
```

- **Check if Item Exists**



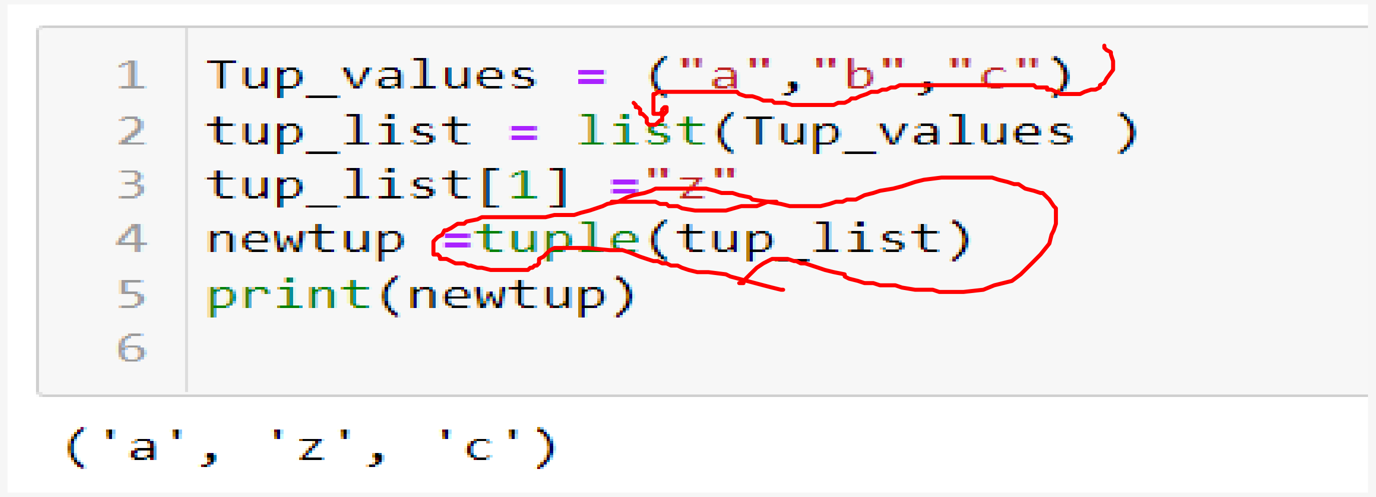
```
ctuple = ("tahaluf","emerat","album")
if "tahaluf" in ctuple :
    print("Yes")

```

```
Yes
```

The Learning Hub

## Update Tuples

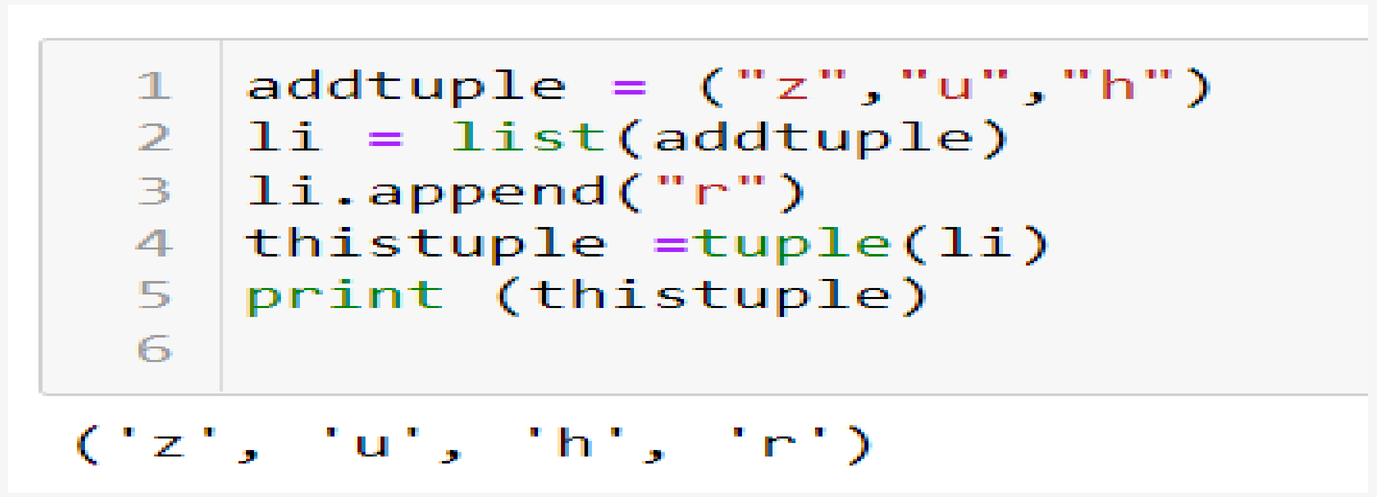
- Change Tuple Values



```
Tup_values = ("a","b","c")
tup_list = list(Tup_values )
tup_list[1] ="z"
newtup =tuple(tup_list)
print(newtup)
```

```
('a', 'z', 'c')
```

- Add Items



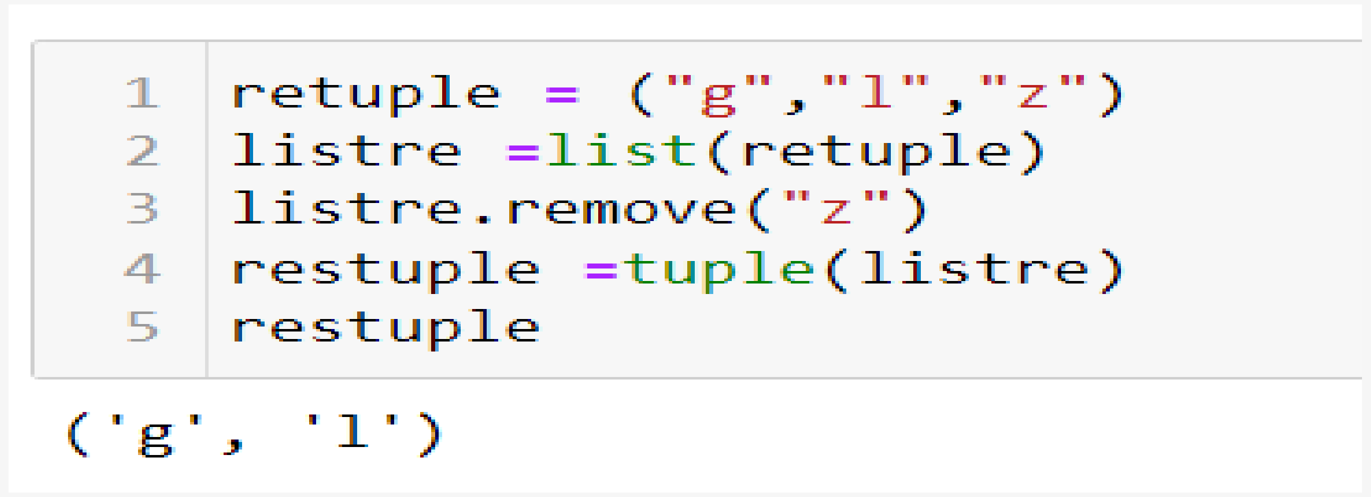
```
addtuple = ("z","u","h")
li = list(addtuple)
li.append("r")
thistuple =tuple(li)
print (thistuple)
```

```
('z', 'u', 'h', 'r')
```

The Learning Hub

## Update Tuples

- Remove Items



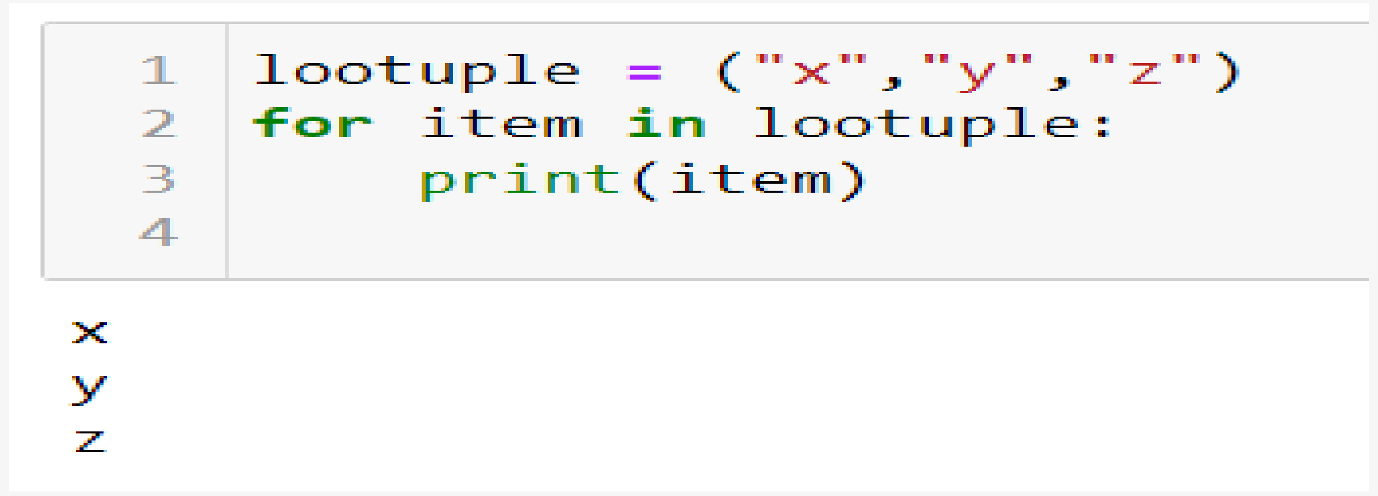
```
retuple = ("g","l","z")
listre =list(retuple)
listre.remove("z")
restuple =tuple(listre)
restuple
```

```
('g', 'l')
```

The Learning Hub

# Loop inside Tuples

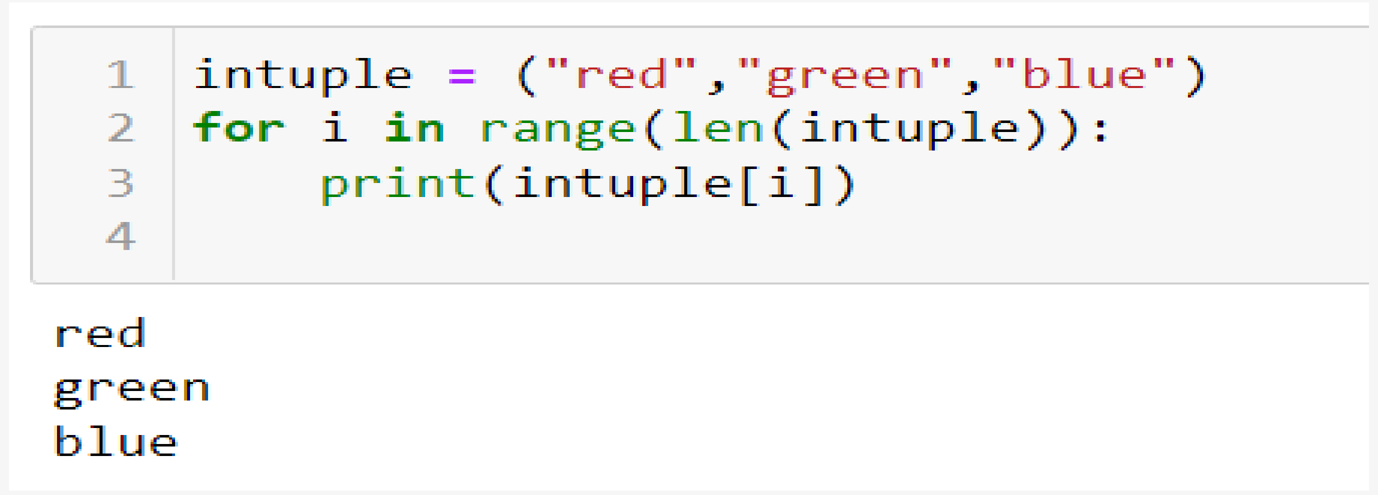
- **Loop Through a Tuple**



```
lootuple = ("x","y","z")
for item in lootuple:
    print(item)

```

```
x
y
z
```

- **Loop Through the Index Numbers**

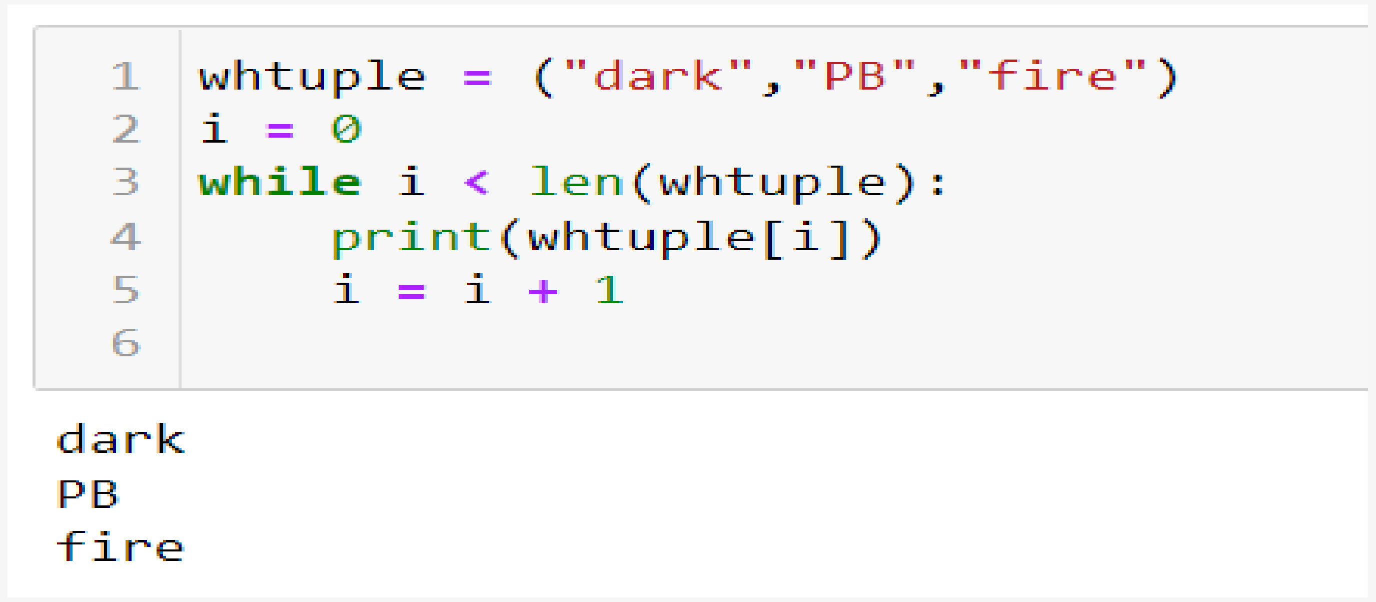


```
intuple = ("red","green","blue")
for i in range(len(intuple)):
    print(intuple[i])

```

```
red
green
blue
```

## Loop inside Tuples

- **Using a While Loop**

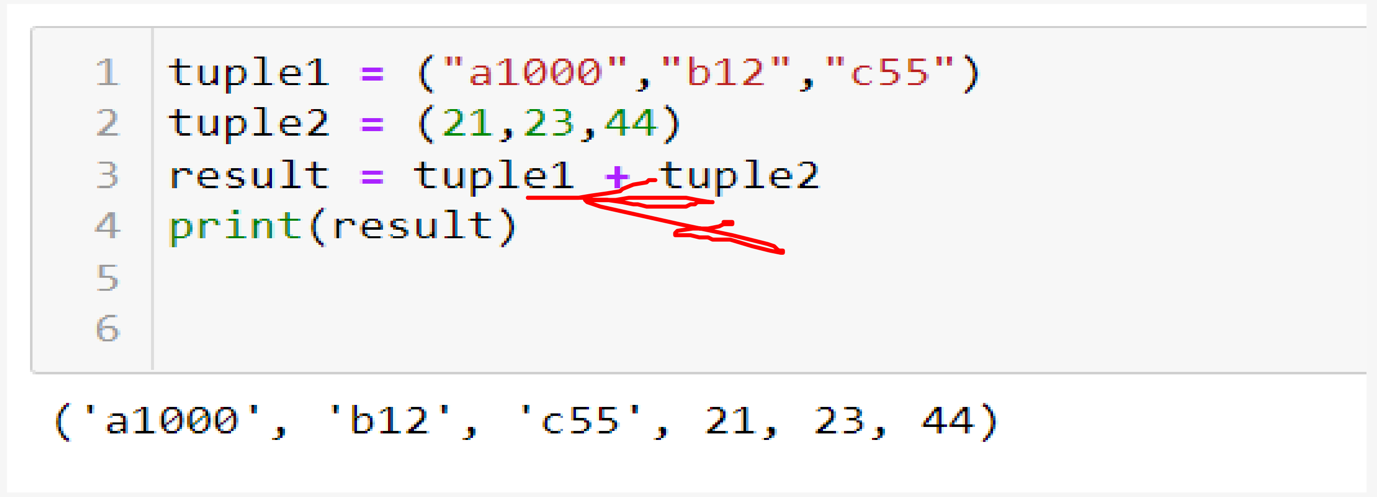


```
whtuple = ("dark","PB","fire")
i = 0
while i < len(whtuple):
    print(whtuple[i])
    i = i + 1

```

```
dark
PB
fire
```

# Join Tuples

- Join Two Tuples

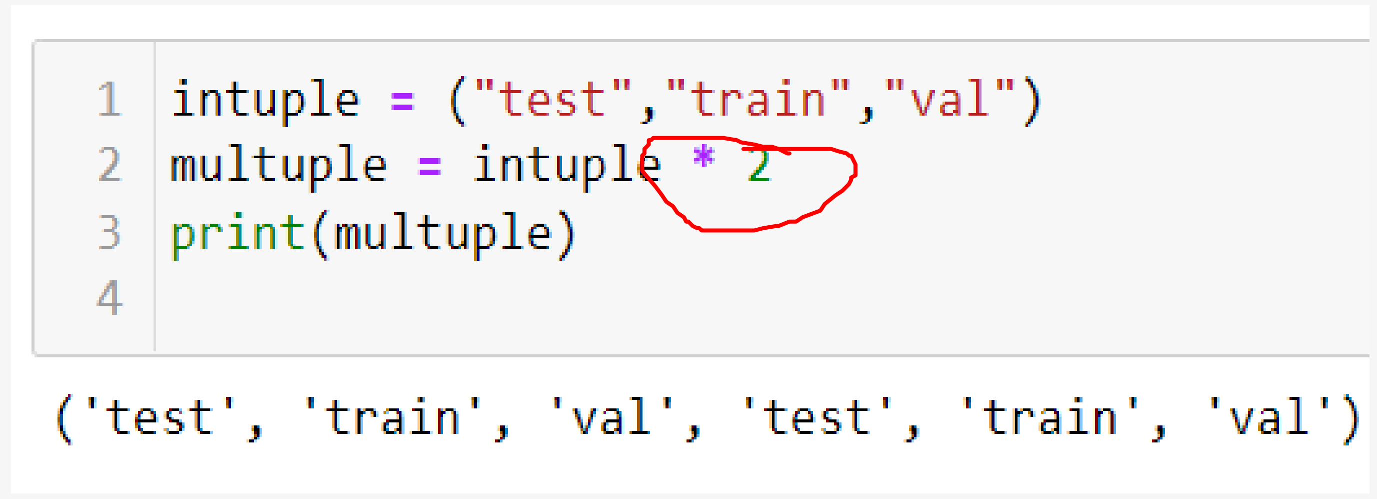


```python
tuple1 = ("a1000","b12","c55")
tuple2 = (21,23,44)
result = tuple1 + tuple2
print(result)
```

```
('a1000', 'b12', 'c55', 21, 23, 44)
```

- Multiply Tuples

$\rightarrow$ * 2



```python
intuple = ("test","train","val")
multuple = intuple * 2
print(multuple)
```

```
('test', 'train', 'val', 'test', 'train', 'val')
```

The Learning Hub

# Tuples Methods

| Method | Description |
|---|---|
| count() | Returns the number of times a specified value occurs in a tuple |
| index() | Searches the tuple for a specified value and returns the position of where it was found |

The Learning Hub

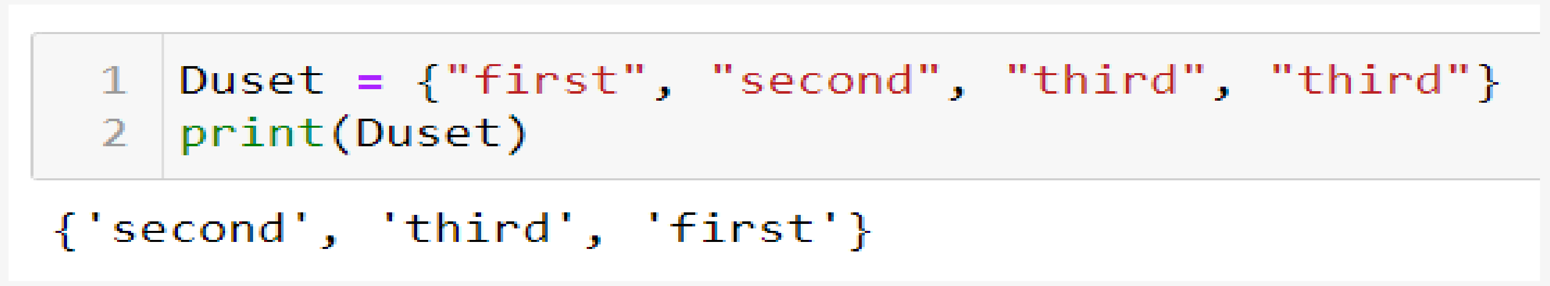
## Sets

The Learning Hub

# Introduction to Sets

- **Set:** is a collection which is unordered and unindexed **(immutable).**

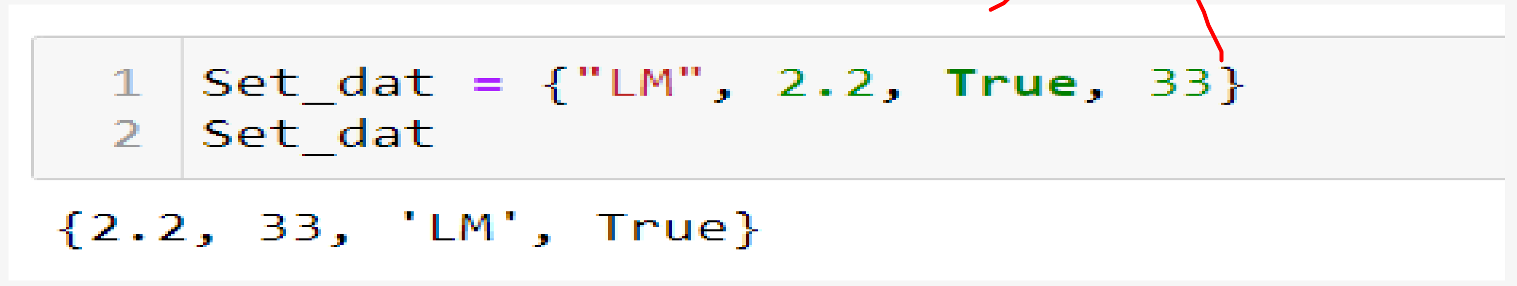
- **Duplicates not allowed**



```
Duset = {"first", "second", "third", "third"}
print(Duset)
```

```
{'second', 'third', 'first'}
```

- **Set Data Types**



```
Set_dat = {"LM", 2.2, True, 33}
Set_dat
```

```
{2.2, 33, 'LM', True}
```

L1 = [1, 2, 3, 3]

S = Set(L1)

S = {1, 2, 3}

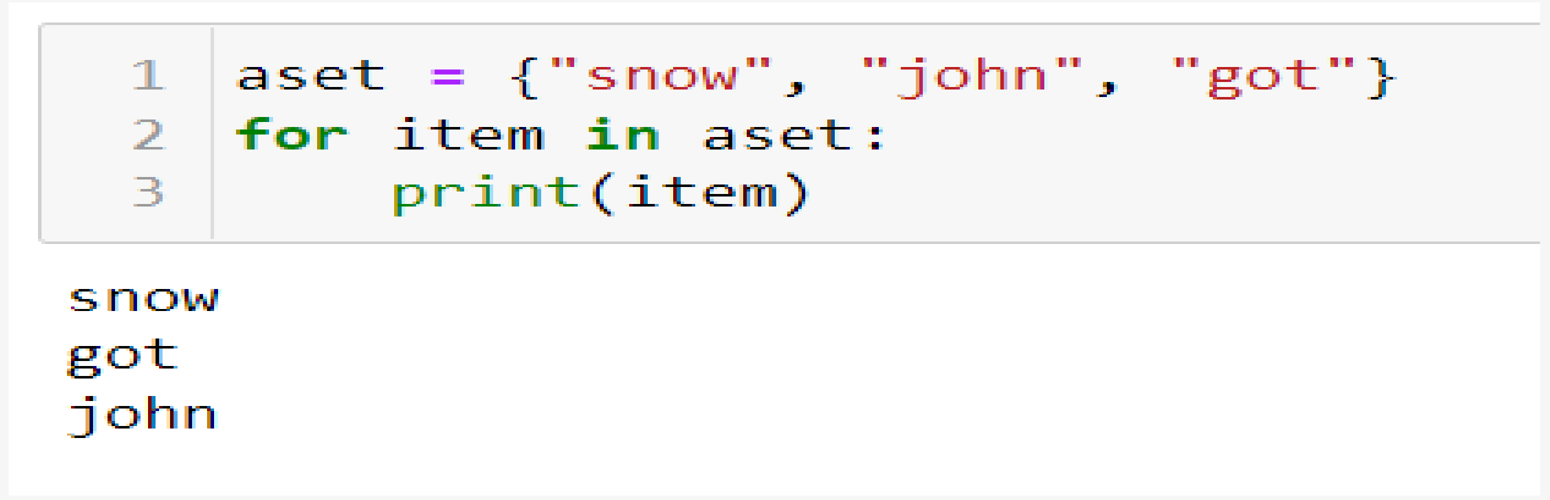
3 1 2

2 1 3

2 2 1

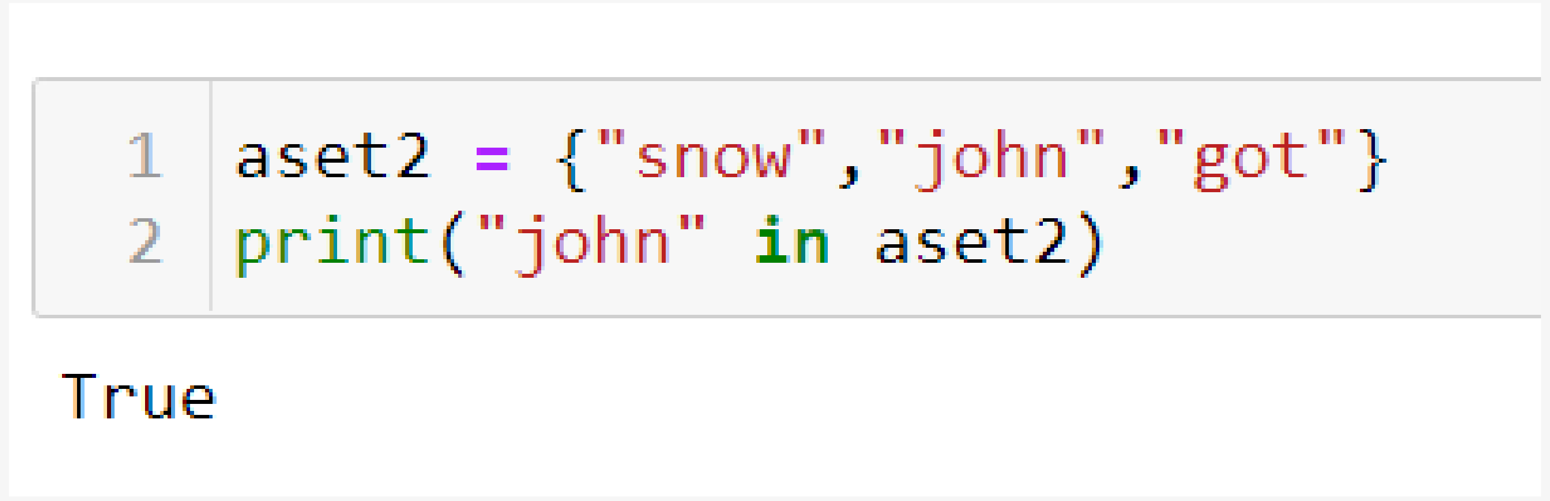
## Access Set Items

- Access Items



```
aset = {"snow", "john", "got"}
for item in aset:
    print(item)
```

```
snow
got
john
```



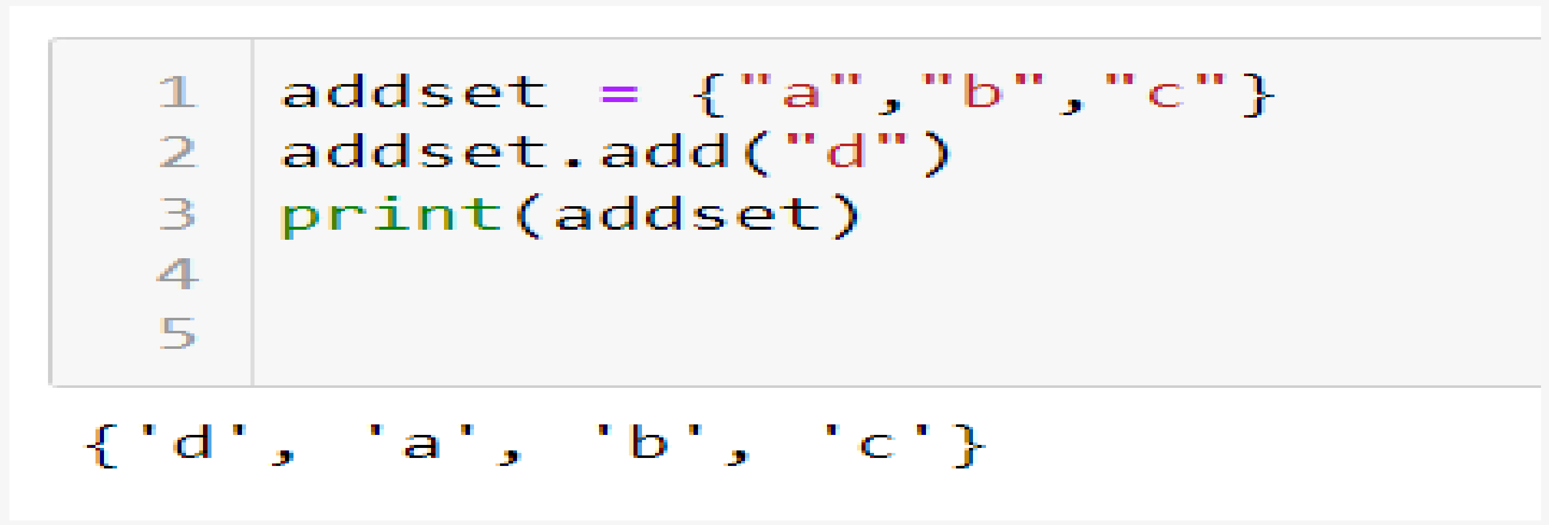
```
aset2 = {"snow","john","got"}
print("john" in aset2)
```

```
True
```

The Learning Hub

# Add Set Items

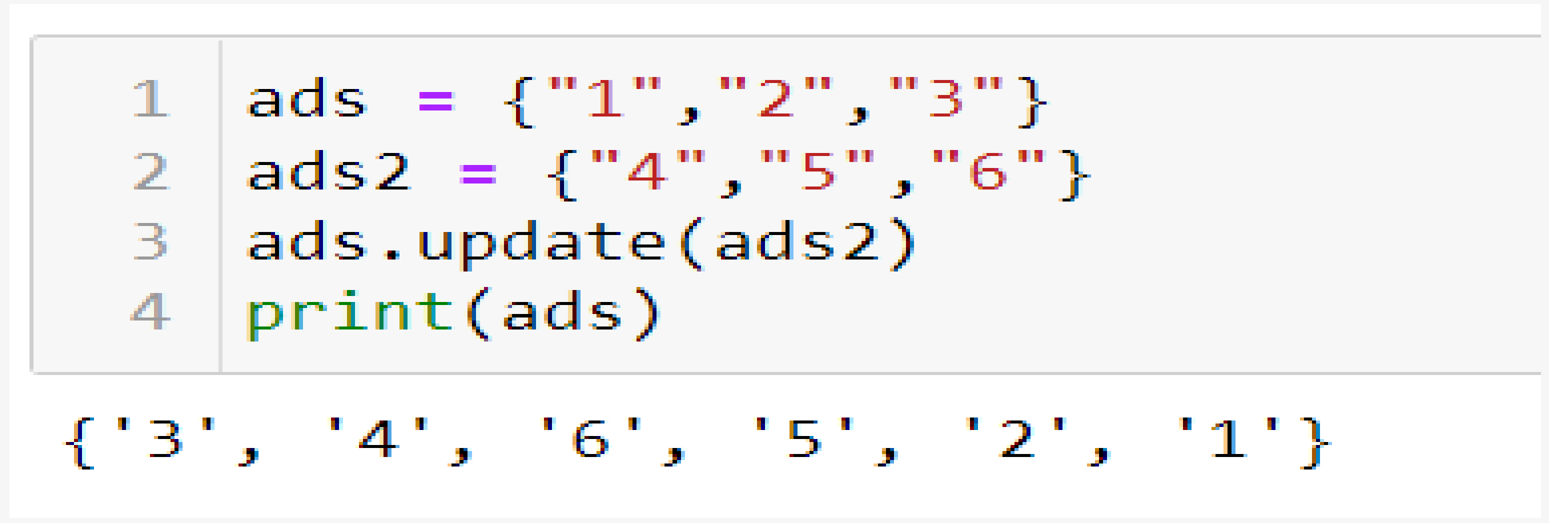
- **Add Items**



```
addset = {"a","b","c"}
addset.add("d")
print(addset)
```

```
{'d', 'a', 'b', 'c'}
```

- **Add Sets**

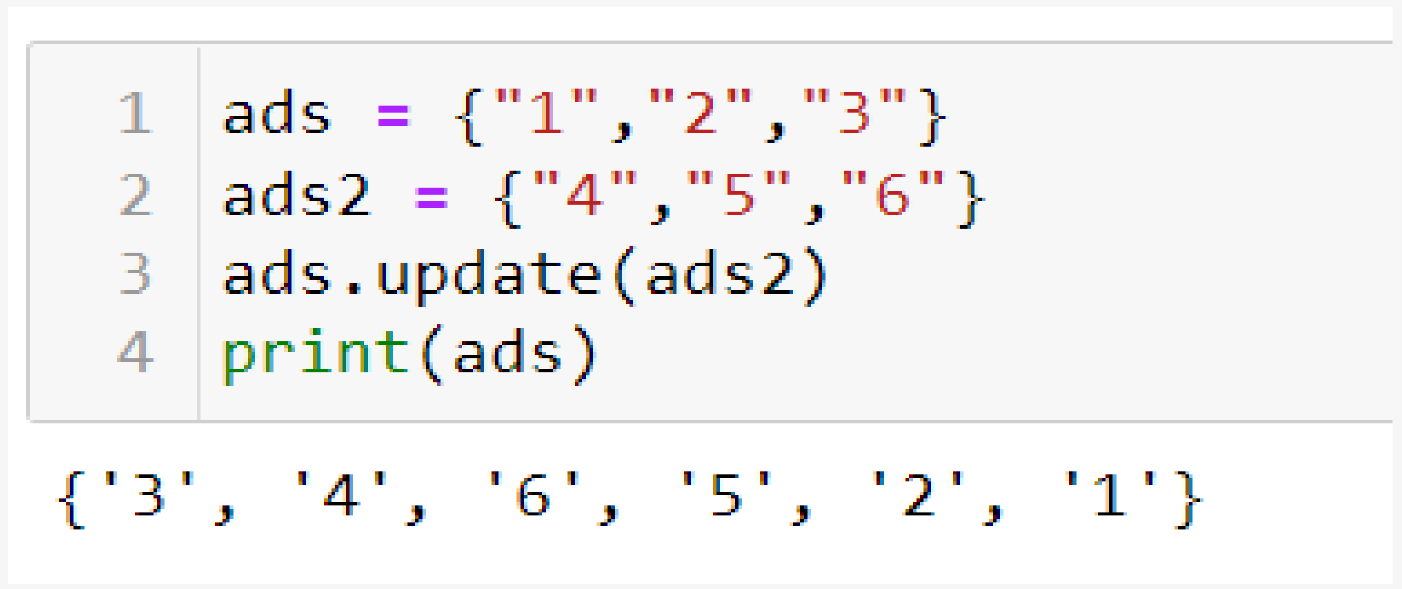


```
ads = {"1","2","3"}
ads2 = {"4","5","6"}
ads.update(ads2)
print(ads)
```

```
{'3', '4', '6', '5', '2', '1'}
```

## Add Set Items

- **Add Any Iterable**



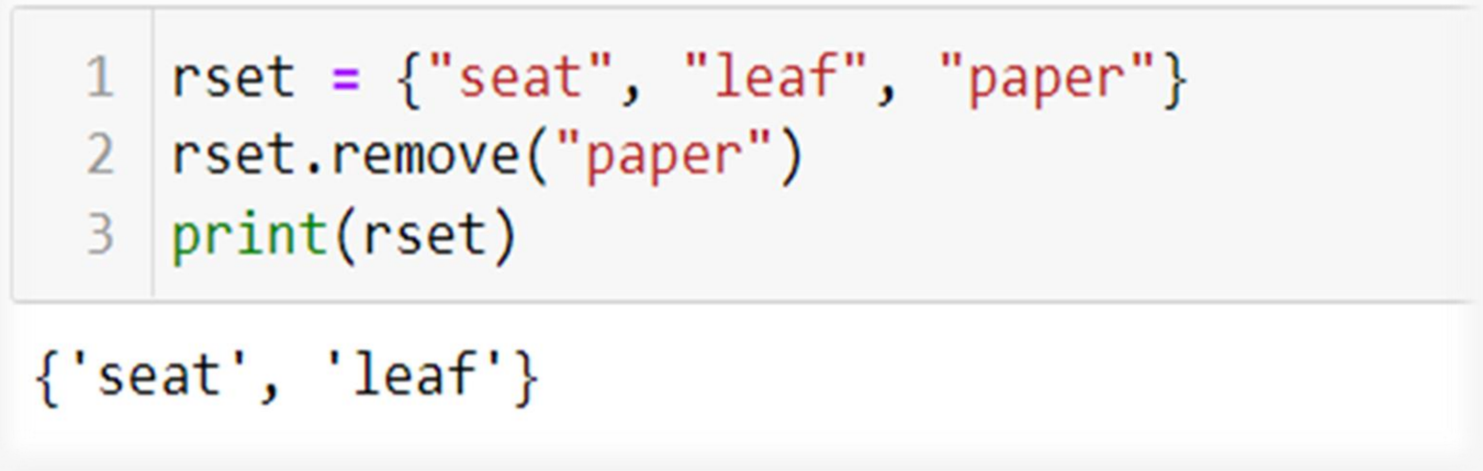
```
ads = {"1","2","3"}
ads2 = {"4","5","6"}
ads.update(ads2)
print(ads)
```

```
{'3', '4', '6', '5', '2', '1'}
```

The Learning Hub

# Remove Set Items

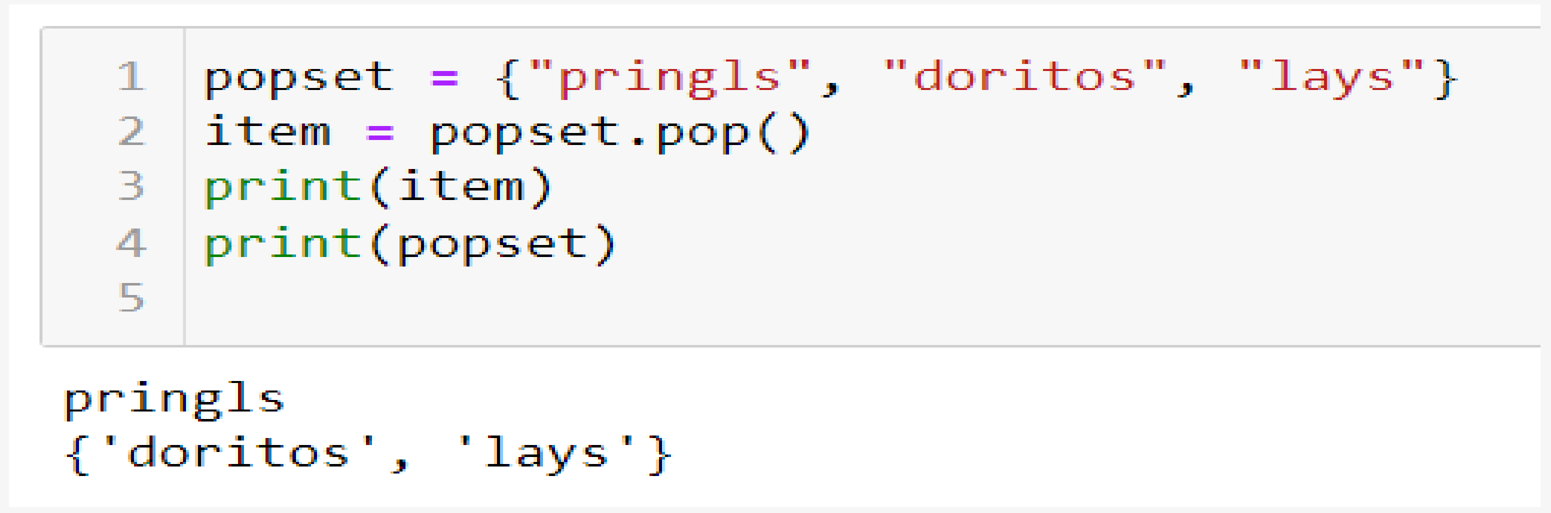
- **Remove Item using remove**



```
rset = {"seat", "leaf", "paper"}
rset.remove("paper")
print(rset)
```

```
{'seat', 'leaf'}
```

- **Remove Item using pop**



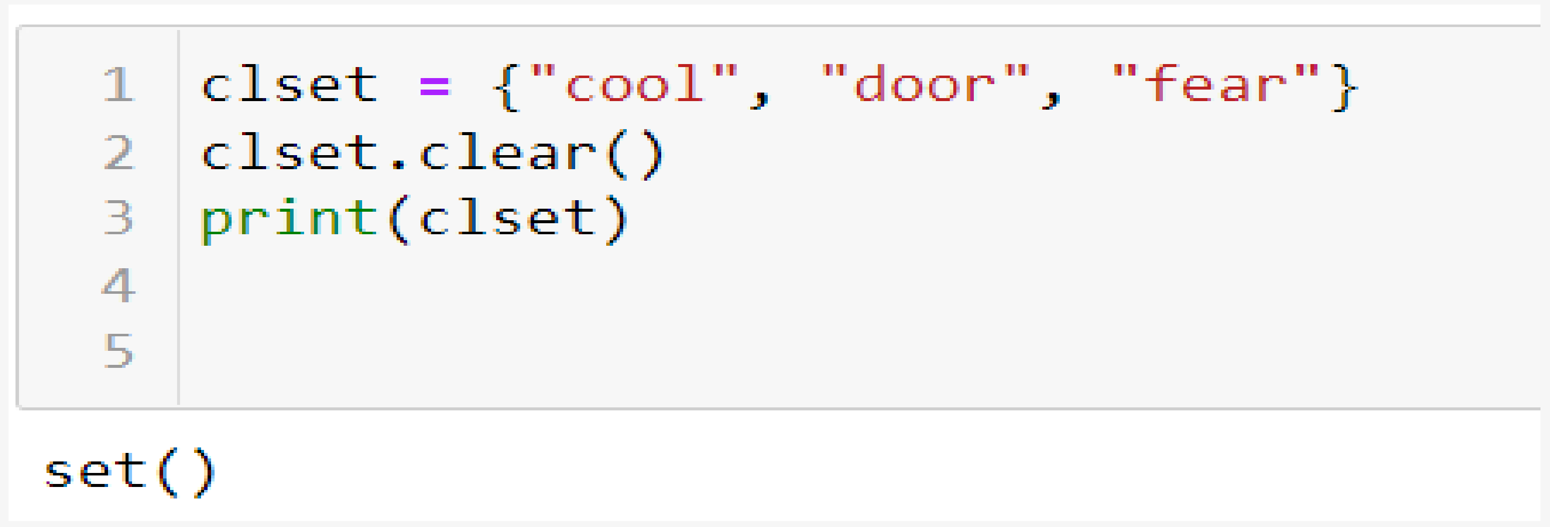
```
popset = {"pringls", "doritos", "lays"}
item = popset.pop()
print(item)
print(popset)

```

```
pringls
{'doritos', 'lays'}
```

The Learning Hub

## Remove Set Items

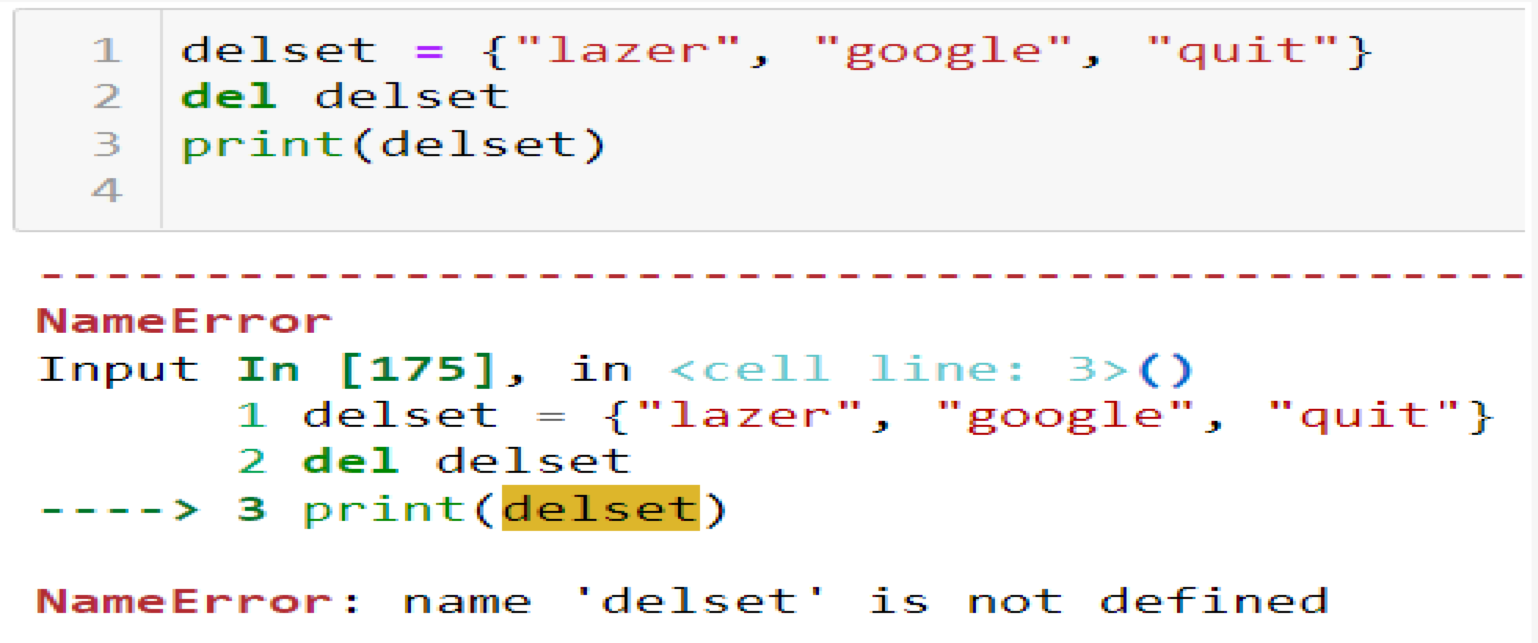
- **Remove Item using clear**



```
clset = {"cool", "door", "fear"}
clset.clear()
print(clset)
```

```
set()
```

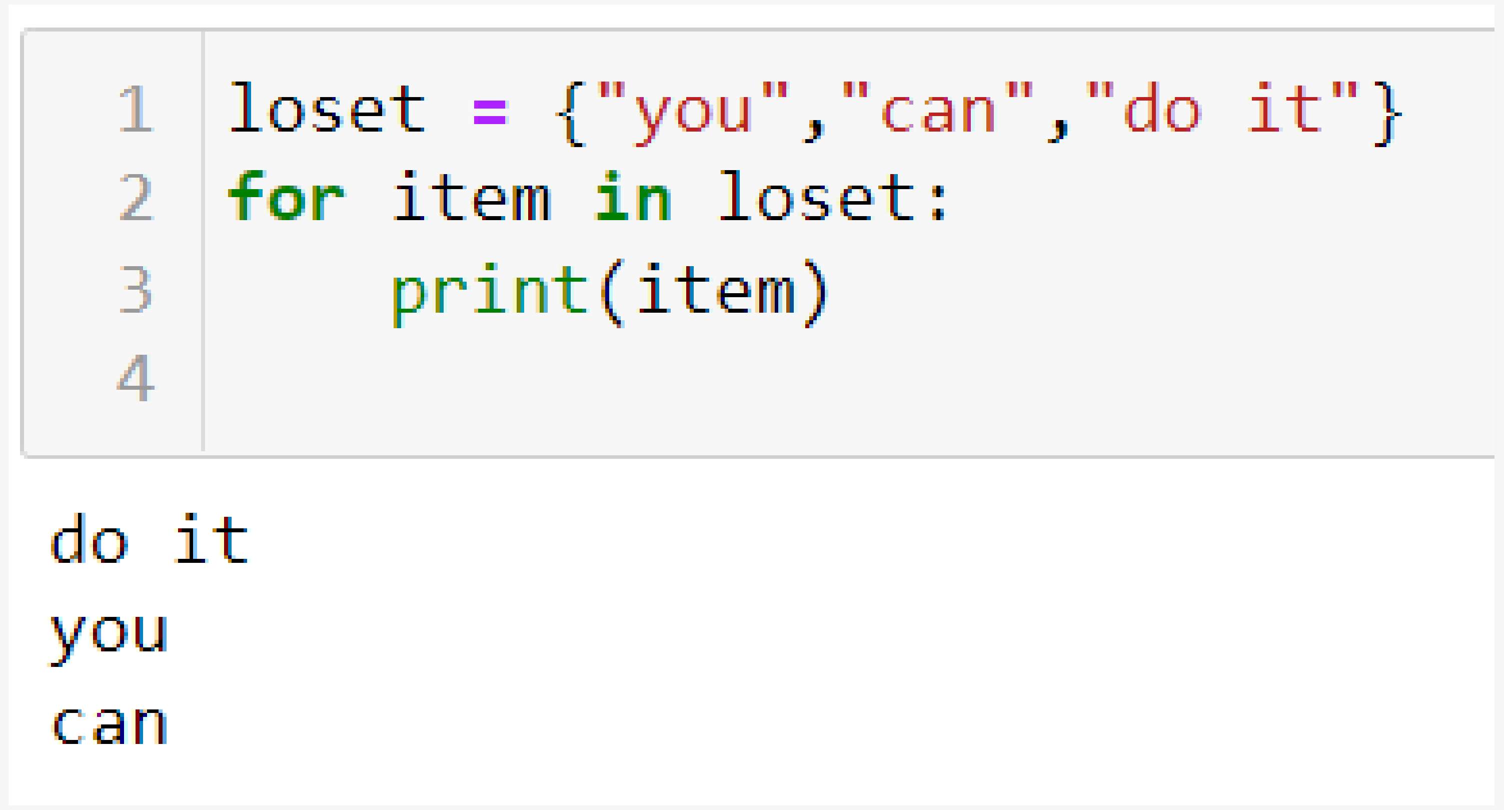
- **Remove Item using delete**



```
delset = {"lazer", "google", "quit"}
del delset
print(delset)
```

```
---------------------------------------------------------------------------
NameError
Input In [175], in <cell line: 3>()
      1 delset = {"lazer", "google", "quit"}
      2 del delset
----> 3 print(delset)

NameError: name 'delset' is not defined
```

## Loop Sets



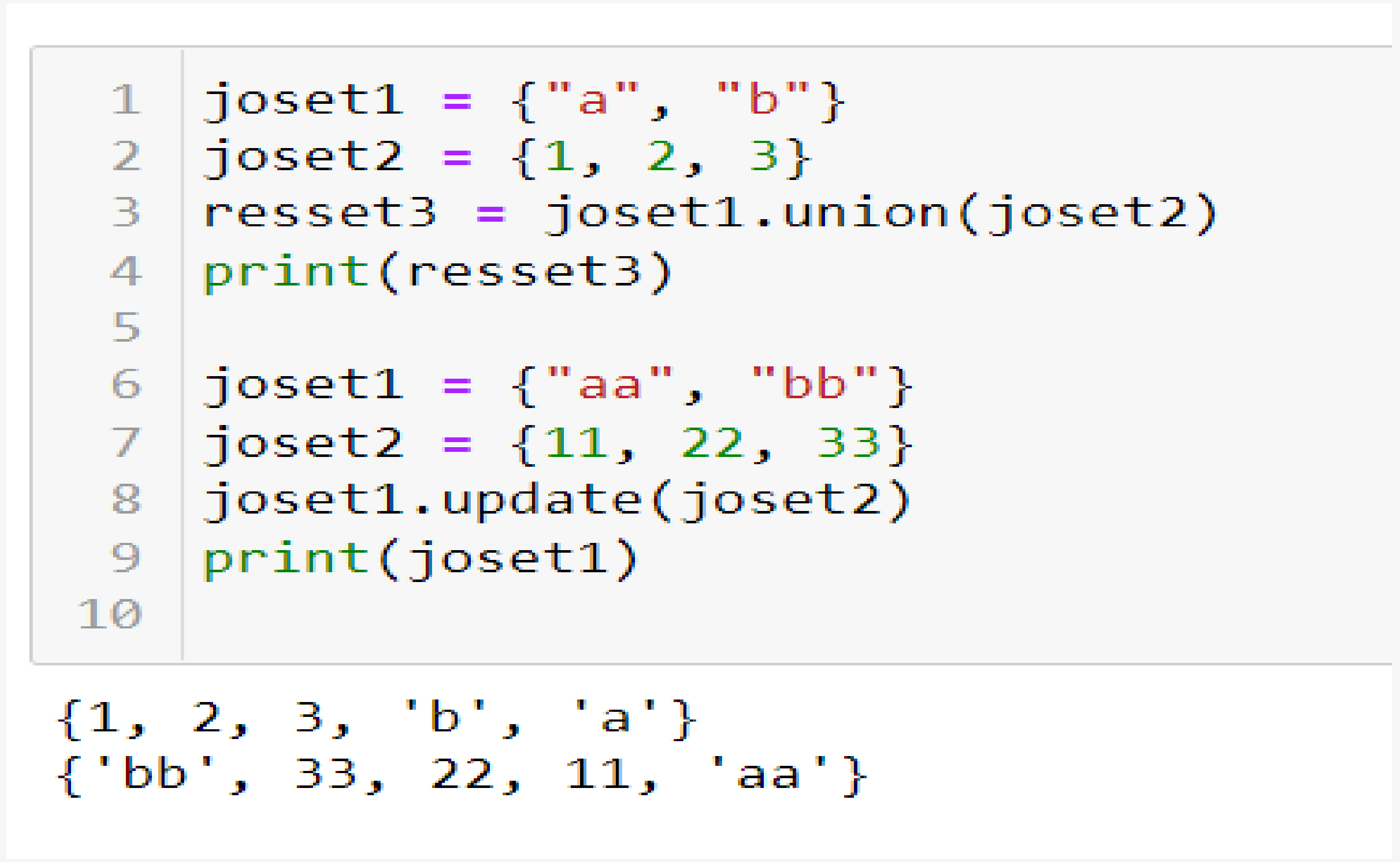
```python
loset = {"you","can","do it"}
for item in loset:
    print(item)

```

```
do it
you
can
```

The Learning Hub

## Join Sets

- **Join Two Sets**

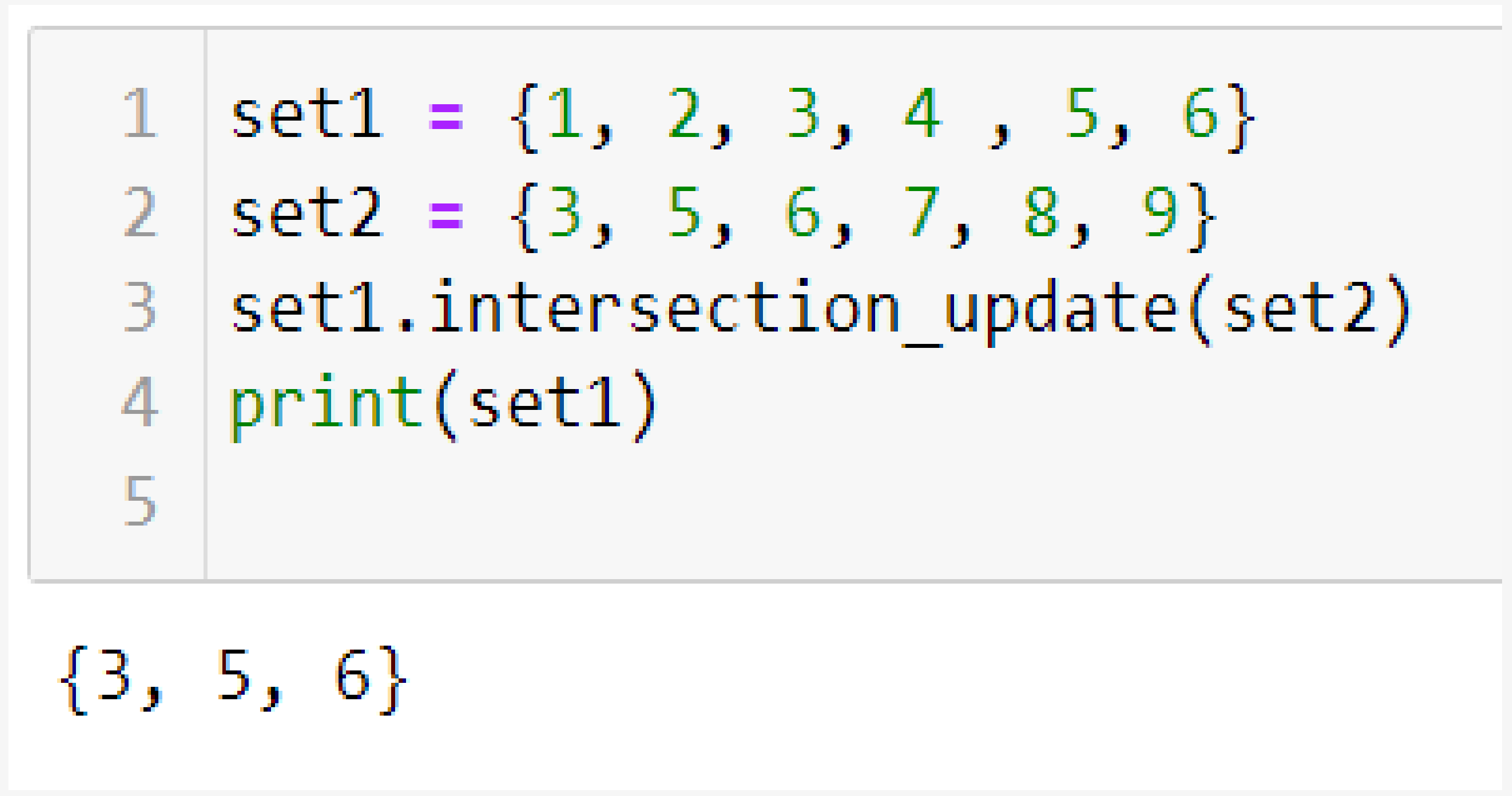


```
joset1 = {"a", "b"}
joset2 = {1, 2, 3}
resset3 = joset1.union(joset2)
print(resset3)

joset1 = {"aa", "bb"}
joset2 = {11, 22, 33}
joset1.update(joset2)
print(joset1)

```

```
{1, 2, 3, 'b', 'a'}
{'bb', 33, 22, 11, 'aa'}
```

## Join Sets

- **Select the Duplicates**



```python
set1 = {1, 2, 3, 4 , 5, 6}
set2 = {3, 5, 6, 7, 8, 9}
set1.intersection_update(set2)
print(set1)

```

```
{3, 5, 6}
```

# Sets Methods

| Method | Description |
|---|---|
| add() | Adds an element to the set |
| clear() | Removes all the elements from the set |
| copy() | Returns a copy of the set |
| difference() | Returns a set containing the difference between two or more sets |
| difference_update() | Removes the items in this set that are also included in another, specified set |
| discard() | Remove the specified item |
| intersection() | Returns a set, that is the intersection of two other sets |
| intersection_update() | Removes the items in this set that are not present in other, specified set(s) |
| isdisjoint() | Returns whether two sets have a intersection or not |
| issubset() | Returns whether another set contains this set or not |
| issuperset() | Returns whether this set contains another set or not |
| pop() | Removes an element from the set |
| remove() | Removes the specified element |

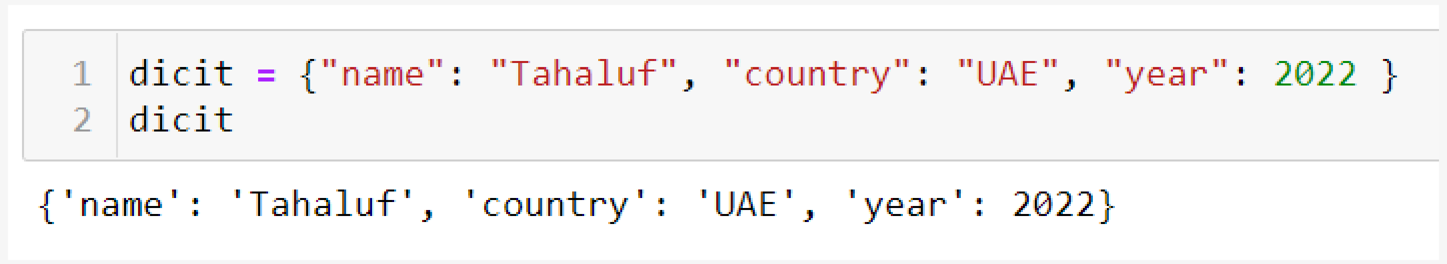
The Learning Hub

## Dictionary

The Learning Hub

# Introduction to Dictionaries

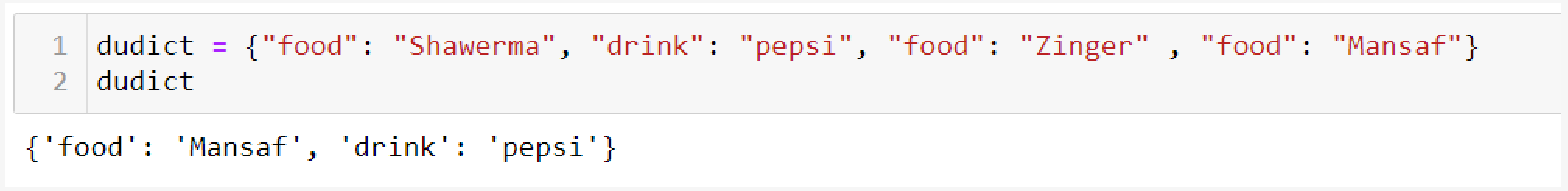
- **Dictionary:** is a collection that is ordered, changeable, and does not allow duplicates **(mutable).**
- **Dictionary Items --> (key: value pairs)**



```
dicit = {"name": "Tahaluf", "country": "UAE", "year": 2022 }
dicit
```

```
{'name': 'Tahaluf', 'country': 'UAE', 'year': 2022}
```

- **Duplicates Not Allowed**



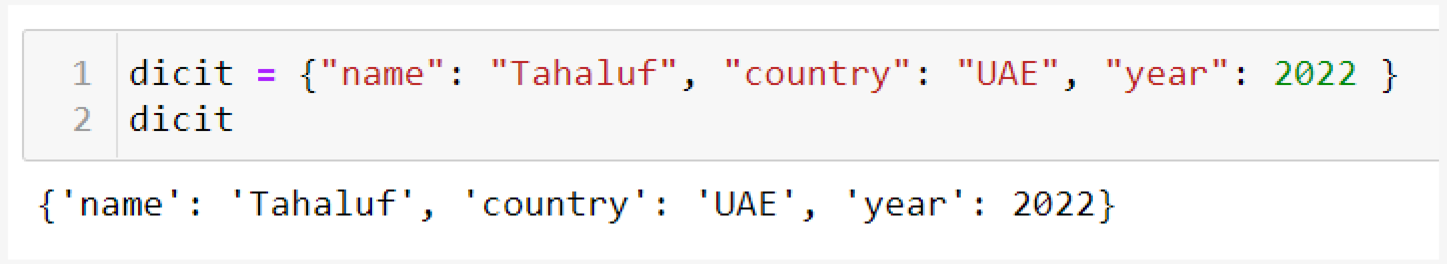
```
dudict = {"food": "Shawerma", "drink": "pepsi", "food": "Zinger" , "food": "Mansaf"}
dudict
```

```
{'food': 'Mansaf', 'drink': 'pepsi'}
```

The Learning Hub

# Introduction to Dictionaries

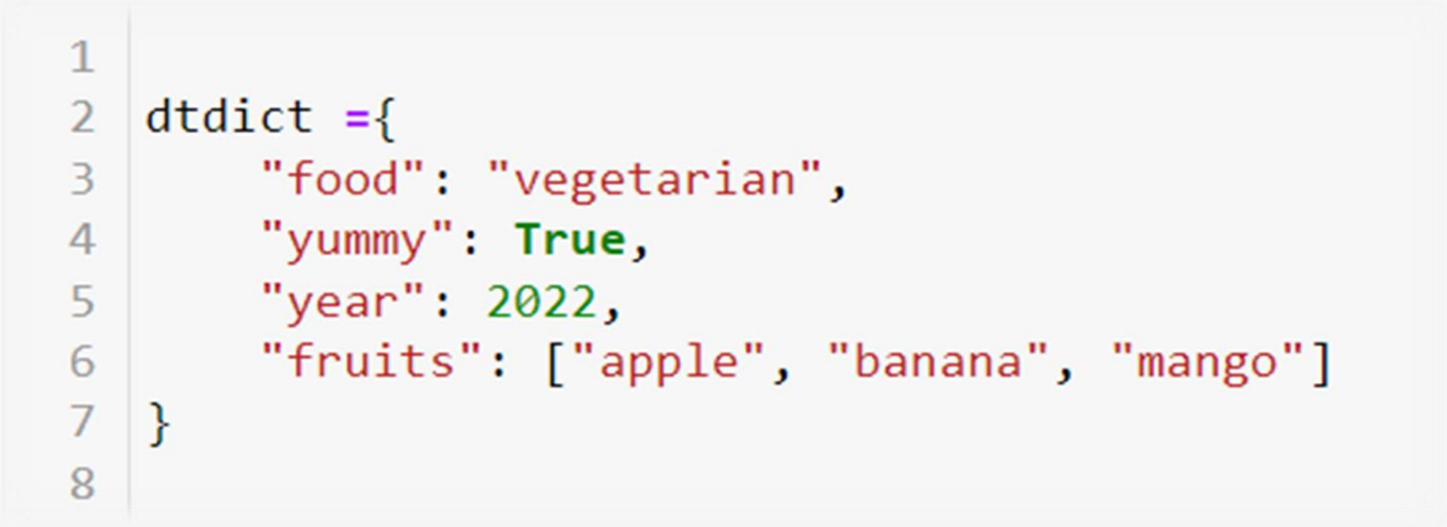
- Dictionary Data Types



```
dicit = {"name": "Tahaluf", "country": "UAE", "year": 2022 }
dicit
```

```
{'name': 'Tahaluf', 'country': 'UAE', 'year': 2022}
```

- Duplicates Not Allowed

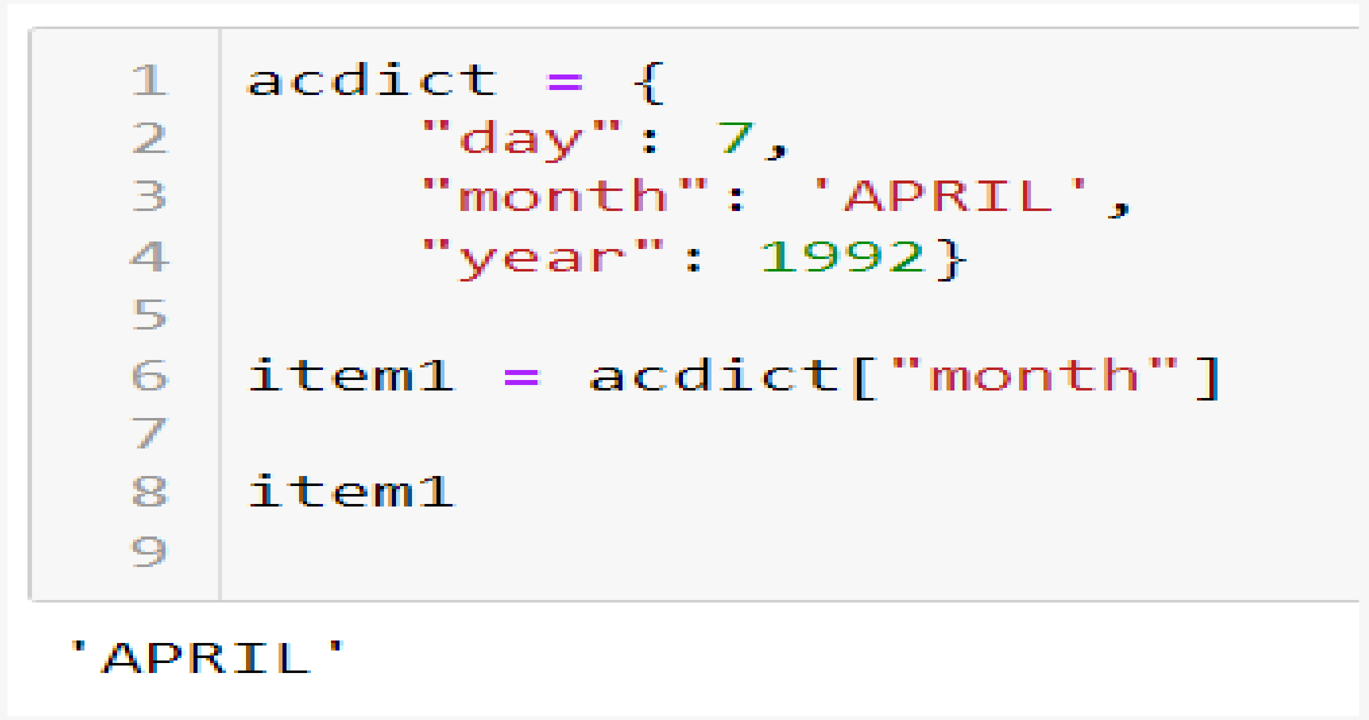


```

dtdict ={
    "food": "vegetarian",
    "yummy": True,
    "year": 2022,
    "fruits": ["apple", "banana", "mango"]
}

```

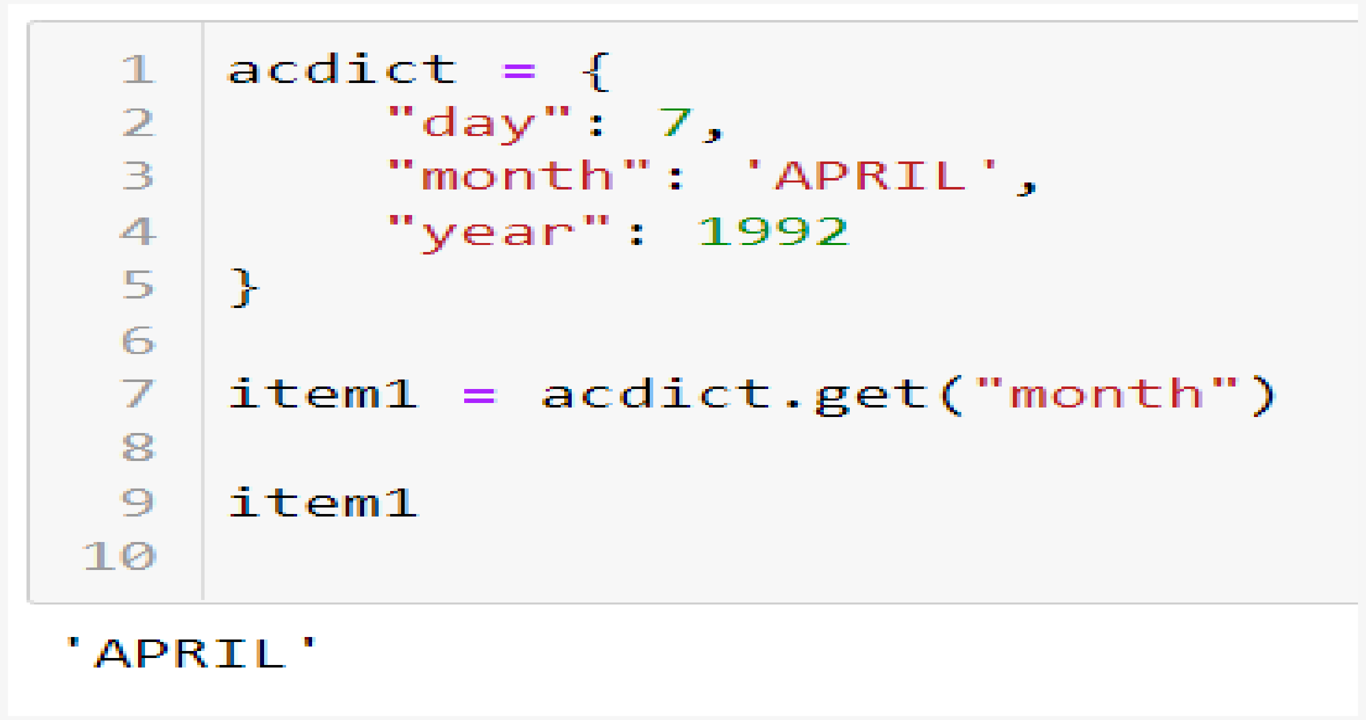
The Learning Hub

# Access Items

- **Access Items**



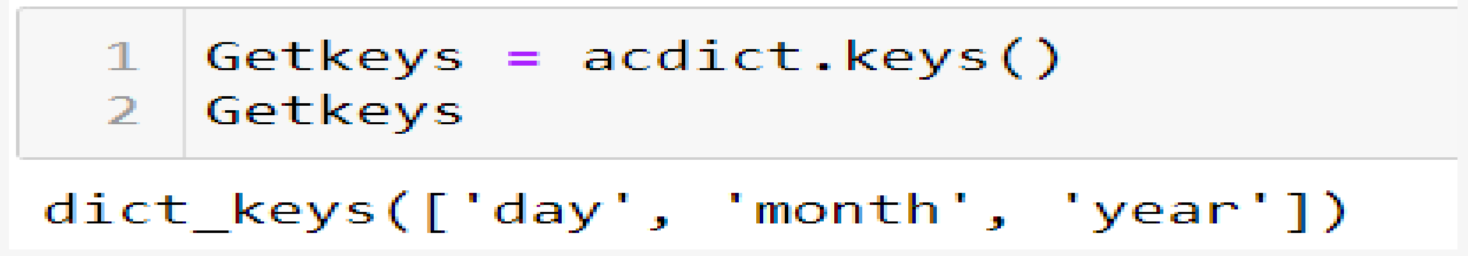
```
acdict = {
        "day": 7,
        "month": 'APRIL',
        "year": 1992}

item1 = acdict["month"]

item1

'APRIL'
```



```
acdict = {
       "day": 7,
       "month": 'APRIL',
       "year": 1992
}

item1 = acdict.get("month")

item1

'APRIL'
```

# Access Items

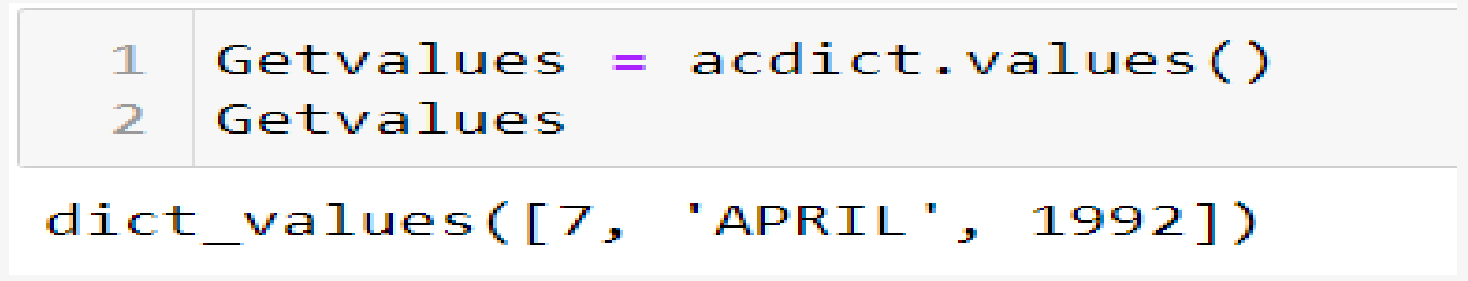
- Get Keys



```
Getkeys = acdict.keys()
Getkeys
```

```
dict_keys(['day', 'month', 'year'])
```

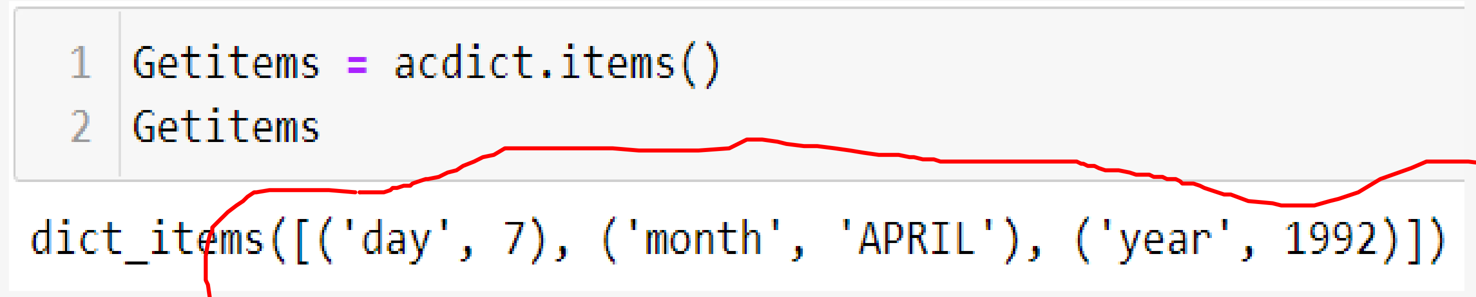
- Get Values



```
Getvalues = acdict.values()
Getvalues
```

```
dict_values([7, 'APRIL', 1992])
```

- Get Items

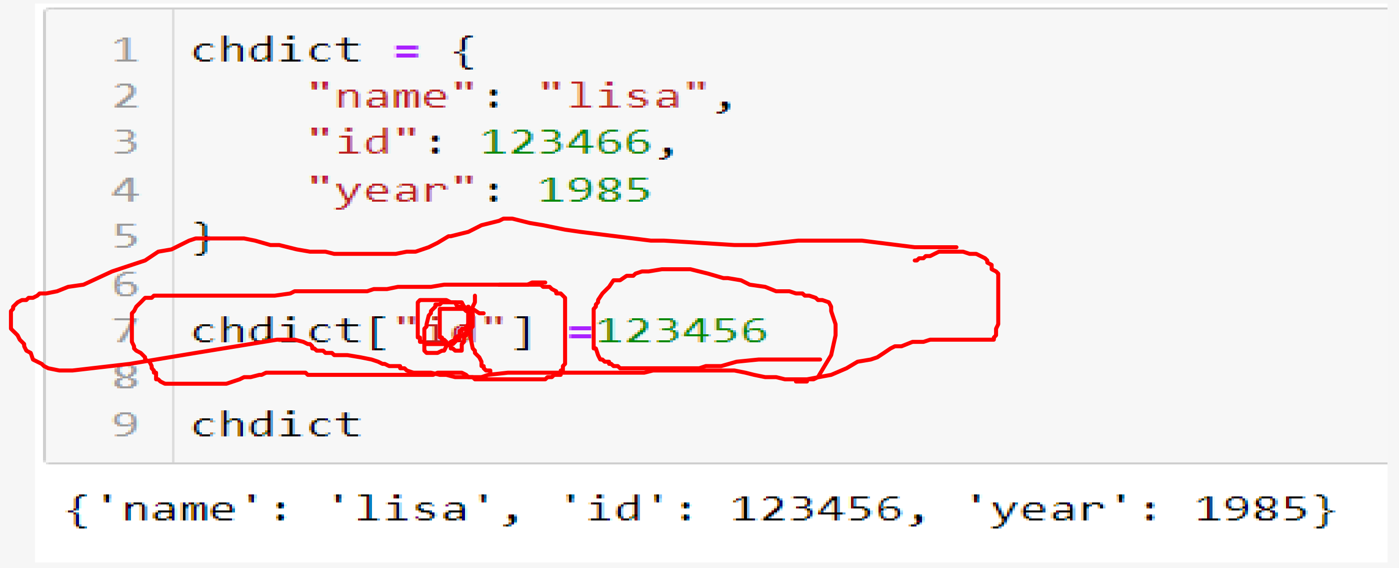


```
Getitems = acdict.items()
Getitems
```

```
dict_items([('day', 7), ('month', 'APRIL'), ('year', 1992)])
```

## Change Items

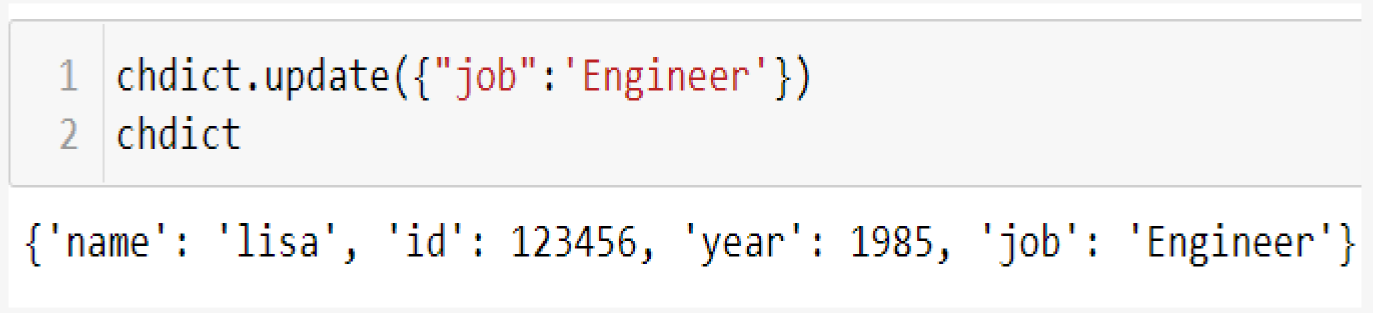
- **Change Values**



```
chdict = {
    "name": "lisa",
    "id": 123466,
    "year": 1985
}

chdict["id"] =123456

chdict
```

```
{'name': 'lisa', 'id': 123456, 'year': 1985}
```

- **Update Dictionary**



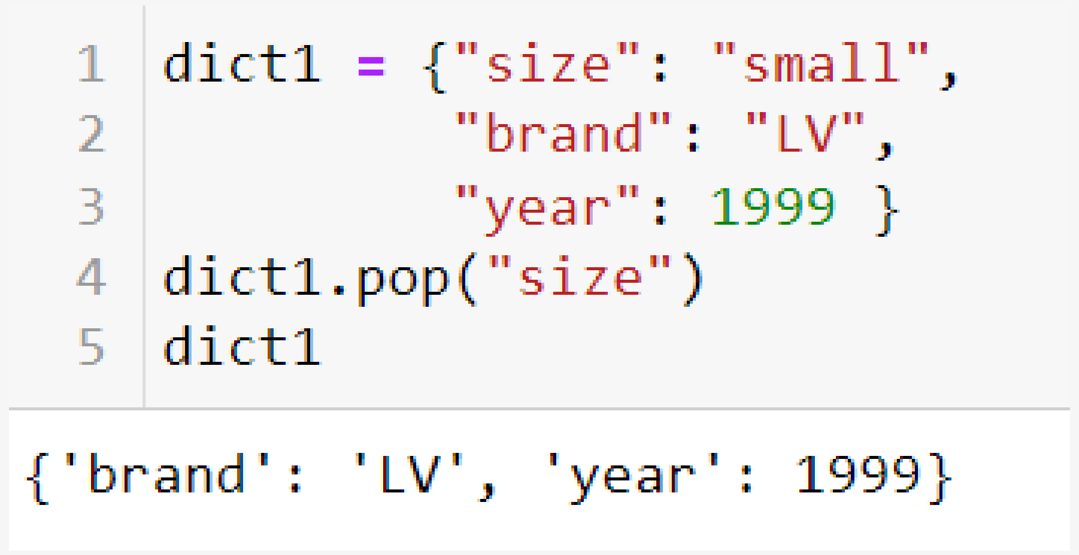
```
chdict.update({"job":'Engineer'})
chdict
```

```
{'name': 'lisa', 'id': 123456, 'year': 1985, 'job': 'Engineer'}
```

The Learning Hub

# Remove Items

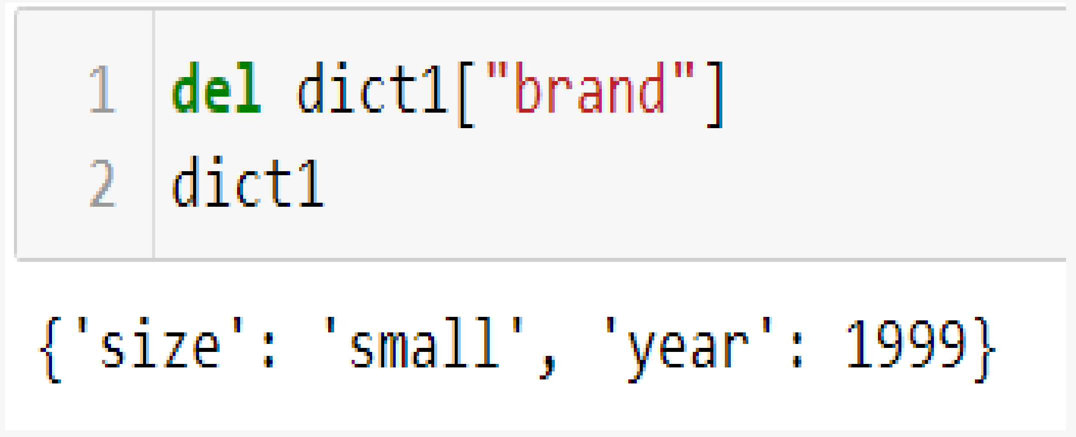
- **Remove Items using pop**



```
dict1 = {"size": "small",
         "brand": "LV",
         "year": 1999 }
dict1.pop("size")
dict1
```

```
{'brand': 'LV', 'year': 1999}
```

- **Remove Items using del**

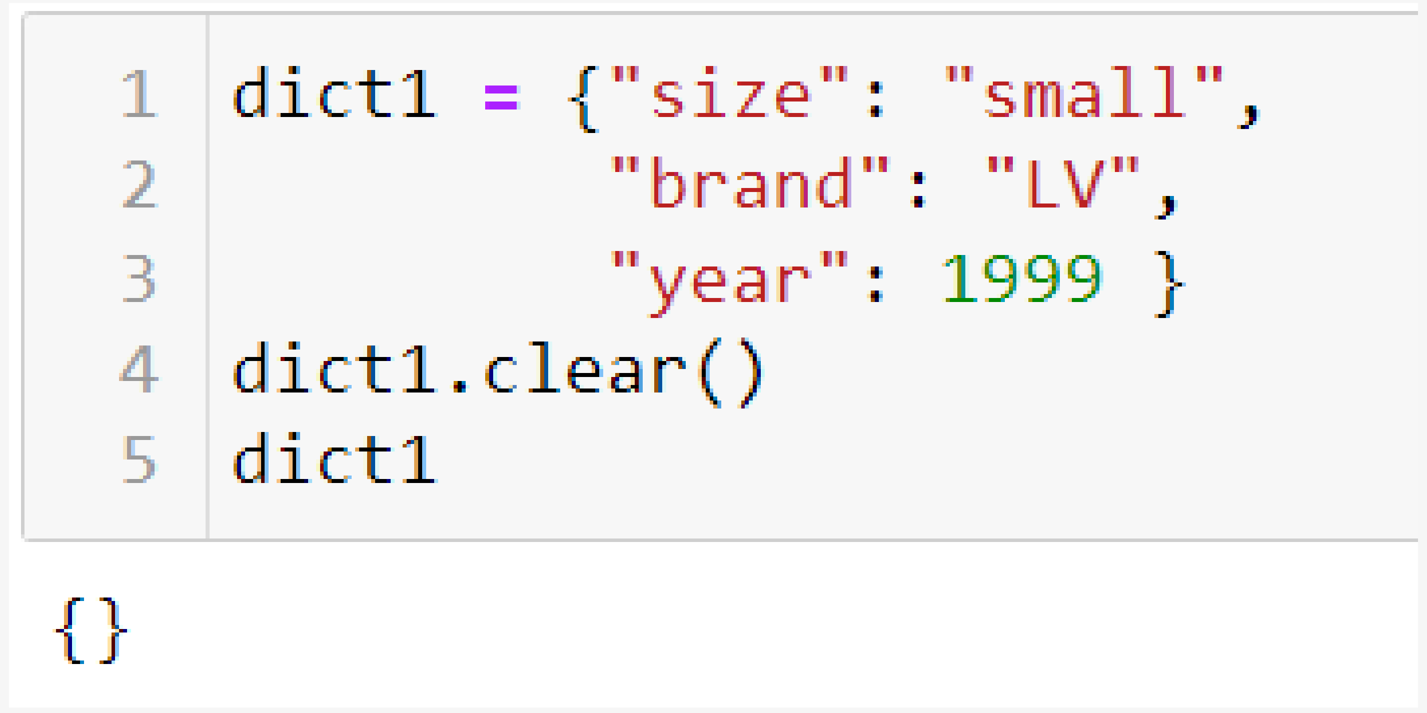


```
del dict1["brand"]
dict1
```

```
{'size': 'small', 'year': 1999}
```

## Remove Items

- **Remove Items using clear**

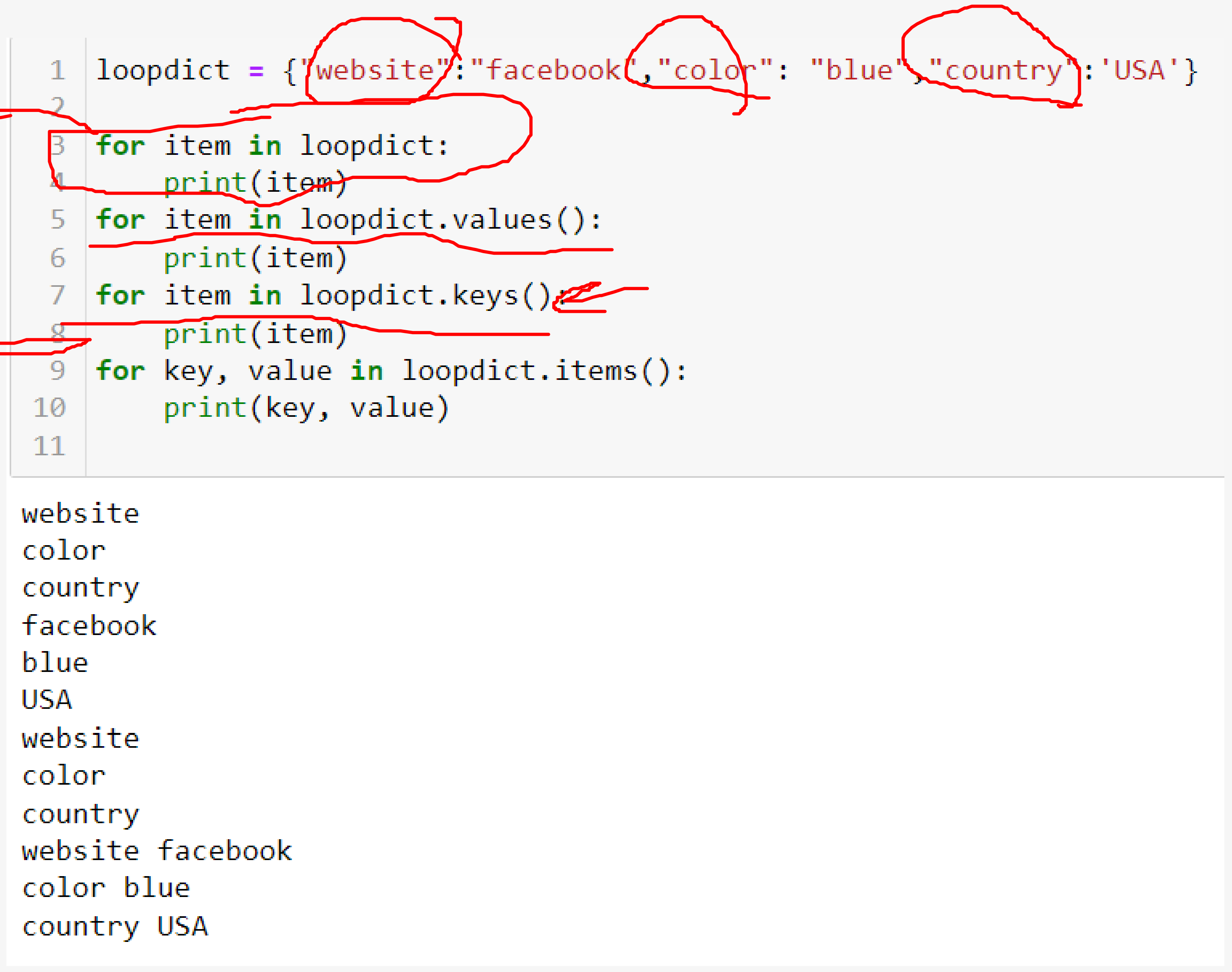


```
dict1 = {"size": "small",
          "brand": "LV",
          "year": 1999 }
dict1.clear()
dict1
```

```
{}
```

# Loop inside Dictionary

A dictionary can be looped through using the for loop.

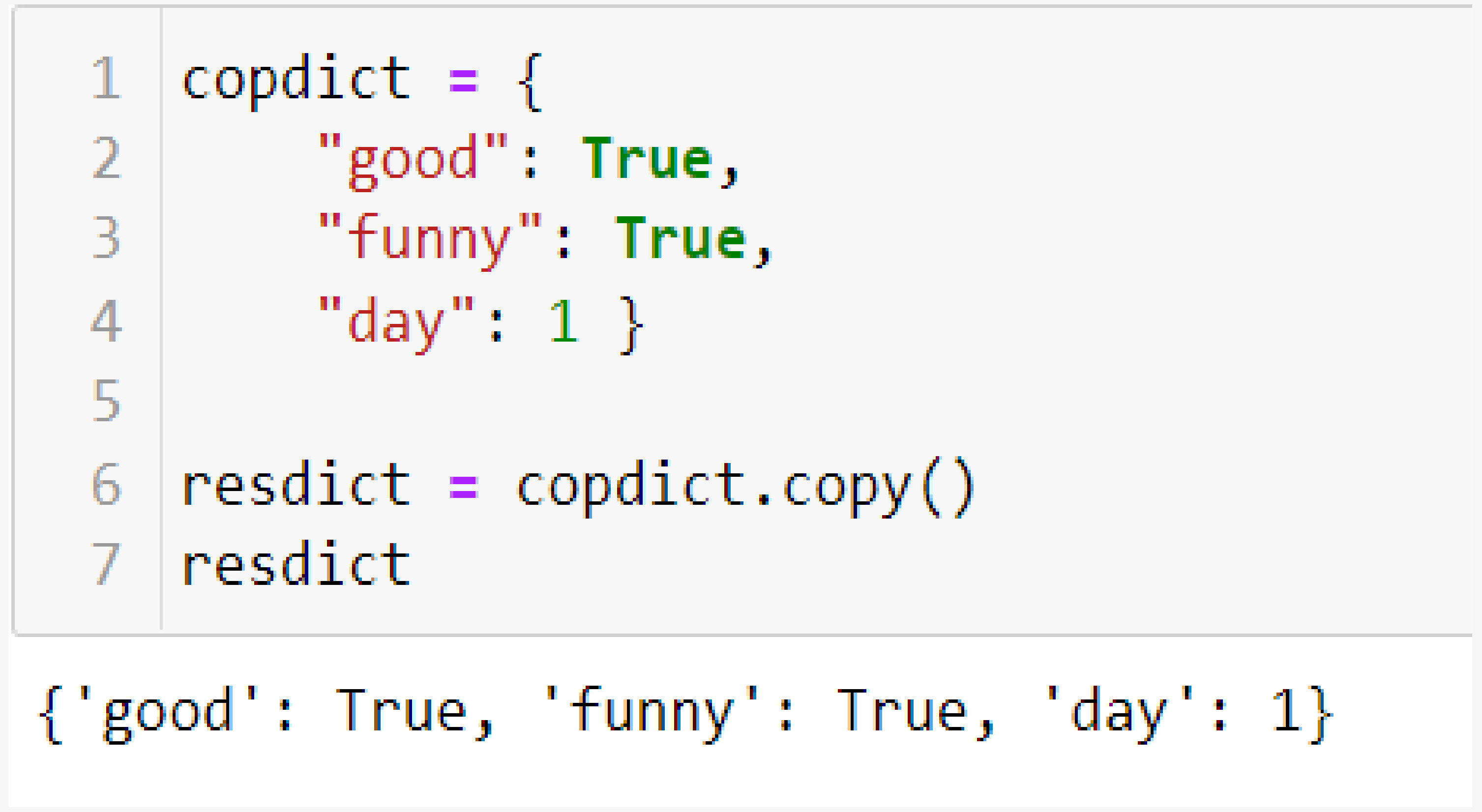


```python
loopdict = {"website":"facebook","color": "blue","country":'USA'}

for item in loopdict:
    print(item)
for item in loopdict.values():
    print(item)
for item in loopdict.keys():
    print(item)
for key, value in loopdict.items():
    print(key, value)
```

```
website
color
country
facebook
blue
USA
website
color
country
website facebook
color blue
country USA
```

## Copy Dictionaries

- There are many ways to make a copy of dictionary, one way is to use the built-Dictionary in method copy()



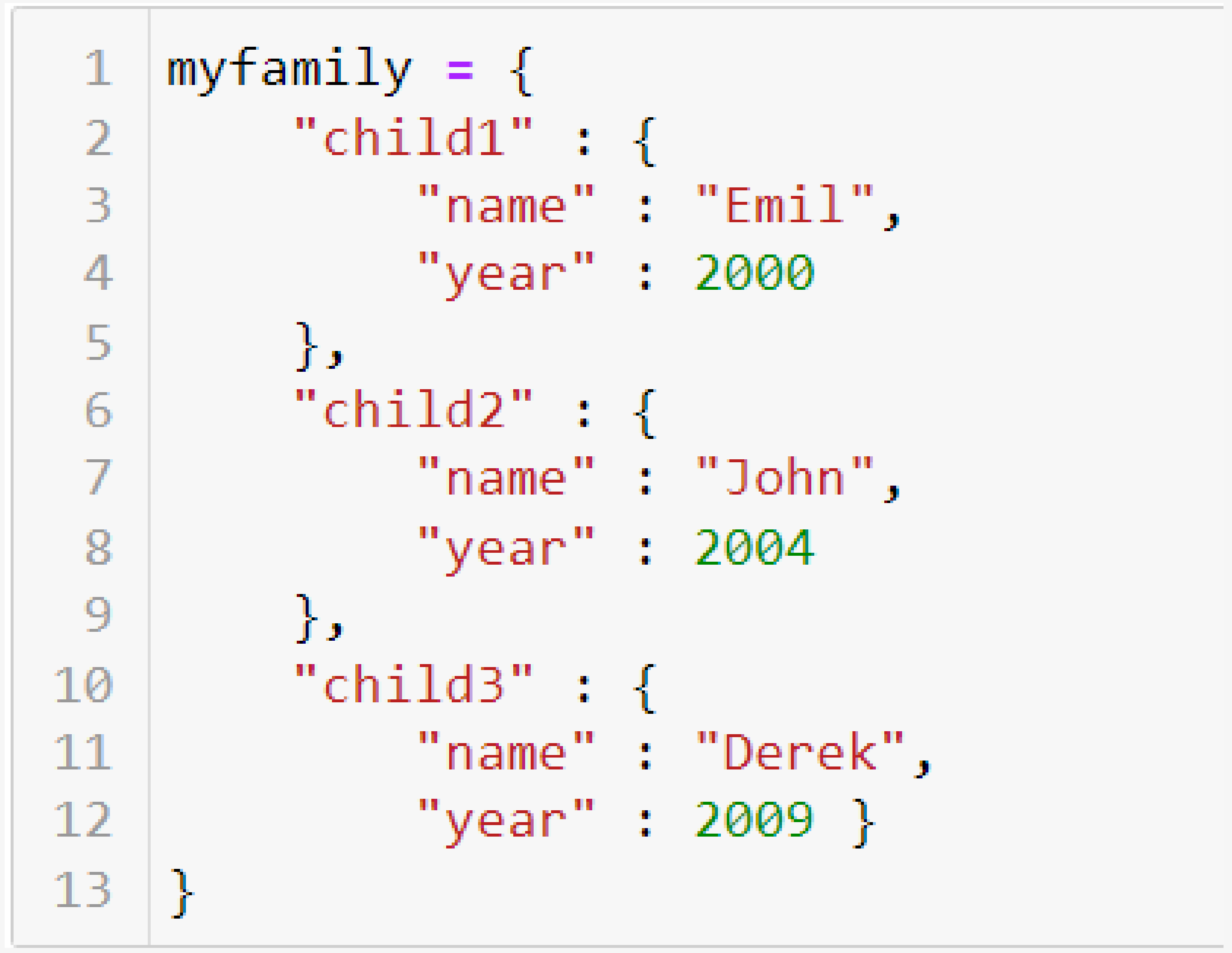
```
copdict = {
    "good": True,
    "funny": True,
    "day": 1 }

resdict = copdict.copy()
resdict
```

```
{'good': True, 'funny': True, 'day': 1}
```

The Learning Hub

## Nested Dictionaries

- **Nested Dictionaries:** when the dictionary contains dictionaries



```
myfamily = {
    "child1" : {
        "name" : "Emil",
        "year" : 2000
    },
    "child2" : {
        "name" : "John",
        "year" : 2004
    },
    "child3" : {
        "name" : "Derek",
        "year" : 2009 }
}
```

The Learning Hub

# Dictionary Methods

| Method | Description |
|---|---|
| clear() | Removes all the elements from the dictionary |
| copy() | Returns a copy of the dictionary |
| fromkeys() | Returns a dictionary with the specified keys and value |
| get() | Returns the value of the specified key |
| items() | Returns a list containing a tuple for each key value pair |
| keys() | Returns a list containing the dictionary's keys |
| pop() | Removes the element with the specified key |
| popitem() | Removes the last inserted key-value pair |

The Learning Hub

## References

- Python Tutorial (w3schools.com)
- The Python Tutorial — Python 3.10.7 documentation
- Python Tutorial for Beginners: Learn Programming Basics [PDF] (guru99.com)

THANK YOU