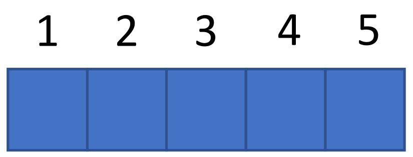
On list, tuples, string and dictionaries

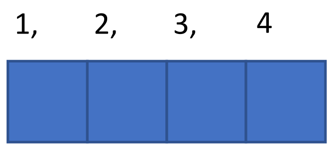
support

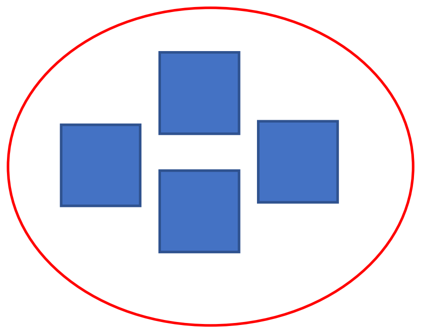
Python collection

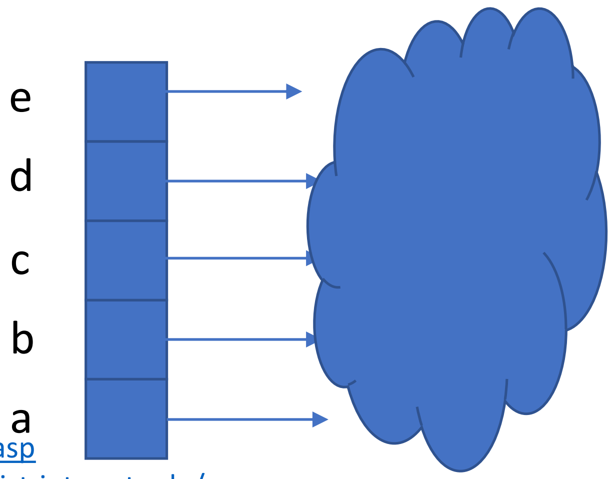
There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.



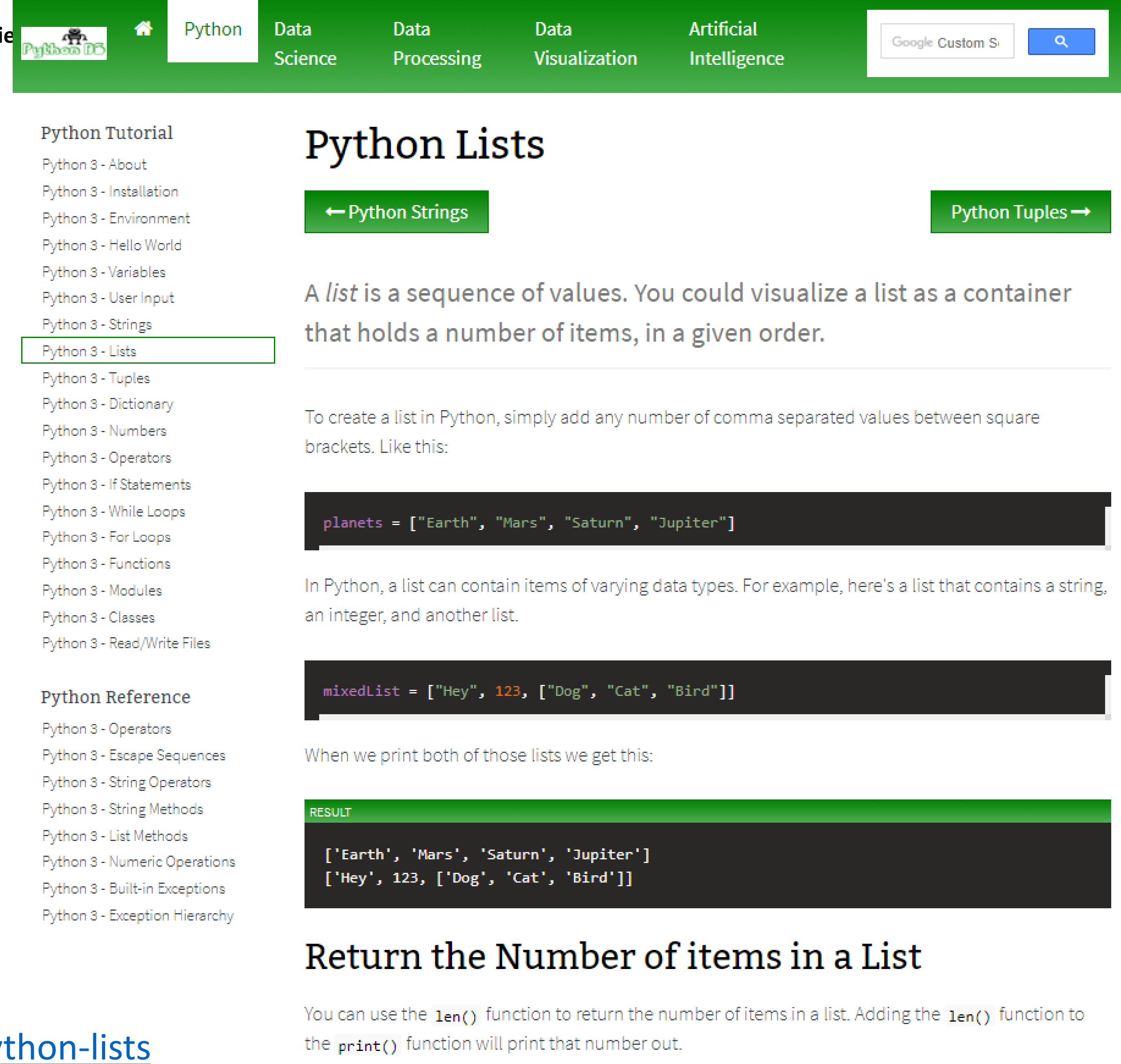






https://www.w3schools.com/python/python_lists.asp
https://www.geeksforgeeks.org/python-convert-a-list-into-a-tuple/

List



Python Tutorial

Python 3 - About
Python 3 - Installation
Python 3 - Environment
Python 3 - Hello World
Python 3 - Variables
Python 3 - User Input
Python 3 - Strings
Python 3 - Lists
Python 3 - Tuples
Python 3 - Dictionary
Python 3 - Numbers
Python 3 - Operators
Python 3 - If Statements
Python 3 - While Loops
Python 3 - For Loops
Python 3 - Functions
Python 3 - Modules
Python 3 - Classes
Python 3 - Read/Write Files

Python Reference

Python 3 - Operators
Python 3 - Escape Sequences
Python 3 - String Operators
Python 3 - String Methods
Python 3 - List Methods
Python 3 - Numeric Operations
Python 3 - Built-in Exceptions
Python 3 - Exception Hierarchy

Python Lists

←Python Strings   Python Tuples→

A *list* is a sequence of values. You could visualize a list as a container that holds a number of items, in a given order.

To create a list in Python, simply add any number of comma separated values between square brackets. Like this:

```
planets = ["Earth", "Mars", "Saturn", "Jupiter"]
```

In Python, a list can contain items of varying data types. For example, here's a list that contains a string, an integer, and another list.

```
mixedList = ["Hey", 123, ["Dog", "Cat", "Bird"]]
```

When we print both of those lists we get this:

RESULT

```
['Earth', 'Mars', 'Saturn', 'Jupiter']
['Hey', 123, ['Dog', 'Cat', 'Bird']]
```

Return the Number of items in a List

You can use the `len()` function to return the number of items in a list. Adding the `len()` function to the `print()` function will print that number out.

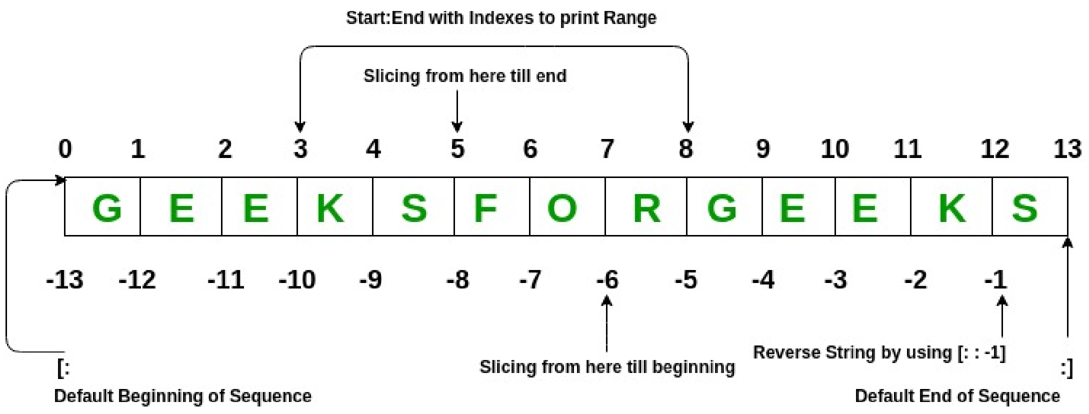
http://python-ds.com/python-lists
https://realpython.com/python-lists-tuples/

list

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

```
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

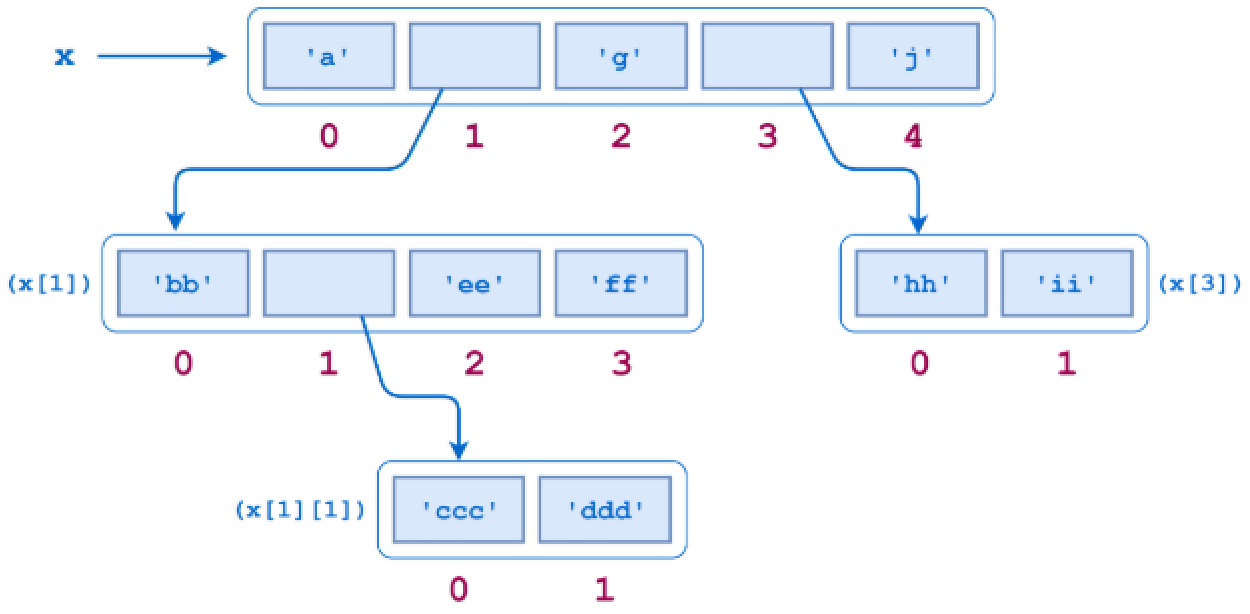

Start:End with Indexes to print Range
Slicing from here till end
0 1 2 3 4 5 6 7 8 9 10 11 12 13
G E E K S F O R G E E K S
-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
[:
Default Beginning of Sequence
Slicing from here till beginning
Reverse String by using [: : -1]
:]
Default End of Sequence

https://www.tutorialspoint.com/python/python_lists.htm

List can have different data

```
Python                                                                    >>>

>>> x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']
>>> x
['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']
```

The object structure that x references is diagrammed below:

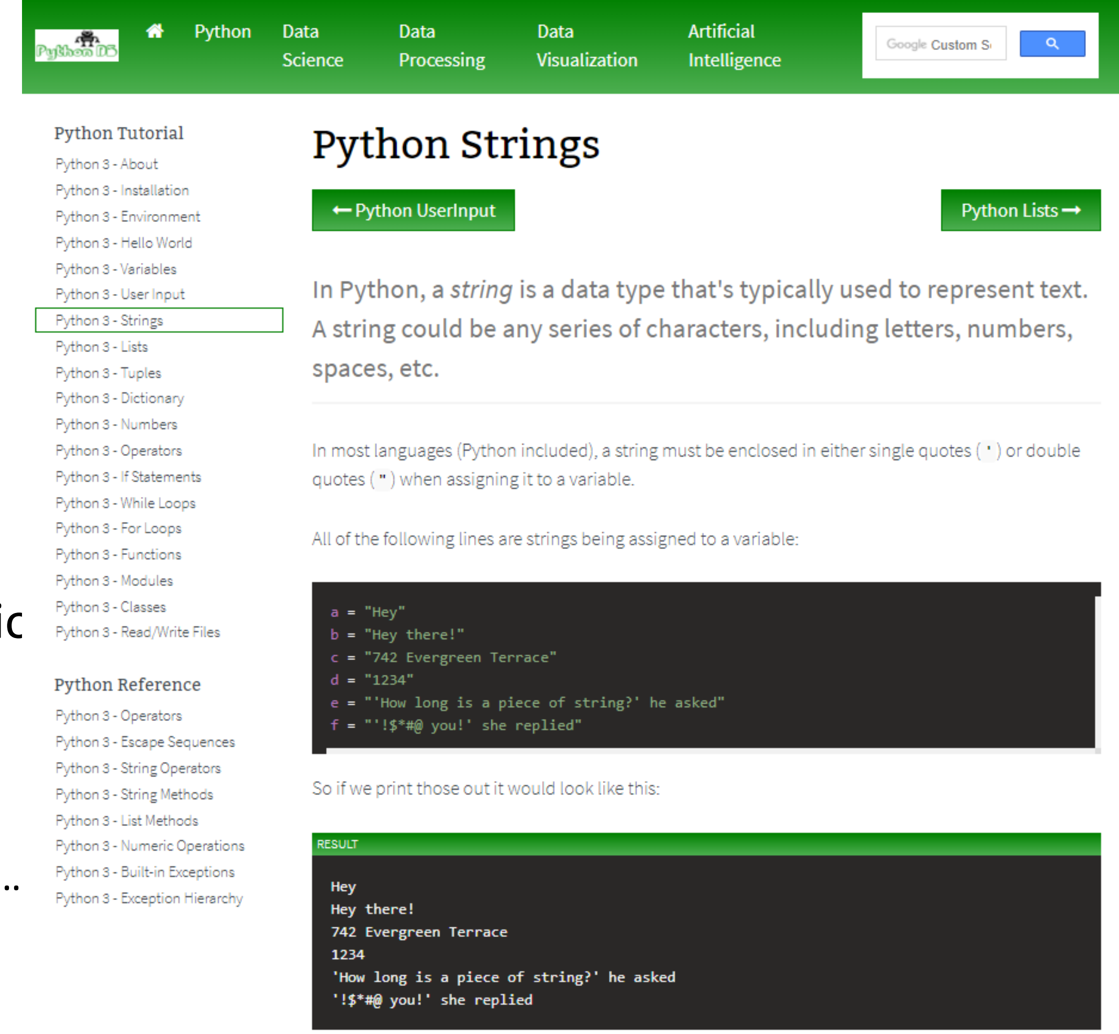


A Nested List

https://realpython.com/python-lists-tuples/

String

- Are list of characters
  - What you know on list is "mostly" valid
- Have extra features & functic
  - Ease handling text
    - center, capitalize,…
  - Manipulate & search
    - Add, find, remove part,…



Python Strings

In Python, a *string* is a data type that's typically used to represent text. A string could be any series of characters, including letters, numbers, spaces, etc.

In most languages (Python included), a string must be enclosed in either single quotes (') or double quotes (") when assigning it to a variable.

All of the following lines are strings being assigned to a variable:

```
a = "Hey"
b = "Hey there!"
c = "742 Evergreen Terrace"
d = "1234"
e = "'How long is a piece of string?' he asked"
f = "'!$*#@ you!' she replied"
```

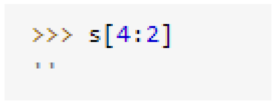
So if we print those out it would look like this:

```
Hey
Hey there!
742 Evergreen Terrace
1234
'How long is a piece of string?' he asked
'!$*#@ you!' she replied
```

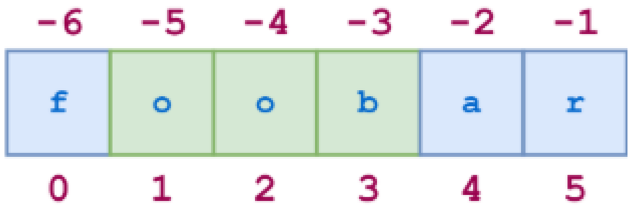
http://python-ds.com/python-strings
https://realpython.com/python-strings/

String are similar to list

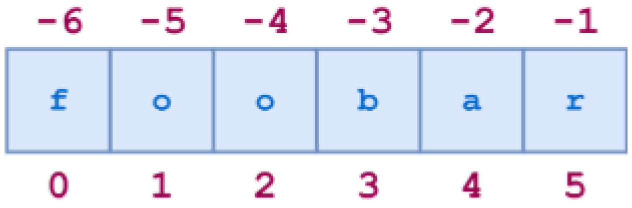
G	E	E	K	S	F	O	R	G	E	E	K	S
0	1	2	3	4	5	6	7	8	9	10	11	12
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	



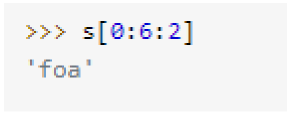
```
>>> s[4:2]
''
```



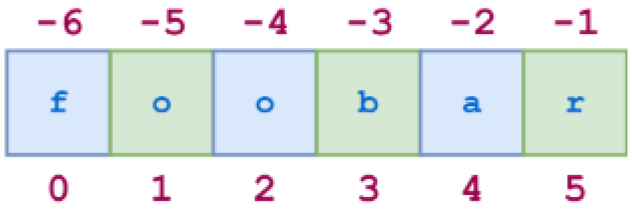
String Slicing with Positive and Negative Indices



Positive and Negative String Indices



```
>>> s[0:6:2]
'foa'
```

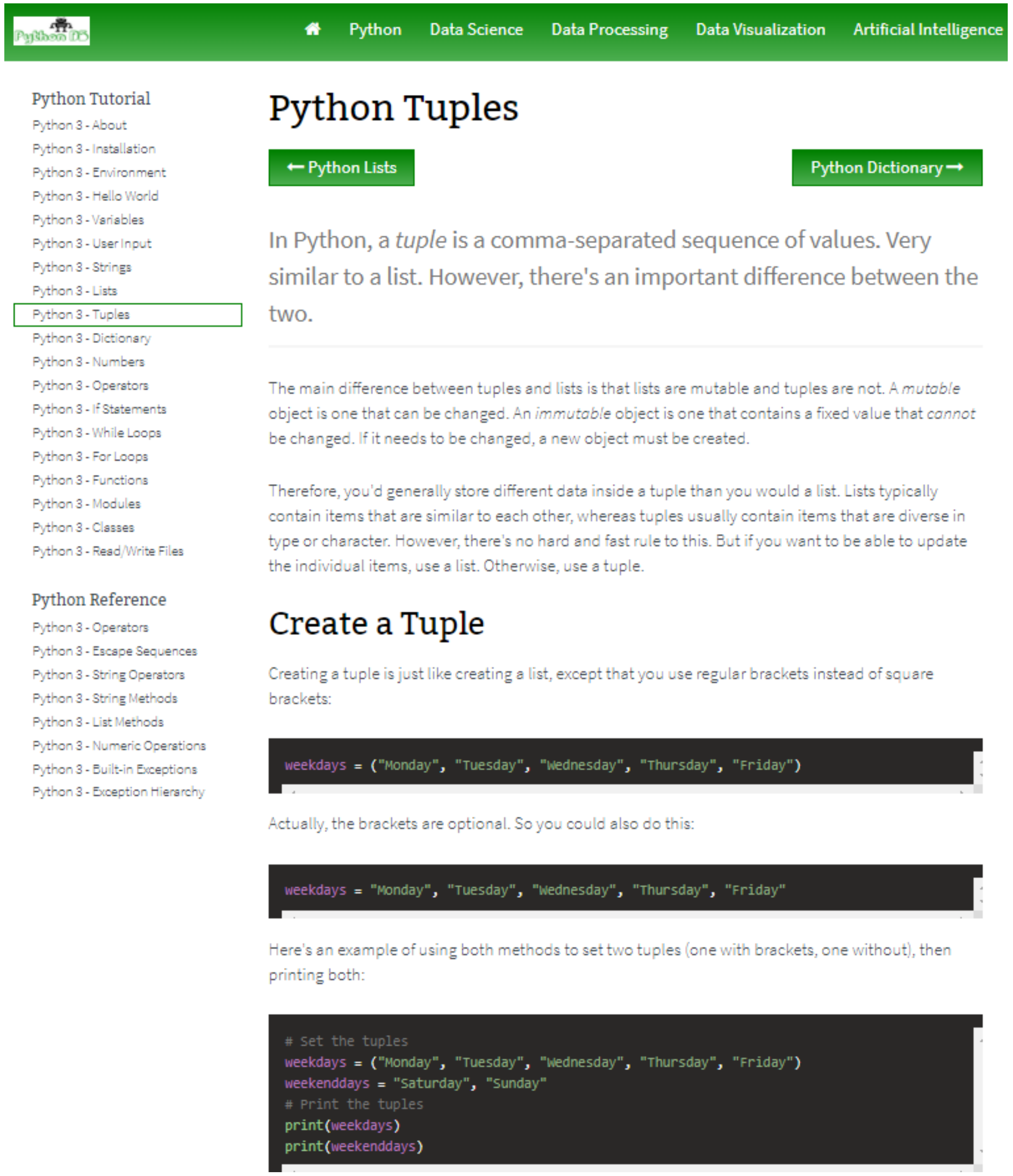


Another String Indexing with Stride

https://realpython.com/python-strings/
https://www.geeksforgeeks.org/python-strings/

Tuples

- Can be seen as static list
- Can access individual positions with index
- …



Python Tuples

In Python, a *tuple* is a comma-separated sequence of values. Very similar to a list. However, there's an important difference between the two.

The main difference between tuples and lists is that lists are mutable and tuples are not. A *mutable* object is one that can be changed. An *immutable* object is one that contains a fixed value that *cannot* be changed. If it needs to be changed, a new object must be created.

Therefore, you'd generally store different data inside a tuple than you would a list. Lists typically contain items that are similar to each other, whereas tuples usually contain items that are diverse in type or character. However, there's no hard and fast rule to this. But if you want to be able to update the individual items, use a list. Otherwise, use a tuple.

Create a Tuple

Creating a tuple is just like creating a list, except that you use regular brackets instead of square brackets:

```
weekdays = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
```

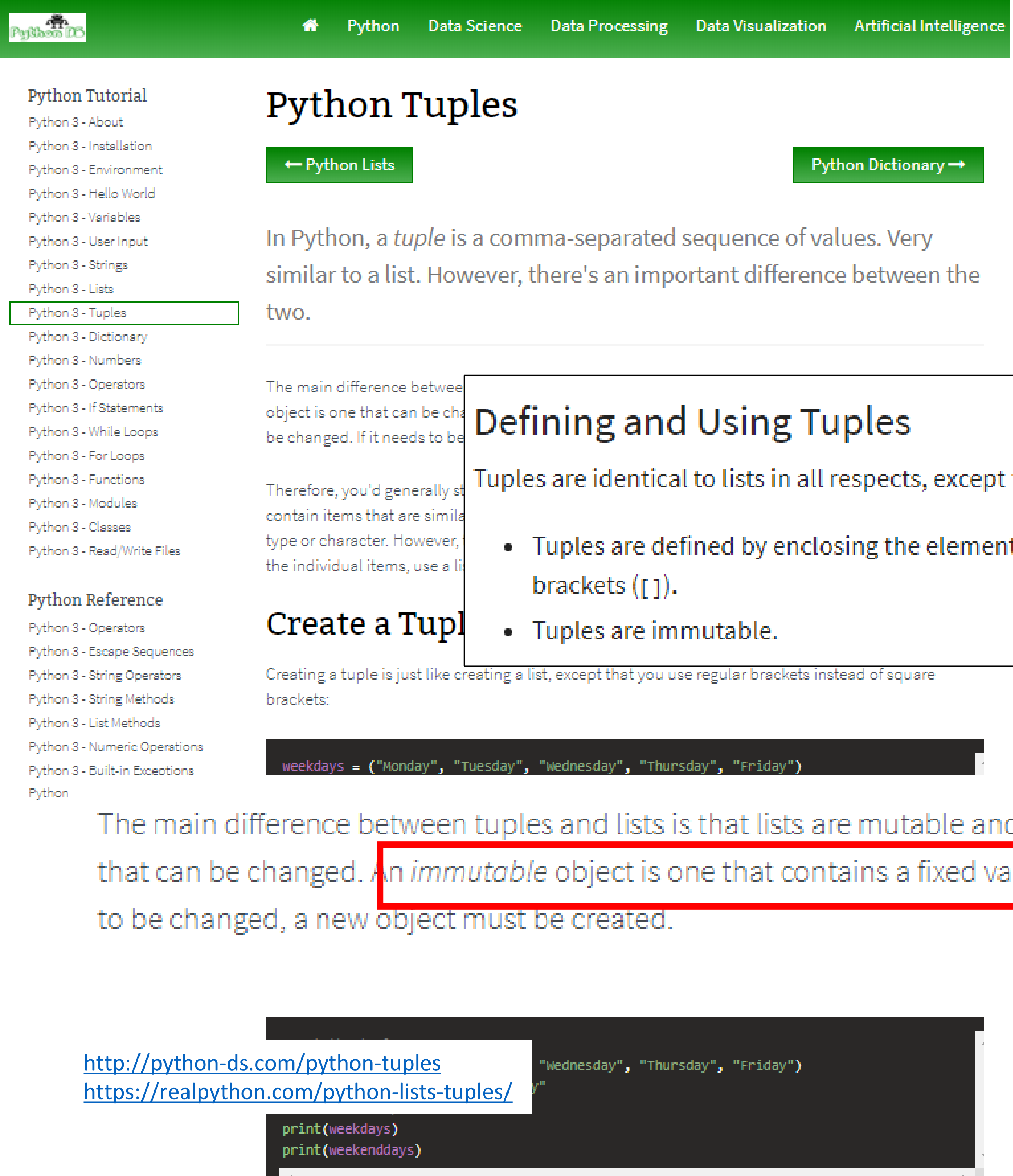
Actually, the brackets are optional. So you could also do this:

```
weekdays = "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"
```

Here's an example of using both methods to set two tuples (one with brackets, one without), then printing both:

```
# Set the tuples
weekdays = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
weekenddays = "Saturday", "Sunday"
# Print the tuples
print(weekdays)
print(weekenddays)
```

http://python-ds.com/python-tuples
https://realpython.com/python-lists-tuples/



Python Tuples

In Python, a *tuple* is a comma-separated sequence of values. Very similar to a list. However, there's an important difference between the two.

The main difference betwee
object is one that can be ch
be changed. If it needs to be

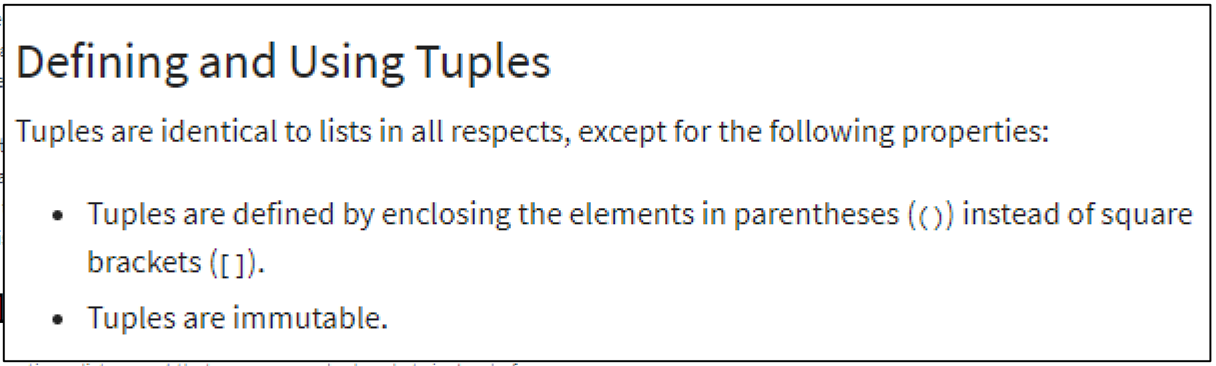
Therefore, you'd generally s
contain items that are simila
type or character. However,
the individual items, use a li

Create a Tupl

Creating a tuple is just like creating a list, except that you use regular brackets instead of square brackets:

```
weekdays = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
```

```
"Wednesday", "Thursday", "Friday")
y"

print(weekdays)
print(weekenddays)
```



Defining and Using Tuples

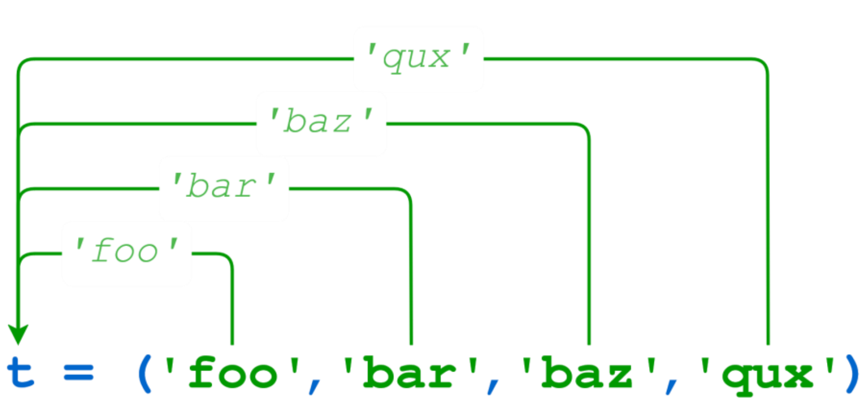
Tuples are identical to lists in all respects, except for the following properties:

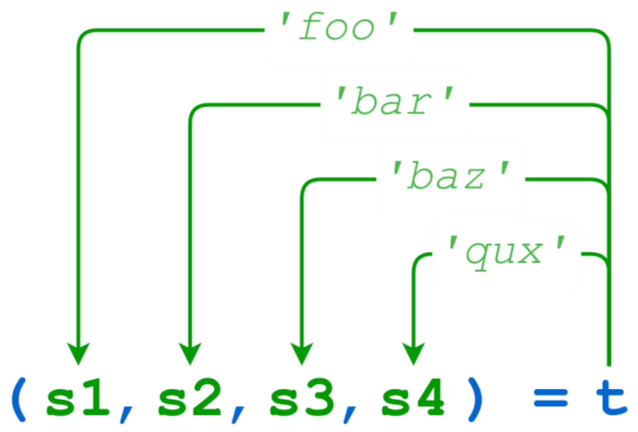
- Tuples are defined by enclosing the elements in parentheses (`()`) instead of square brackets (`[]`).
- Tuples are immutable.

The main difference between tuples and lists is that lists are mutable and tuples are not. A *mutable* object is one that can be changed. An *immutable* object is one that contains a fixed value that *cannot* be changed. If it needs to be changed, a new object must be created.

http://python-ds.com/python-tuples
https://realpython.com/python-lists-tuples/

tuples





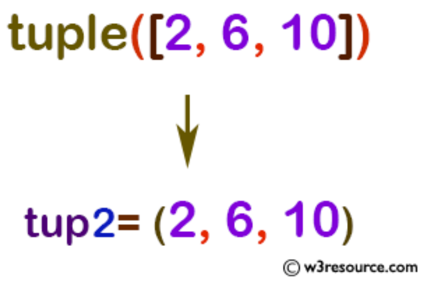
```
>>> t = ('foo', 'bar', 'baz', 'qux')
```

https://realpython.com/python-lists-tuples/

converting

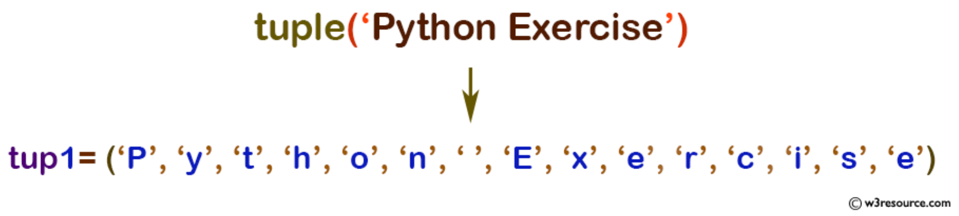
```
>>> t=((1,'a'), (2,'b'))
>>> dict(t)
{1: 'a', 2: 'b'}
```

```
Input : [1, 2, 3, 4]
Output : (1, 2, 3, 4)

Input : ['a', 'b', 'c']
Output : ('a', 'b', 'c')
```



```
tuple([2, 6, 10])
        ↓
tup2= (2, 6, 10)
```

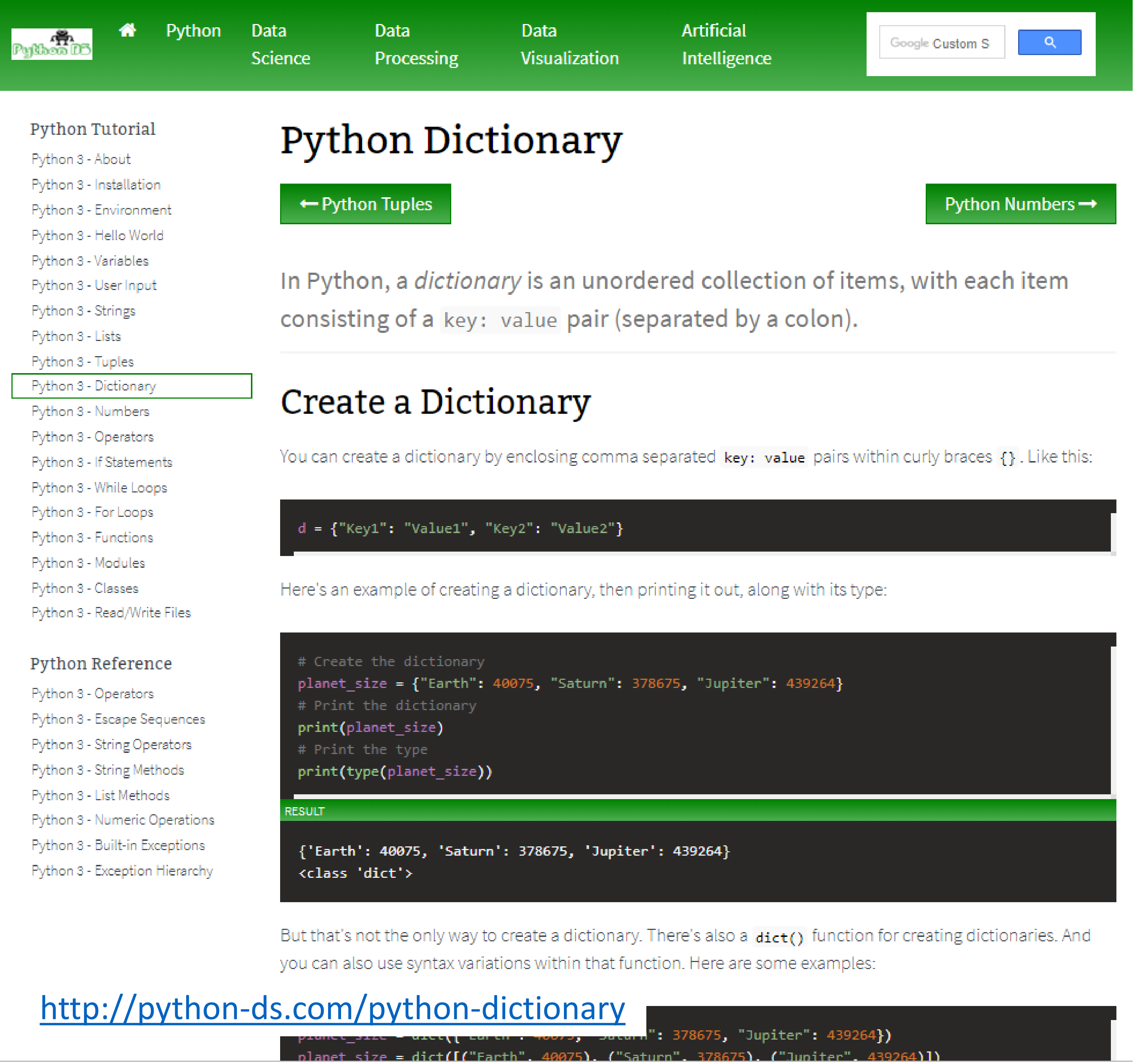
```
listNumbers = [6,3,7,4]
x = tuple(listNumbers)
print(x)
```



```
tuple('Python Exercise')
        ↓
tup1= ('P', 'y', 't', 'h', 'o', 'n', ' ', 'E', 'x', 'e', 'r', 'c', 'i', 's', 'e')
```

```
x = (4,5)
listNumbers = list(x)
print(listNumbers)
```

https://pythonspot.com/convert-tuple/
https://www.geeksforgeeks.org/python-convert-list-tuples-dictionary/
https://www.tutorialspoint.com/How-I-can-convert-a-Python-Tuple-into-Dictionary
https://www.w3resource.com/python/built-in-function/tuple.php



Python Tutorial

Python 3 - About
Python 3 - Installation
Python 3 - Environment
Python 3 - Hello World
Python 3 - Variables
Python 3 - User Input
Python 3 - Strings
Python 3 - Lists
Python 3 - Tuples
Python 3 - Dictionary
Python 3 - Numbers
Python 3 - Operators
Python 3 - If Statements
Python 3 - While Loops
Python 3 - For Loops
Python 3 - Functions
Python 3 - Modules
Python 3 - Classes
Python 3 - Read/Write Files

Python Reference

Python 3 - Operators
Python 3 - Escape Sequences
Python 3 - String Operators
Python 3 - String Methods
Python 3 - List Methods
Python 3 - Numeric Operations
Python 3 - Built-in Exceptions
Python 3 - Exception Hierarchy

Python Dictionary

⟵Python Tuples   Python Numbers⟶

In Python, a *dictionary* is an unordered collection of items, with each item consisting of a `key: value` pair (separated by a colon).

Create a Dictionary

You can create a dictionary by enclosing comma separated `key: value` pairs within curly braces `{}`. Like this:

```
d = {"Key1": "Value1", "Key2": "Value2"}
```

Here's an example of creating a dictionary, then printing it out, along with its type:

```
# Create the dictionary
planet_size = {"Earth": 40075, "Saturn": 378675, "Jupiter": 439264}
# Print the dictionary
print(planet_size)
# Print the type
print(type(planet_size))
```

RESULT

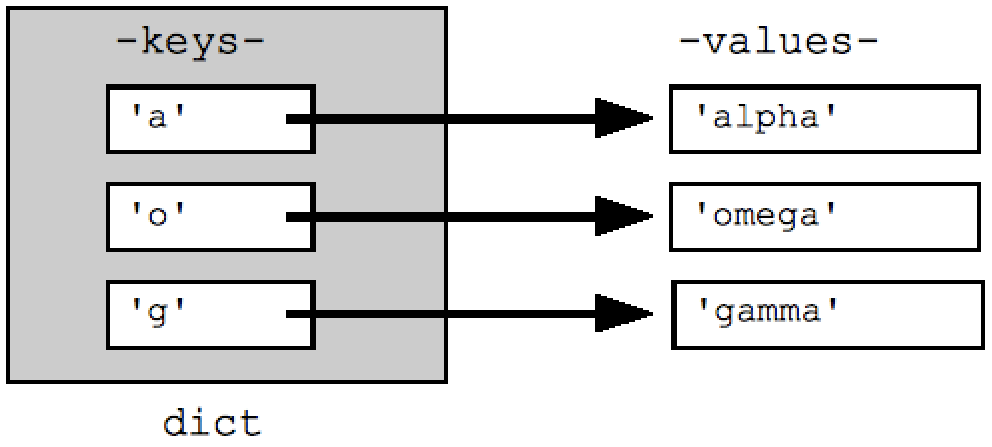
```
{'Earth': 40075, 'Saturn': 378675, 'Jupiter': 439264}
<class 'dict'>
```

But that's not the only way to create a dictionary. There's also a `dict()` function for creating dictionaries. And you can also use syntax variations within that function. Here are some examples:

```
": 378675, "Jupiter": 439264})
planet_size = dict([("Earth", 40075), ("Saturn", 378675), ("Jupiter", 439264)])
```

http://python-ds.com/python-dictionary

dictionaries

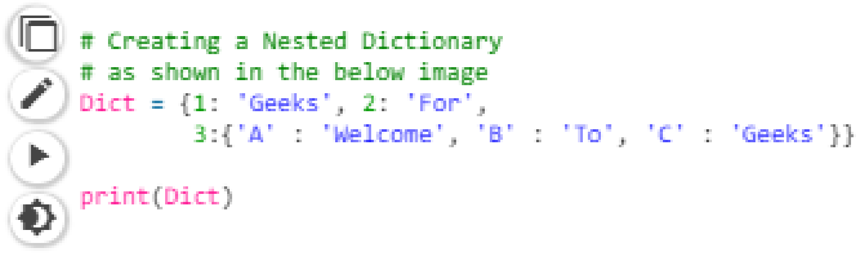


```
webstersDict = {'person': 'a human being',
                'marathon': 'a running race that is about 26 miles',
                'resist': 'to remain strong against the force',
                'run': 'to move with haste; act quickly'}
```

🟥 dictionary key

🟦 dictionary value

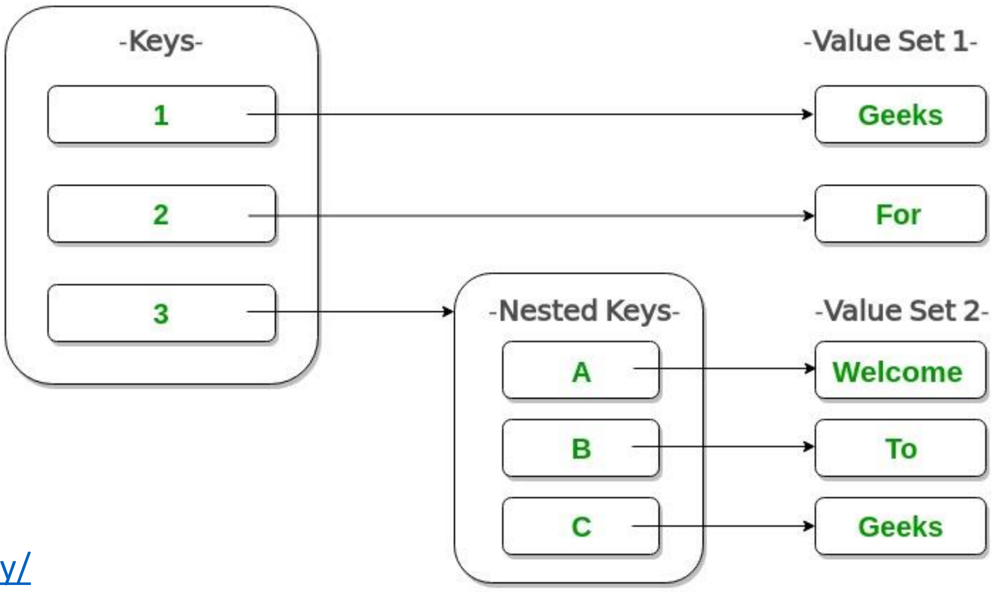
dictionaries: mixing…



```
# Creating a Nested Dictionary
# as shown in the below image
Dict = {1: 'Geeks', 2: 'For',
        3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'Geeks'}}

print(Dict)
```

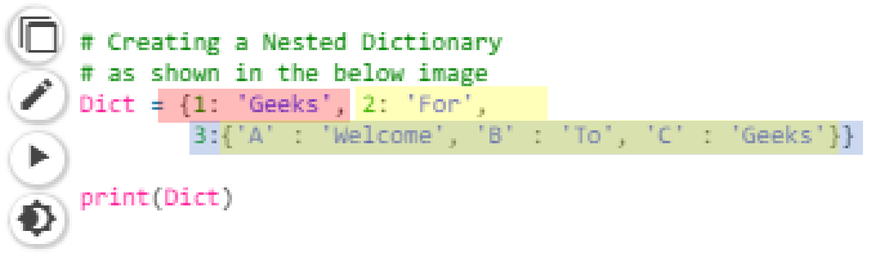
Output:

```
{1: 'Geeks', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
```



https://www.geeksforgeeks.org/python-dictionary/

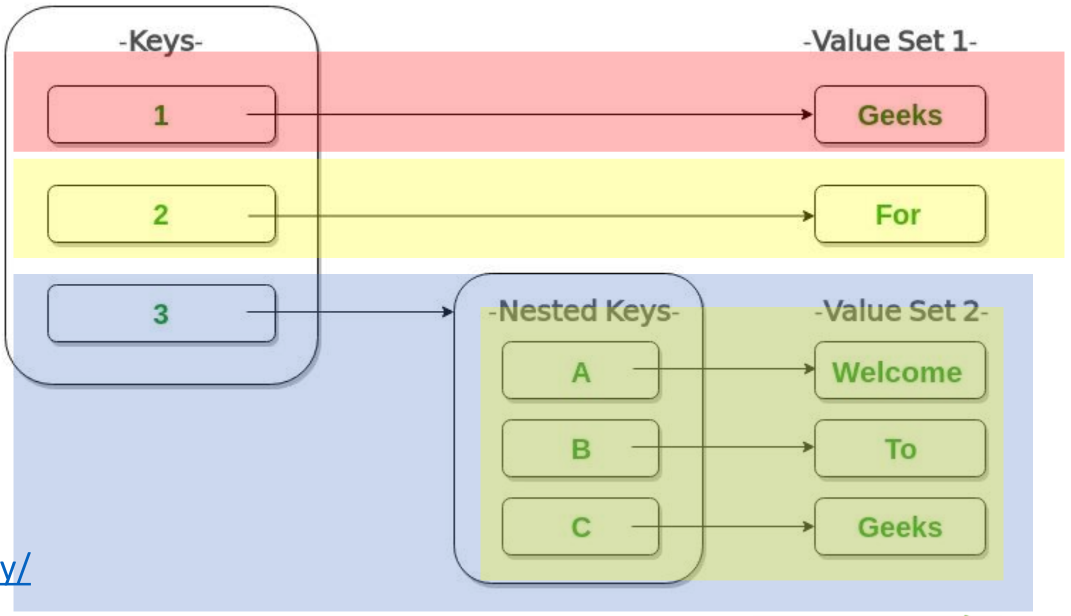
dictionaries: mixing…



```
# Creating a Nested Dictionary
# as shown in the below image
Dict = {1: 'Geeks', 2: 'For',
        3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'Geeks'}}

print(Dict)
```

Output:

```
{1: 'Geeks', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
```



https://www.geeksforgeeks.org/python-dictionary/

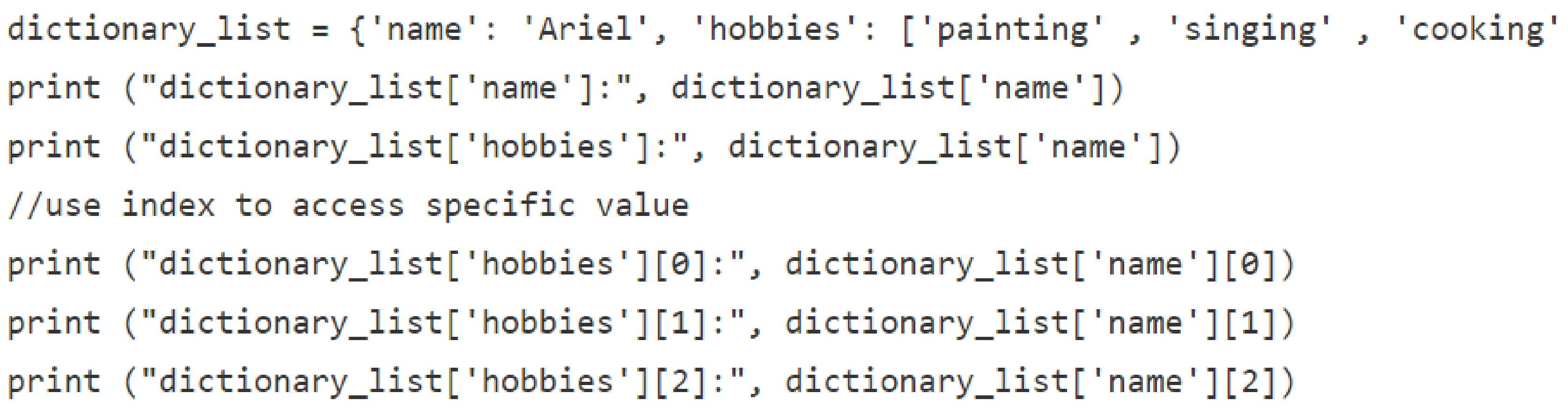
dictionaries: mixing...

```
dictionary_list = {'name': 'Ariel', 'hobbies': ['painting' , 'singing' , 'cooking'
print ("dictionary_list['name']:", dictionary_list['name'])
print ("dictionary_list['hobbies']:", dictionary_list['name'])
//use index to access specific value
print ("dictionary_list['hobbies'][0]:", dictionary_list['name'][0])
print ("dictionary_list['hobbies'][1]:", dictionary_list['name'][1])
print ("dictionary_list['hobbies'][2]:", dictionary_list['name'][2])
```

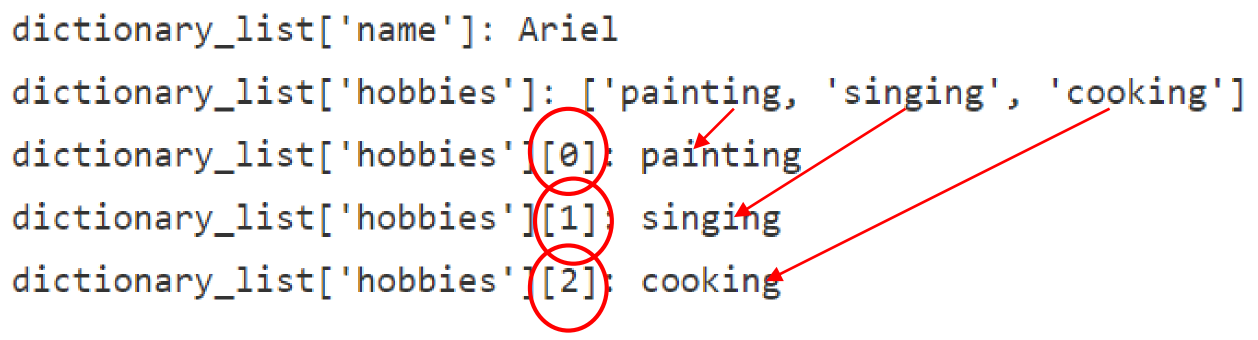
```
dictionary_list['name']: Ariel
dictionary_list['hobbies']: ['painting, 'singing', 'cooking']
dictionary_list['hobbies'][0]: painting
dictionary_list['hobbies'][1]: singing
dictionary_list['hobbies'][2]: cooking
```

https://www.gangboard.com/blog/python-dictionary/

dictionaries: mixing…



```
dictionary_list = {'name': 'Ariel', 'hobbies': ['painting' , 'singing' , 'cooking'
print ("dictionary_list['name']:", dictionary_list['name'])
print ("dictionary_list['hobbies']:", dictionary_list['name'])
//use index to access specific value
print ("dictionary_list['hobbies'][0]:", dictionary_list['name'][0])
print ("dictionary_list['hobbies'][1]:", dictionary_list['name'][1])
print ("dictionary_list['hobbies'][2]:", dictionary_list['name'][2])
```

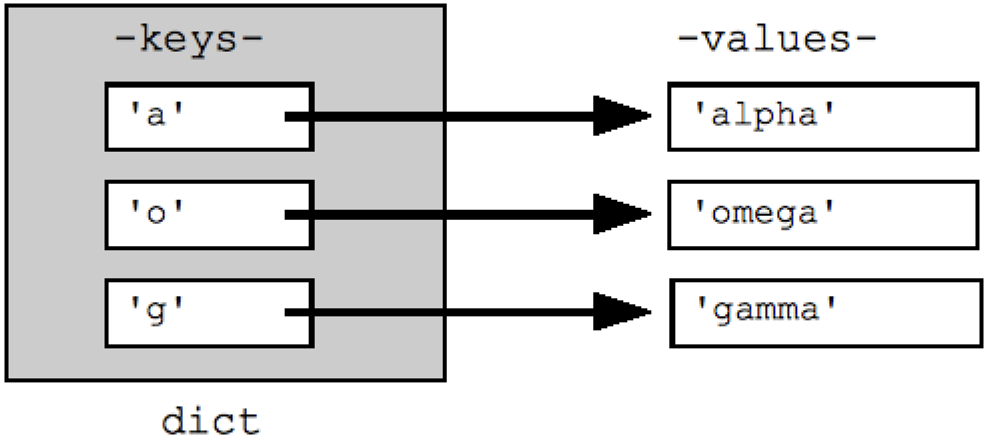


```
dictionary_list['name']: Ariel
dictionary_list['hobbies']: ['painting, 'singing', 'cooking']
dictionary_list['hobbies'][0]: painting
dictionary_list['hobbies'][1]: singing
dictionary_list['hobbies'][2]: cooking
```

https://www.gangboard.com/blog/python-dictionary/

universidade de aveiro

dictionaries_: keys, values & items



```
d.keys() → [ ‘a’, ‘o’, ‘g’ ]
d.values() →  [ ‘alpha’, ‘omega’, ‘gamma’ ]
d.items() →  [ ‘a’:’alpha” , ‘o’:’omega’, ‘g’:’gamma’ ]
d.count() →   3
‘a’ in d  → True
‘b’ in d → False
```

Dictionaries: listing, ...

Using the key (implicit)

```
>>> for key in a_dict:
...     print(key, '->', a_dict[key])
...
color -> blue
fruit -> apple
pet -> dog
```

Using the key (explicit)

```
>>> for key in a_dict.keys():
...     print(key, '->', a_dict[key])
...
color -> blue
fruit -> apple
pet -> dog
```

Using the items → list of key,values

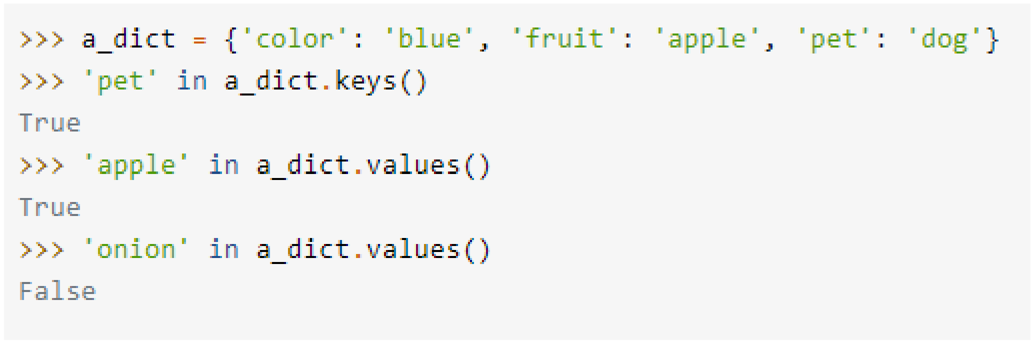
```
>>> for item in a_dict.items():
...     print(item)
...
('color', 'blue')
('fruit', 'apple')
('pet', 'dog')
```

```
>>> for key, value in a_dict.items():
...     print(key, '->', value)
...
color -> blue
fruit -> apple
pet -> dog
```

Using the values (explicit)

```
>>> for value in a_dict.values():
...     print(value)
...
blue
apple
dog
```

dictionaries: in?



```
>>> a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
>>> 'pet' in a_dict.keys()
True
>>> 'apple' in a_dict.values()
True
>>> 'onion' in a_dict.values()
False
```

https://realpython.com/iterate-through-dictionary-python/

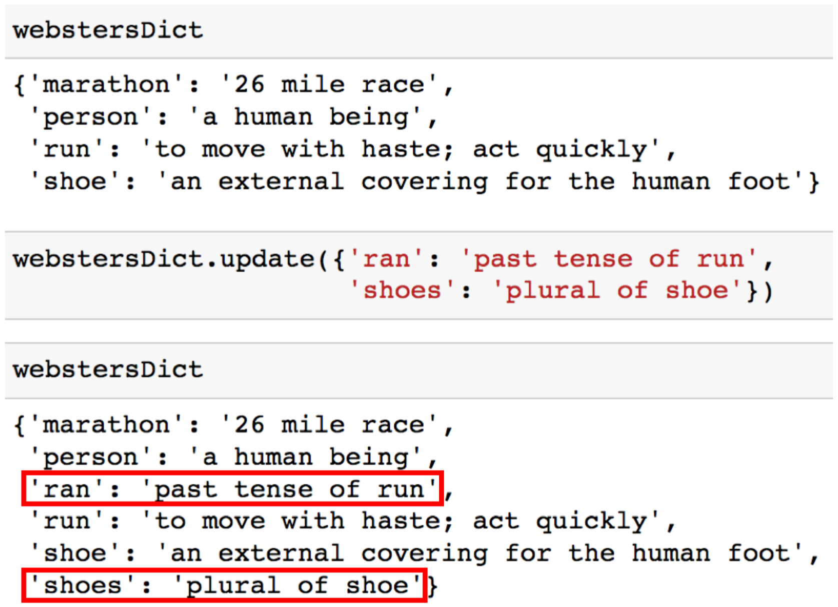
dictionaries: delete & update

```
>>> prices = {'apple': 0.40, 'orange': 0.35, 'banana': 0.25}
>>> for k, v in prices.items():
...     prices[k] = round(v * 0.9, 2)  # Apply a 10% discount
...
>>> prices
{'apple': 0.36, 'orange': 0.32, 'banana': 0.23}
```

```
>>> prices = {'apple': 0.40, 'orange': 0.35, 'banana': 0.25}
>>> for key in list(prices.keys()):  # Use a list instead of a view
...     if key == 'orange':
...         del prices[key]  # Delete a key from prices
...
>>> prices
{'apple': 0.4, 'banana': 0.25}
```

https://realpython.com/iterate-through-dictionary-python/

Dictionaries misc



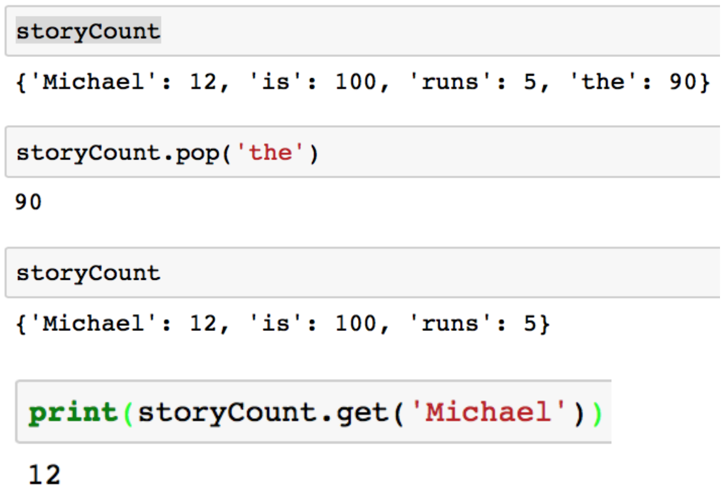
```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot'}
```

```
webstersDict.update({'ran': 'past tense of run',
                     'shoes': 'plural of shoe'})
```

```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'ran': 'past tense of run',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot',
 'shoes': 'plural of shoe'}
```



```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5, 'the': 90}
```

```
storyCount.pop('the')
```

```
90
```

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5}
```

```
print(storyCount.get('Michael'))
```

```
12
```

https://medium.com/@GalarnykMichael/python-basics-10-dictionaries-and-dictionary-methods-4e9efa70f5b9

Dictionaries misc

```
student_dictionary  = {'name' : 'Lisa', 'age' : 6, 'grade' : '1' }
student_dictionary.pop('grade')
print (student_dictionary)
```

Output:

```
{'name': 'Lisa', 'age': 6 }
```

```
dict = {1: "one", 2: "three"}
dict_update = {2: "two"}
#value of key 2 is updated
dict.update(dict_update)
print(dict)
```

Output:

```
{1: 'one', 2: 'two'}
```

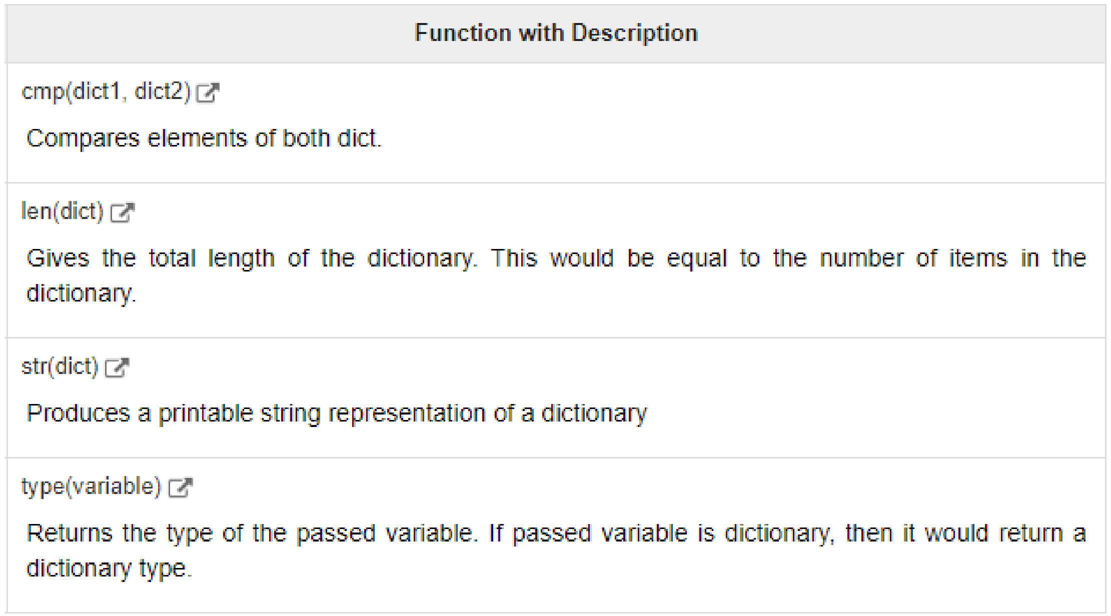
```
dog = { "breed": "labrador", "color": "dusty white", "sex": "female" }
x = dog.values()
print(x)
```

Output:

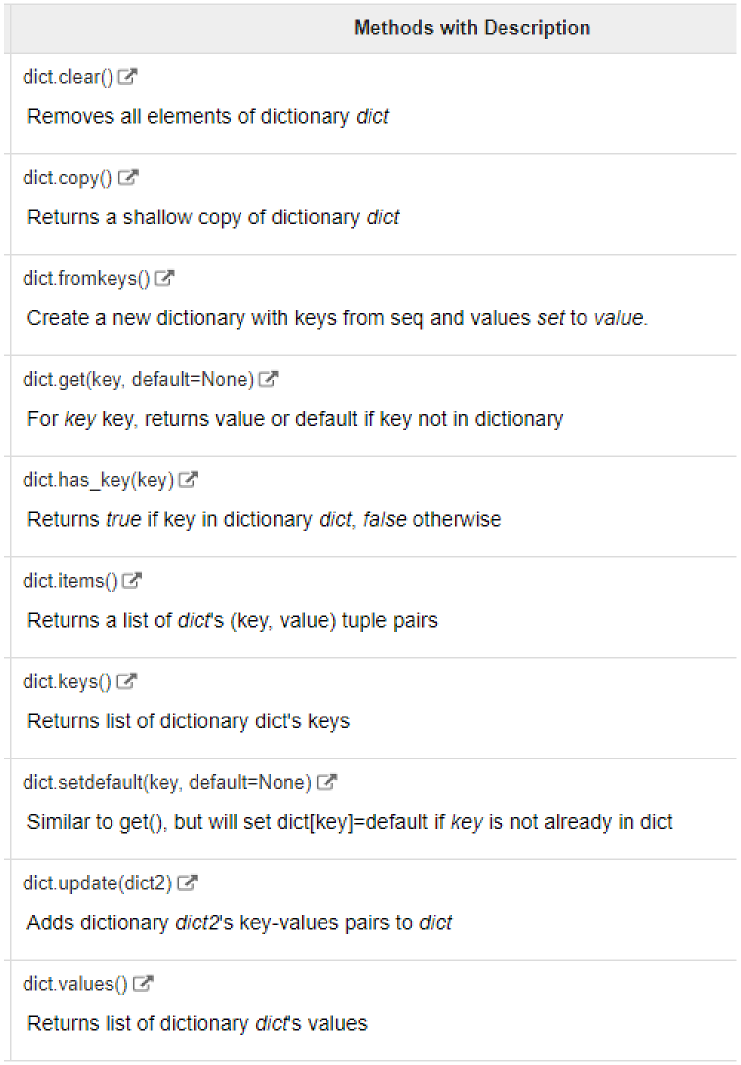
```
dict_values(['labrador', 'dusty white', 'female'])
```

https://www.gangboard.com/blog/python-dictionary/

Support on list, tuples, string and dictionaries



Function with Description
cmp(dict1, dict2) Compares elements of both dict.
len(dict) Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
str(dict) Produces a printable string representation of a dictionary
type(variable) Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.



Methods with Description
dict.clear() Removes all elements of dictionary *dict*
dict.copy() Returns a shallow copy of dictionary *dict*
dict.fromkeys() Create a new dictionary with keys from seq and values *set* to *value*.
dict.get(key, default=None) For *key* key, returns value or default if key not in dictionary
dict.has_key(key) Returns *true* if key in dictionary *dict*, *false* otherwise
dict.items() Returns a list of *dict*'s (key, value) tuple pairs
dict.keys() Returns list of dictionary dict's keys
dict.setdefault(key, default=None) Similar to get(), but will set dict[key]=default if *key* is not already in dict
dict.update(dict2) Adds dictionary *dict2*'s key-values pairs to *dict*
dict.values() Returns list of dictionary *dict*'s values

https://www.tutorialspoint.com/python/python_dictionary.htm

Dictionary: when key does not exist

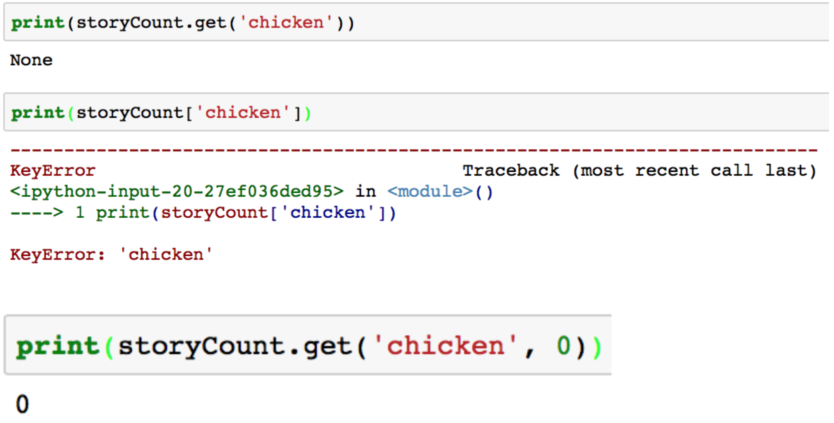
- Getting a non existent key provokes error
  - KeyError

```
country_dict = {'India' : 'IN', 'Australia' : 'AU', 'Brazil' : 'BR'}
print(country_dict['Australia'])
print(country_dict['Canada']) # This will return error
```

- Solution
  - Handle error
  - **Default value (the second parameter)**

https://www.tutorialspoint.com/handling-missing-keys-in-python-dictionaries
https://www.geeksforgeeks.org/handling-missing-keys-python-dictionaries/

Dictionary: default value



```
print(storyCount.get('chicken'))
```

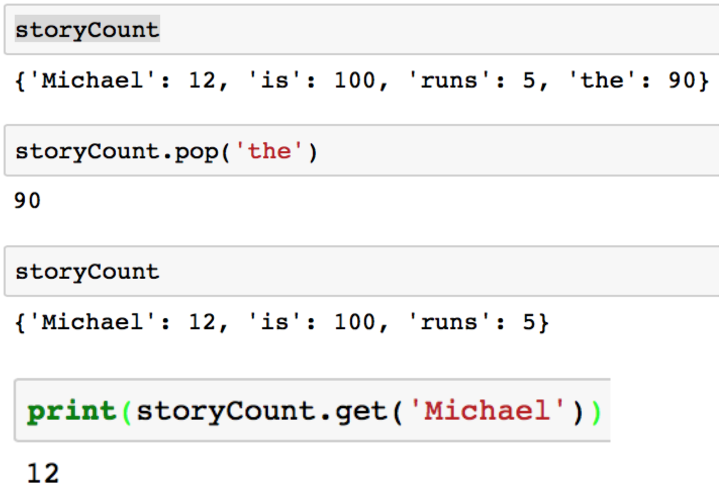
```
None
```

```
print(storyCount['chicken'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-20-27ef036ded95> in <module>()
----> 1 print(storyCount['chicken'])

KeyError: 'chicken'
```

```
print(storyCount.get('chicken', 0))
```

```
0
```



```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5, 'the': 90}
```

```
storyCount.pop('the')
```

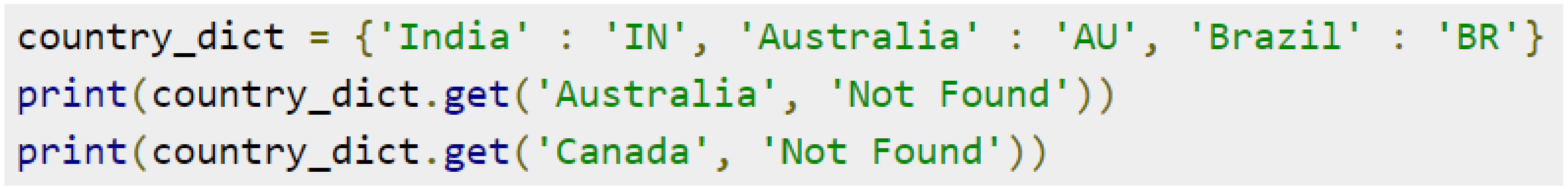
```
90
```

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5}
```

```
print(storyCount.get('Michael'))
```

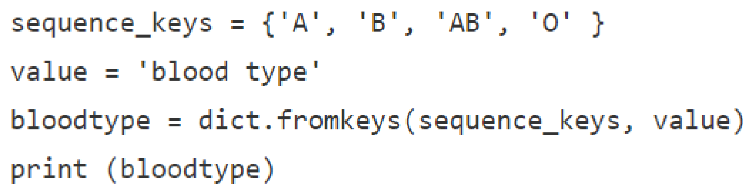
```
12
```



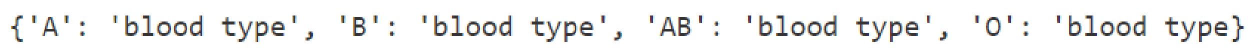
```
country_dict = {'India' : 'IN', 'Australia' : 'AU', 'Brazil' : 'BR'}
print(country_dict.get('Australia', 'Not Found'))
print(country_dict.get('Canada', 'Not Found'))
```

https://www.tutorialspoint.com/handling-missing-keys-in-python-dictionaries
https://www.geeksforgeeks.org/handling-missing-keys-python-dictionaries/

Dictionaries curiosities



```
sequence_keys = {'A', 'B', 'AB', 'O' }
value = 'blood type'
bloodtype = dict.fromkeys(sequence_keys, value)
print (bloodtype)
```



```
{'A': 'blood type', 'B': 'blood type', 'AB': 'blood type', 'O': 'blood type}
```

Similar to

```
bloodtype = {}
Value = ‘blood type’
for k in sequence_keys:
  bloodtype[k]=value
```

https://www.gangboard.com/blog/python-dictionary/

Loops, list and strings

Loops and collections

index

```
colors = ["red", "green", "blue", "purple"]
i = 0
while i < len(colors):
    print(colors[i])
    i += 1
```

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

Iterae, enumerate and zip

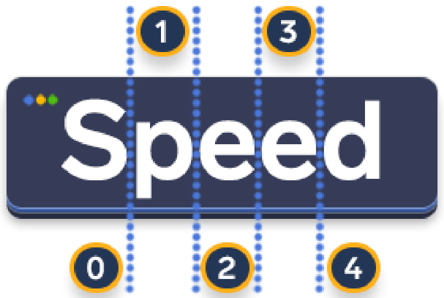
```
colors = ["red", "green", "blue", "purple"]
for color in colors:
    print(color)
```

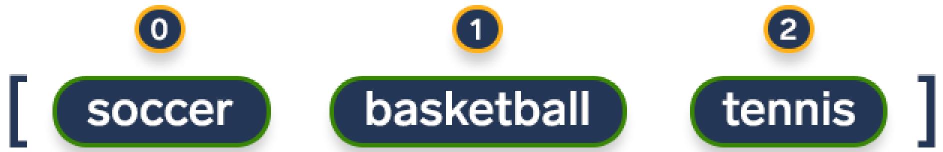
```
presidents = ["Washington", "Adams", "Jefferson", "Madison"
for num, name in enumerate(presidents, start=1):
    print("President {}: {}".format(num, name))
```

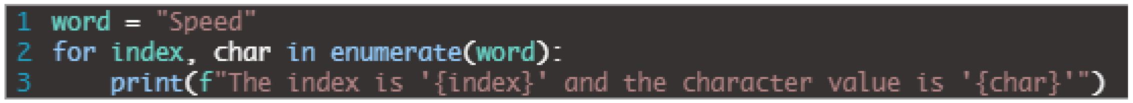
```
colors = ["red", "green", "blue", "purple"]
ratios = [0.2, 0.3, 0.1, 0.4]
for color, ratio in zip(colors, ratios):
    print("{}% {}".format(ratio * 100, color))
```

https://treyhunner.com/2016/04/how-to-loop-with-indexes-in-python/#enumerate

Enumerate string, list,...

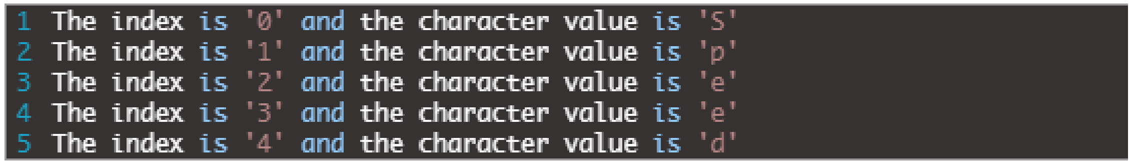






```python
word = "Speed"
for index, char in enumerate(word):
    print(f"The index is '{index}' and the character value is '{char}'")
```

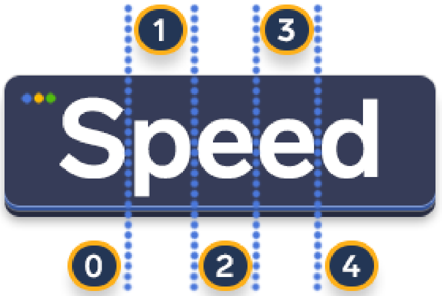
And here's the output:

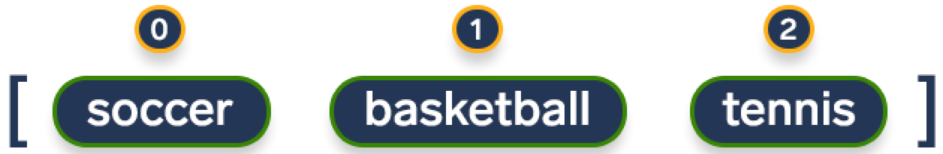


```
The index is '0' and the character value is 'S'
The index is '1' and the character value is 'p'
The index is '2' and the character value is 'e'
The index is '3' and the character value is 'e'
The index is '4' and the character value is 'd'
```

https://blog.soshace.com/en/python/python-enumerate-explained-and-visualized/

Enumerate string, list,...





```python
word = "Speed"
for index, char in enumerate(word):
    print(f"The index is '{index}' and the character value is '{char}'")
```

And here's the output:

```
The index is '0' and the ch
The index is '1' and the ch
The index is '2' and the ch
The index is '3' and the ch
The index is '4' and the ch
```

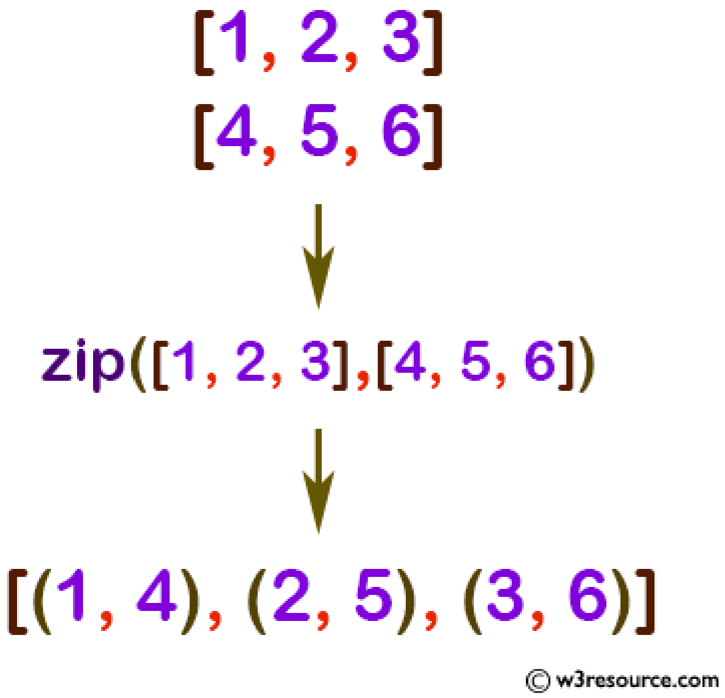
```python
sports = ['soccer', 'basketball', 'tennis']
for index, value in enumerate(sports):
    print(f"The item's index is {index} and its value is '{value}'")
```

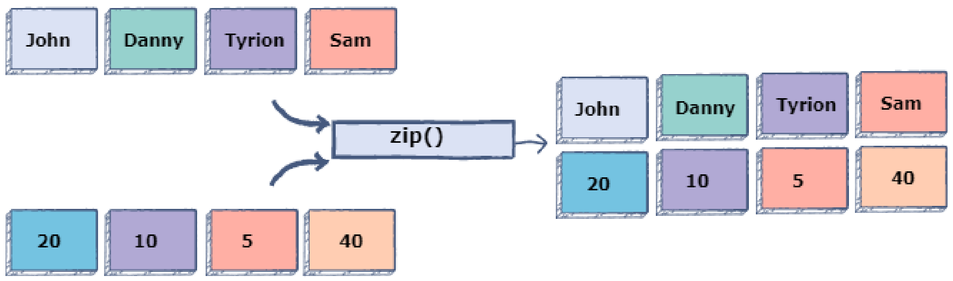
The output will be:

```
The item's index is 0 and its value is 'soccer'
The item's index is 1 and its value is 'basketball'
The item's index is 2 and its value is 'tennis'
```

https://blog.soshace.com/en/python/python-enumerate-explained-and-visualized/

zip





https://www.w3resource.com/python/built-in-function/zip.php

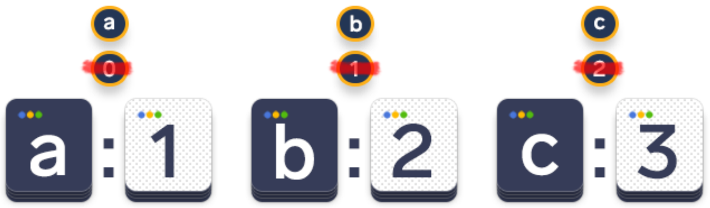
Enumerate & Zip

similar to example below

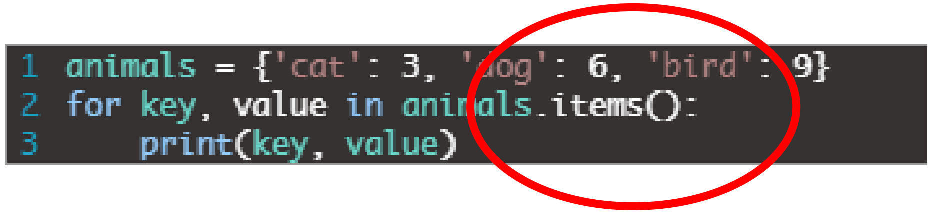
```
def enumerateNew( lst ) :
    l = []
    pos =range( len( lst ))
    for i in pos :
        l.append( (i,lst[i]))
    return  l
```

```
def zipNew( l1 , l2 ) :
    l=[]
    n = min( len(l1), len(l2))

    for i in range(n ) :
        l.append(  (l1[i] , l2[i]) )

    return l
```

Enumerate a dictionary

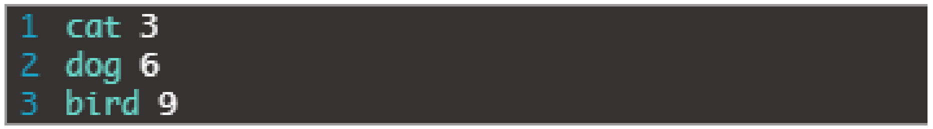


The keys are the "index"



```python
animals = {'cat': 3, 'dog': 6, 'bird': 9}
for key, value in animals.items():
    print(key, value)
```

The output will be:



```
cat 3
dog 6
bird 9
```

https://blog.soshace.com/en/python/python-enumerate-explained-and-visualized/

Read a list ...

Not different of reading a set of numbers, strings…

The difference is that you put them in a list

What does the following code does?

```
a=int( input("a?"))
while a>= 0 :
    a=int(input("a?"))
```

```
a=int( input("a?"))
while a!= 0 :
    a=int(input("a?"))
```

```
str=input("a?")
while str!="" :
    str= input("a?")
```

Condition on input
As a number
As a string
…

```
a=int( input("a?"))
while a%2== 0 :
    a=int(input("a?"))
```

```
str=input("a?")
while str!="end" :
    str= input("a?")
```

What does the following code does?

Read values while they >=0

```
a=int( input(“a?”))
while a>= 0 :
    a=int(input(“a?”))
```

Read values while they are !=0

```
a=int( input(“a?”))
while a!= 0 :
    a=int(input(“a?”))
```

Read string while they not empty

```
str=input(“a?”)
while str!=“” :
    str= input(“a?”)
```

Condition on input

As a number

As a string

…

```
a=int( input(“a?”))
while a%2== 0 :
    a=int(input(“a?”))
```

Read values while they are even

```
str=input(“a?”)
while str!=“end” :
    str= input(“a?”)
```

Read strings until “end” string is read

What does the following code does?

```
a=int( input(“a?”))
l=[]
while a>= 0 :
    l.append(a)
    a=int(input(“a?”))
```

```
a=int( input(“a?”))
l=[]
while a!= 0 :
    l.append(a)
    a=int(input(“a?”))
```

```
str=input(“a?”)
l=[]
while str!=“” :
    l.append(str)
    str= input(“a?”)
```

Condition on input

As a number

As a string

…

```
a=int( input(“a?”))
l=[]
while a%2== 0 :
    l.append(a)
    a=int(input(“a?”))
```

```
str=input(“a?”)
l=[]
while str!=“end” :
    l.append(a)
    str= input(“a?”)
```

Doing something with list & strings

Doing something with values

```
a=int(input("a?"))
while a!=0 :
    a=int( input("a?"))
```

The task

```
a=int(input("a?"))
mx=a
while a!=0 :
    If a>mx :
        mx = a
    a=int( input("a?"))
```

```
a=int(input("a?"))
mn=a
while a!=0 :
    If a<mn :
        mn = a
    a=int( input("a?"))
```

```
a=int(input("a?"))
sum=0
while a>= 0 :
    sum=sum+a
    a=int(input("a?"))
```

Doing something with values

```
a=int(input("a?"))
while a!=0 :
    a=int( input("a?"))
```

The task

Read values while they are !=0 and finds the minimum

```
a=int(input("a?"))
mn=a
while a!=0 :
    If a<mn :
        mn = a
    a=int( input("a?"))
```

```
a=int(input("a?"))
mx=a
while a!=0 :
```

Read values while they are !=0 and finds the maximum

```
    a=int( input("a?"))
```

```
a=int(input("a?"))
sum=0
while a>= 0 :
    sum=sum+a
    a=int(input("a?"))
```

Read values while they are >=0 and finds their sum

Doing something with values in a list l

The task

```
mx=0
i=1
while i<len(l) :
    If l[i]>l[mx] :
        mx = i
```

```
mn=l[0]
for a in l :
    If a<mn :
        mn = a
```

```
…
sum=0
for i,a in enumerate(l):
    sum=sum+a
```

Doing something with values in a list l

The task

```
mx=0
i=1
while i<len(l) :
```

finds the position of maximum in list l

```
      mx = i
```

```
mn=l[0]
for a in l :
   If a<mn :
      mn = a
```

finds the minimum in list l

…

```
sum=0
for i,a in enumerate(l) :
   sum=sum+a
```

Calculates the sum of list l

Doing something with values in a list l

The task

```
mx=l[0]
i=1
while i<len(l) :
    If l[i]>mx :
        mx = l[i]
```

```
mn = min( l )
```

```
…
sum=0
for a in l :
    sum=sum+a
```

Doing something with values in a list l

The task

```
mx=l[0]
i=1
while i<len(l) :
```

finds the position of maximum position in list l

```
mn = min( l )
```

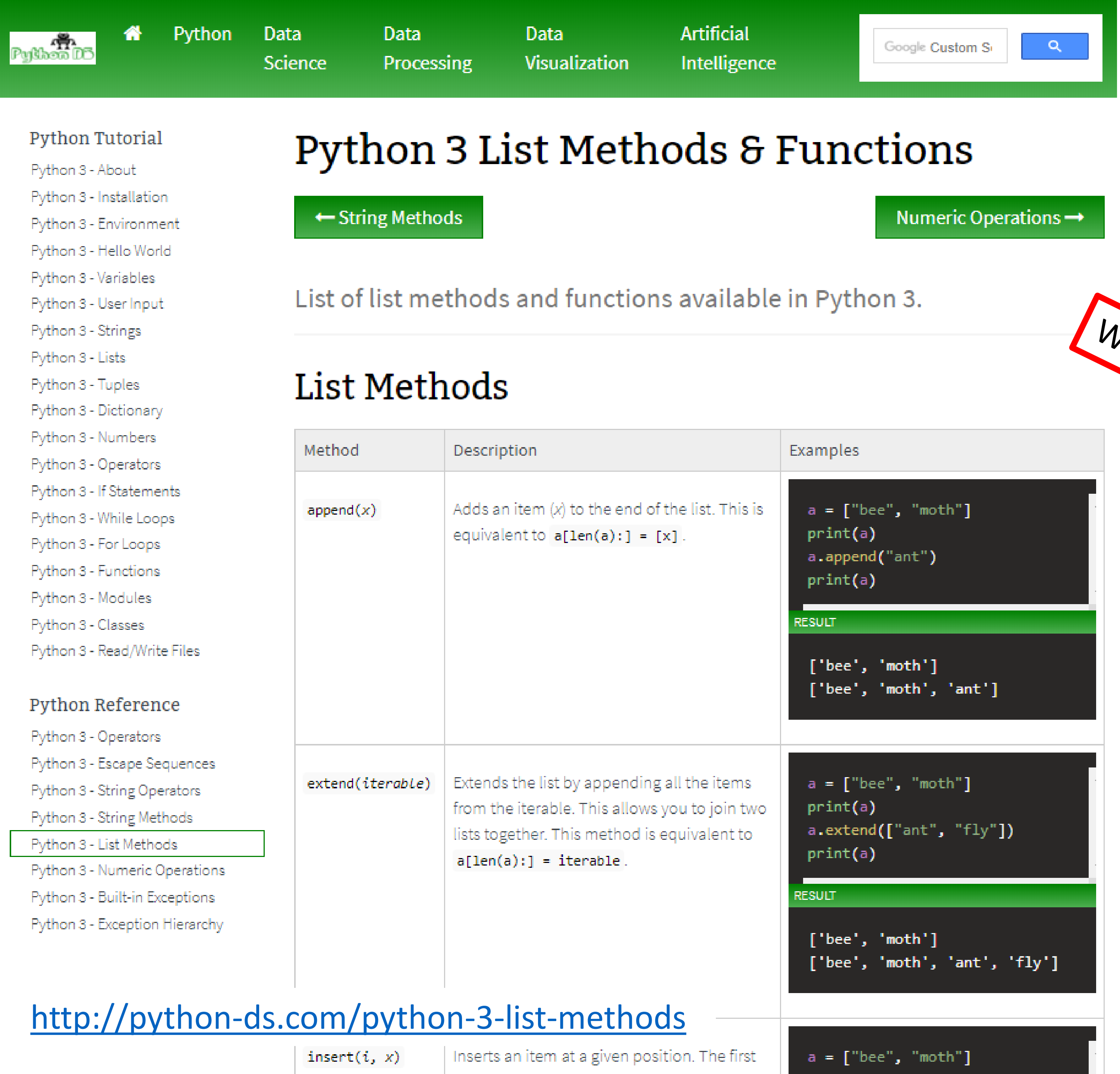
finds the minimum in list l

```
...
sum=0
for a in l :
    sum=sum+a
```

Calculates the sum of list l

Built in function

- Already defined, No need to define them
- Exist in Lists, strings & dictionaries
- Useful / common operations



Python Tutorial

- Python 3 - About
- Python 3 - Installation
- Python 3 - Environment
- Python 3 - Hello World
- Python 3 - Variables
- Python 3 - User Input
- Python 3 - Strings
- Python 3 - Lists
- Python 3 - Tuples
- Python 3 - Dictionary
- Python 3 - Numbers
- Python 3 - Operators
- Python 3 - If Statements
- Python 3 - While Loops
- Python 3 - For Loops
- Python 3 - Functions
- Python 3 - Modules
- Python 3 - Classes
- Python 3 - Read/Write Files

Python Reference

- Python 3 - Operators
- Python 3 - Escape Sequences
- Python 3 - String Operators
- Python 3 - String Methods
- Python 3 - List Methods
- Python 3 - Numeric Operations
- Python 3 - Built-in Exceptions
- Python 3 - Exception Hierarchy

Python 3 List Methods & Functions

← String Methods   Numeric Operations →

List of list methods and functions available in Python 3.

WORTH A LOOK

List Methods

Method	Description	Examples

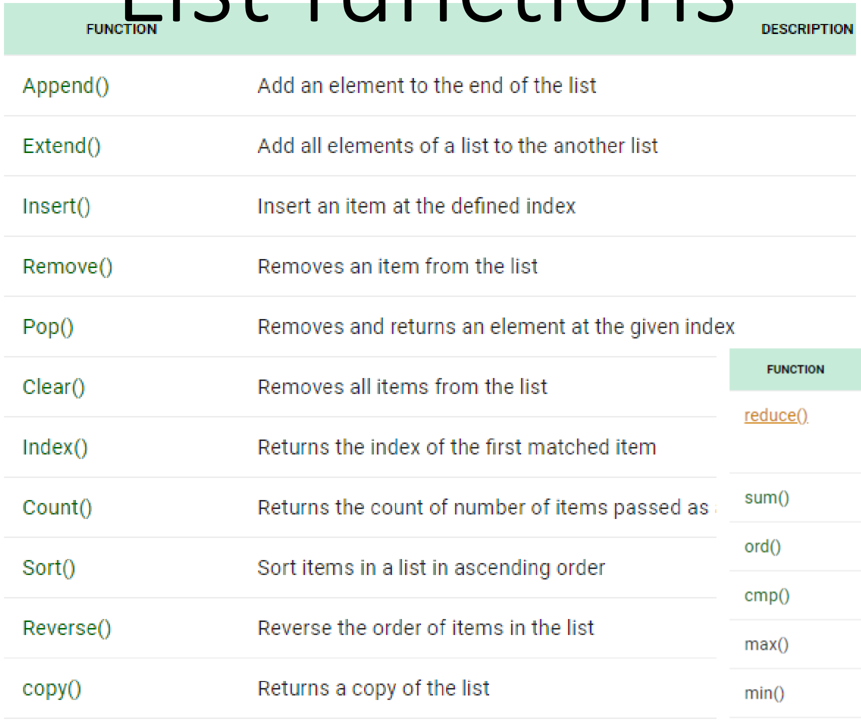
`append(x)`	Adds an item (*x*) to the end of the list. This is equivalent to `a[len(a):] = [x]`.	`a = ["bee", "moth"]`
		`print(a)`
		`a.append("ant")`
		`print(a)`
		RESULT
		`['bee', 'moth']`
		`['bee', 'moth', 'ant']`

`extend(iterable)`	Extends the list by appending all the items from the iterable. This allows you to join two lists together. This method is equivalent to `a[len(a):] = iterable`.	`a = ["bee", "moth"]`
		`print(a)`
		`a.extend(["ant", "fly"])`
		`print(a)`
		RESULT
		`['bee', 'moth']`
		`['bee', 'moth', 'ant', 'fly']`

`insert(i, x)`	Inserts an item at a given position. The first	`a = ["bee", "moth"]`

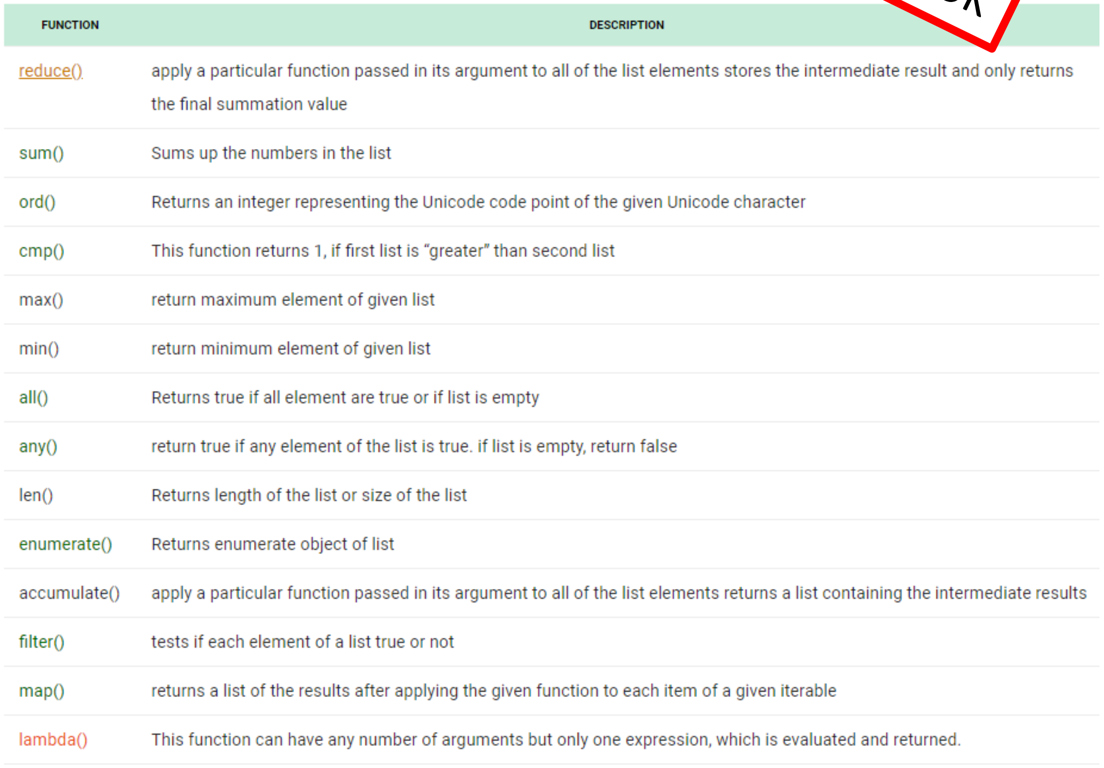
http://python-ds.com/python-3-list-methods

List functions



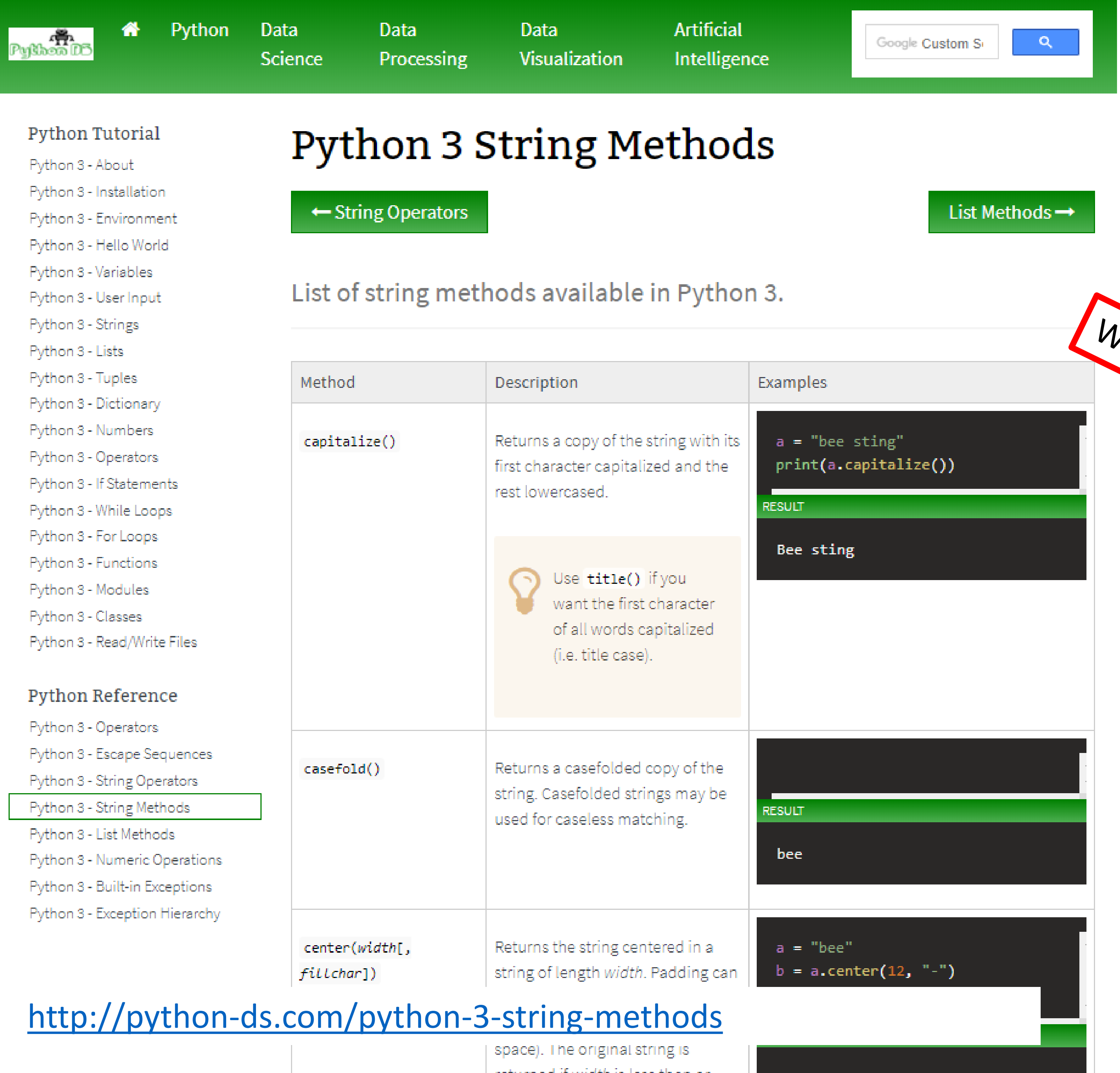
FUNCTION	DESCRIPTION
Append()	Add an element to the end of the list
Extend()	Add all elements of a list to the another list
Insert()	Insert an item at the defined index
Remove()	Removes an item from the list
Pop()	Removes and returns an element at the given index
Clear()	Removes all items from the list
Index()	Returns the index of the first matched item
Count()	Returns the count of number of items passed as
Sort()	Sort items in a list in ascending order
Reverse()	Reverse the order of items in the list
copy()	Returns a copy of the list

WORTH A LOOK



FUNCTION	DESCRIPTION
reduce()	apply a particular function passed in its argument to all of the list elements stores the intermediate result and only returns the final summation value
sum()	Sums up the numbers in the list
ord()	Returns an integer representing the Unicode code point of the given Unicode character
cmp()	This function returns 1, if first list is "greater" than second list
max()	return maximum element of given list
min()	return minimum element of given list
all()	Returns true if all element are true or if list is empty
any()	return true if any element of the list is true. if list is empty, return false
len()	Returns length of the list or size of the list
enumerate()	Returns enumerate object of list
accumulate()	apply a particular function passed in its argument to all of the list elements returns a list containing the intermediate results
filter()	tests if each element of a list true or not
map()	returns a list of the results after applying the given function to each item of a given iterable
lambda()	This function can have any number of arguments but only one expression, which is evaluated and returned.

https://www.geeksforgeeks.org/python-list/



Python Tutorial

- Python 3 - About
- Python 3 - Installation
- Python 3 - Environment
- Python 3 - Hello World
- Python 3 - Variables
- Python 3 - User Input
- Python 3 - Strings
- Python 3 - Lists
- Python 3 - Tuples
- Python 3 - Dictionary
- Python 3 - Numbers
- Python 3 - Operators
- Python 3 - If Statements
- Python 3 - While Loops
- Python 3 - For Loops
- Python 3 - Functions
- Python 3 - Modules
- Python 3 - Classes
- Python 3 - Read/Write Files

Python Reference

- Python 3 - Operators
- Python 3 - Escape Sequences
- Python 3 - String Operators
- Python 3 - String Methods
- Python 3 - List Methods
- Python 3 - Numeric Operations
- Python 3 - Built-in Exceptions
- Python 3 - Exception Hierarchy

Python 3 String Methods

← String Operators List Methods →

List of string methods available in Python 3.

Method	Description	Examples

`capitalize()`	Returns a copy of the string with its first character capitalized and the rest lowercased. Use `title()` if you want the first character of all words capitalized (i.e. title case).	`a = "bee sting"`
		`print(a.capitalize())`
		RESULT
		`Bee sting`

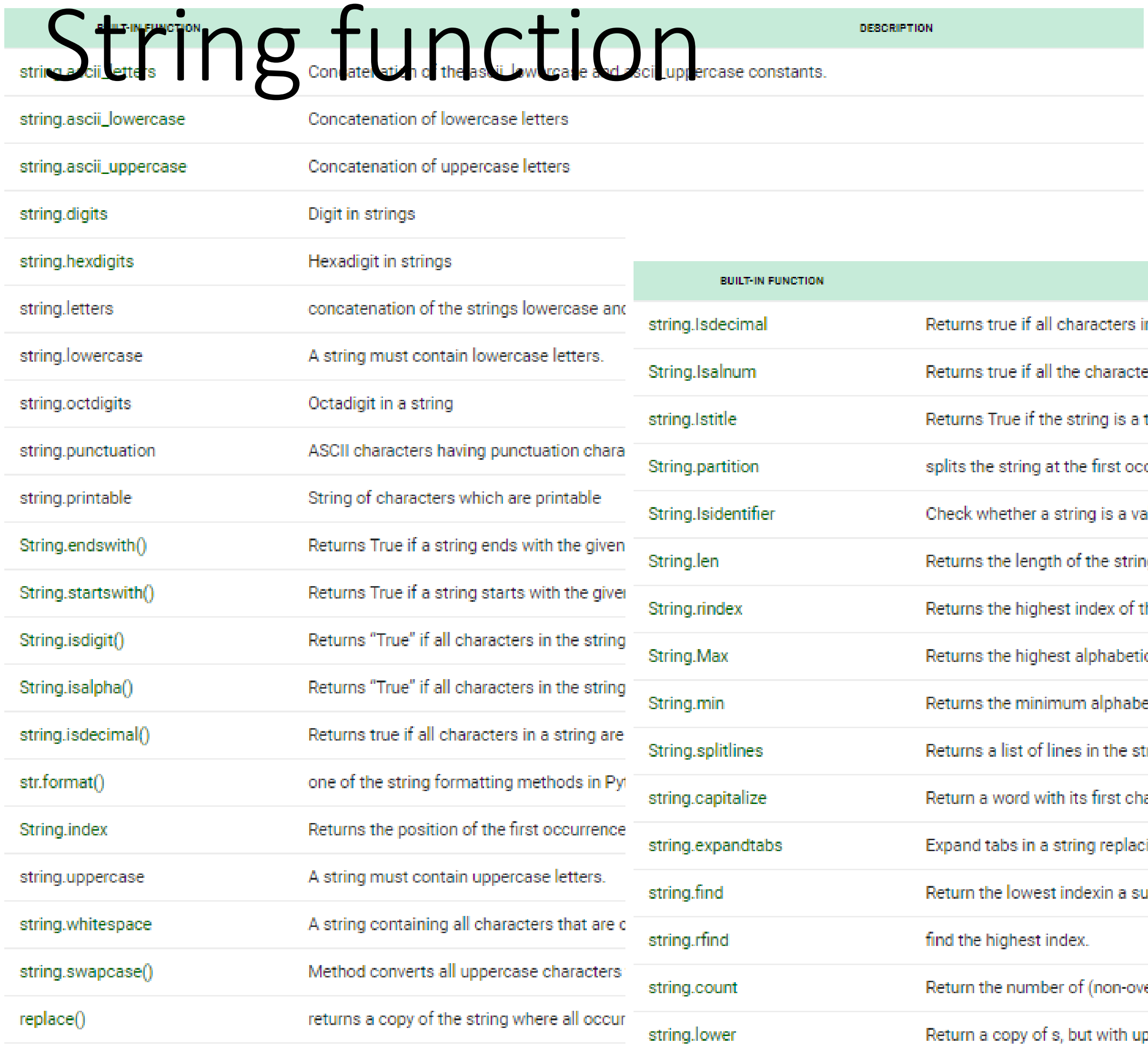
`casefold()`	Returns a casefolded copy of the string. Casefolded strings may be used for caseless matching.	RESULT `bee`

`center(`*`width`*`[, `*`fillchar`*`])`	Returns the string centered in a string of length *width*. Padding can … space). The original string is	`a = "bee"`
		`b = a.center(12, "-")`

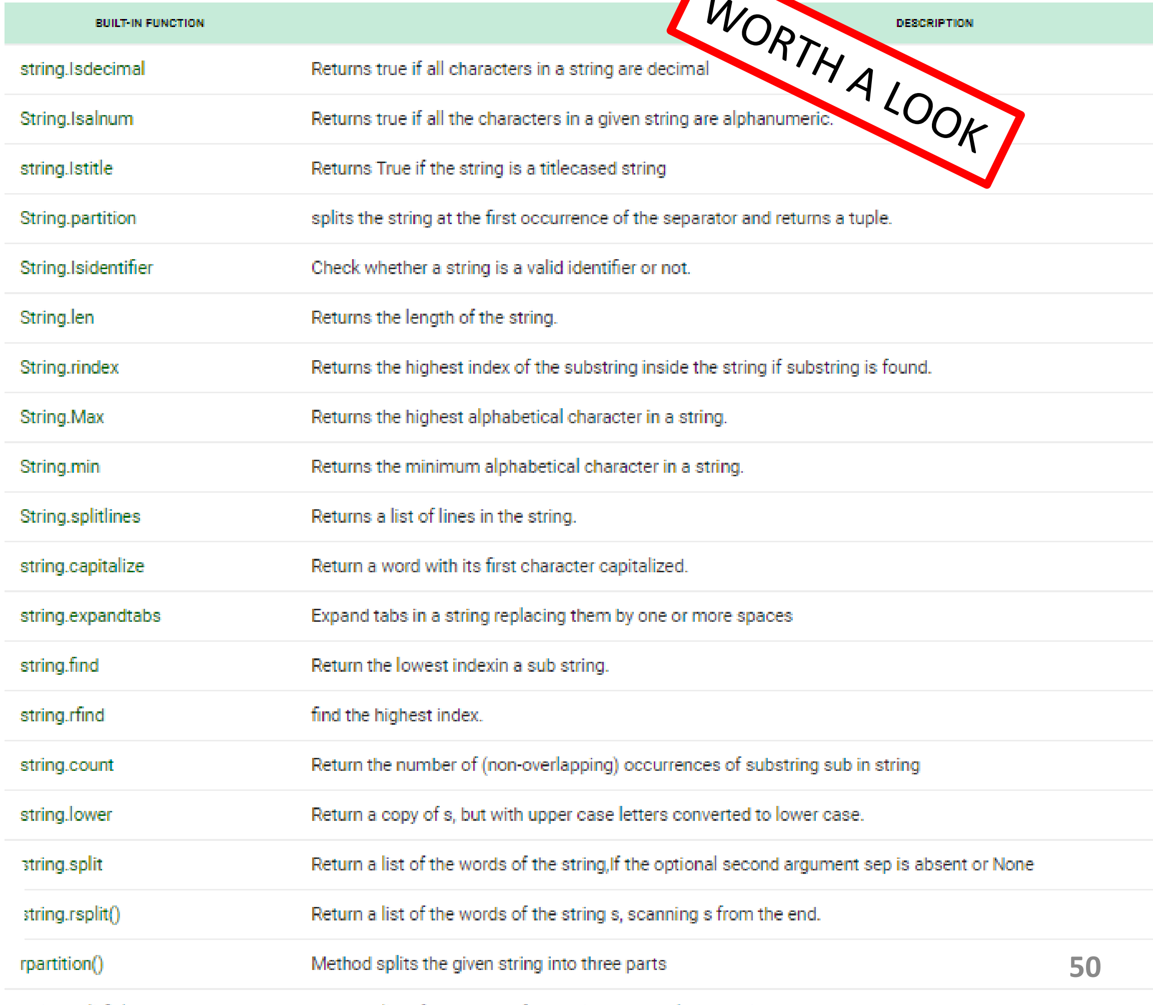
WORTH A LOOK

http://python-ds.com/python-3-string-methods

String function



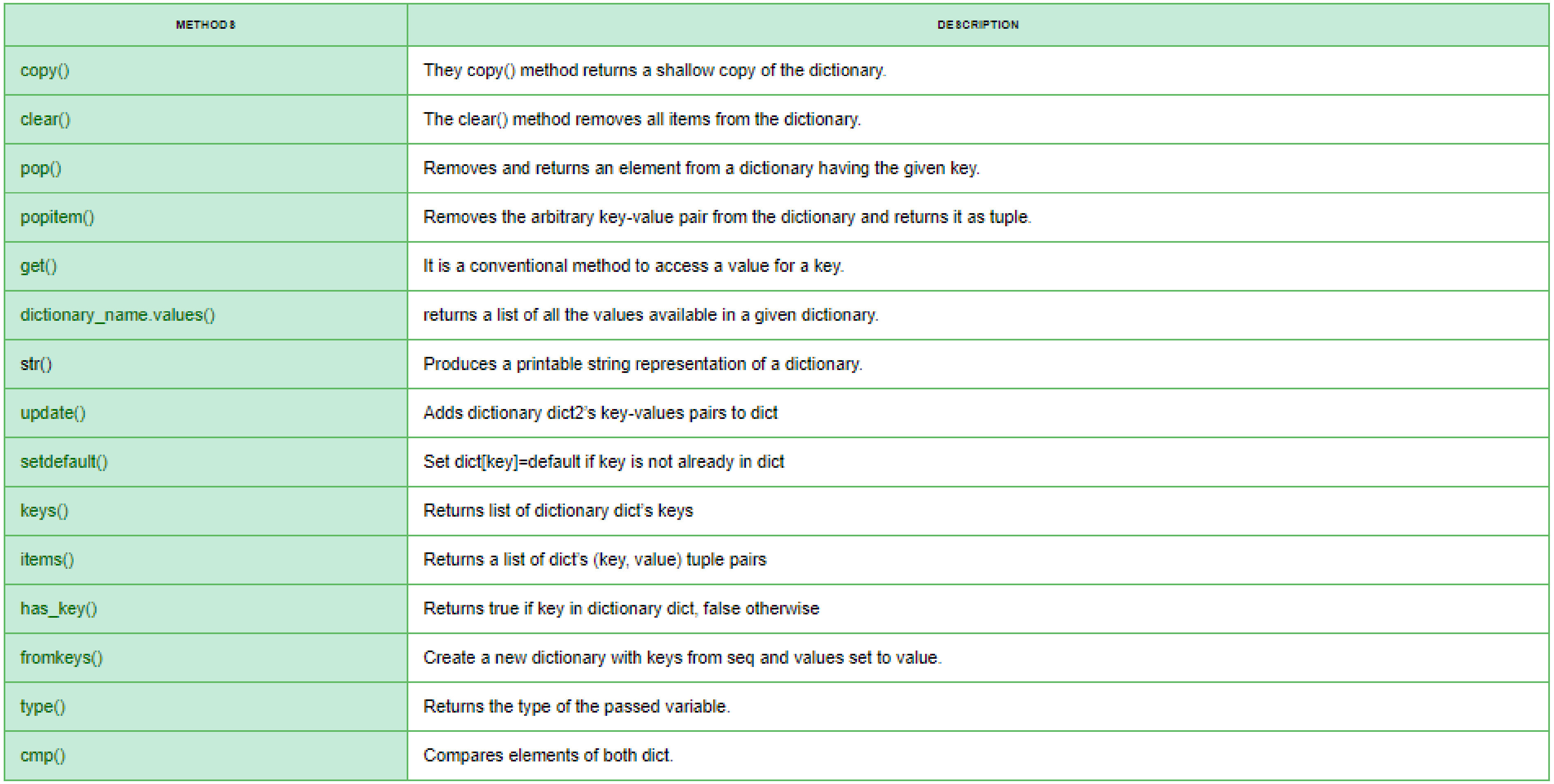
BUILT-IN FUNCTION	DESCRIPTION
string.ascii_letters	Concatenation of the ascii_lowercase and ascii_uppercase constants.
string.ascii_lowercase	Concatenation of lowercase letters
string.ascii_uppercase	Concatenation of uppercase letters
string.digits	Digit in strings
string.hexdigits	Hexadigit in strings
string.letters	concatenation of the strings lowercase anc
string.lowercase	A string must contain lowercase letters.
string.octdigits	Octadigit in a string
string.punctuation	ASCII characters having punctuation chara
string.printable	String of characters which are printable
String.endswith()	Returns True if a string ends with the given
String.startswith()	Returns True if a string starts with the giver
String.isdigit()	Returns "True" if all characters in the string
String.isalpha()	Returns "True" if all characters in the string
string.isdecimal()	Returns true if all characters in a string are
str.format()	one of the string formatting methods in Py
String.index	Returns the position of the first occurrence
string.uppercase	A string must contain uppercase letters.
string.whitespace	A string containing all characters that are c
string.swapcase()	Method converts all uppercase characters
replace()	returns a copy of the string where all occur


WORTH A LOOK

BUILT-IN FUNCTION	DESCRIPTION
string.Isdecimal	Returns true if all characters in a string are decimal
String.Isalnum	Returns true if all the characters in a given string are alphanumeric.
string.Istitle	Returns True if the string is a titlecased string
String.partition	splits the string at the first occurrence of the separator and returns a tuple.
String.Isidentifier	Check whether a string is a valid identifier or not.
String.len	Returns the length of the string.
String.rindex	Returns the highest index of the substring inside the string if substring is found.
String.Max	Returns the highest alphabetical character in a string.
String.min	Returns the minimum alphabetical character in a string.
String.splitlines	Returns a list of lines in the string.
string.capitalize	Return a word with its first character capitalized.
string.expandtabs	Expand tabs in a string replacing them by one or more spaces
string.find	Return the lowest indexin a sub string.
string.rfind	find the highest index.
string.count	Return the number of (non-overlapping) occurrences of substring sub in string
string.lower	Return a copy of s, but with upper case letters converted to lower case.
string.split	Return a list of the words of the string,If the optional second argument sep is absent or None
string.rsplit()	Return a list of the words of the string s, scanning s from the end.
rpartition()	Method splits the given string into three parts

https://www.geeksforgeeks.org/python-strings/
https://realpython.com/python-strings/

Dictionary functions

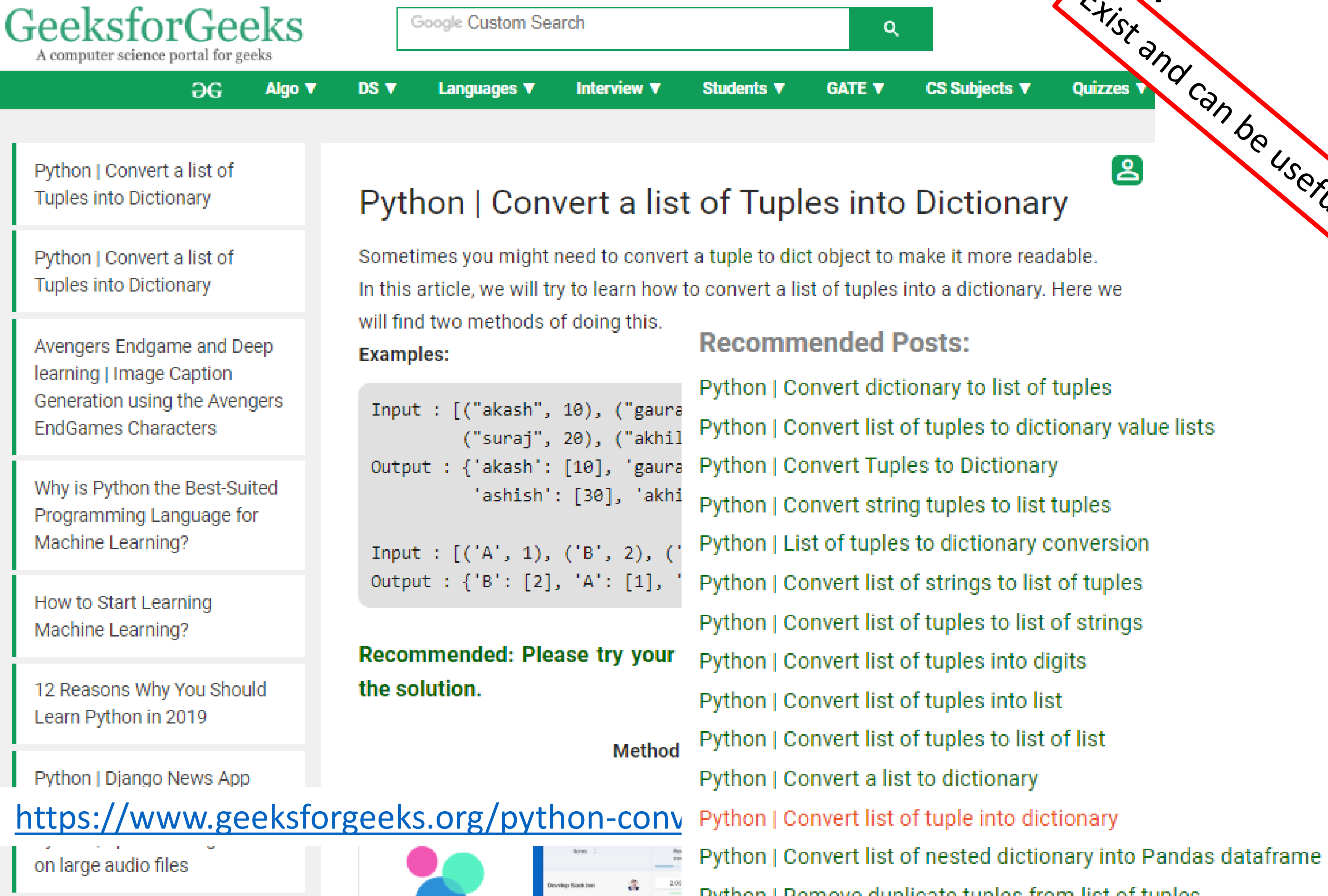


METHODS	DESCRIPTION
copy()	They copy() method returns a shallow copy of the dictionary.
clear()	The clear() method removes all items from the dictionary.
pop()	Removes and returns an element from a dictionary having the given key.
popitem()	Removes the arbitrary key-value pair from the dictionary and returns it as tuple.
get()	It is a conventional method to access a value for a key.
dictionary_name.values()	returns a list of all the values available in a given dictionary.
str()	Produces a printable string representation of a dictionary.
update()	Adds dictionary dict2's key-values pairs to dict
setdefault()	Set dict[key]=default if key is not already in dict
keys()	Returns list of dictionary dict's keys
items()	Returns a list of dict's (key, value) tuple pairs
has_key()	Returns true if key in dictionary dict, false otherwise
fromkeys()	Create a new dictionary with keys from seq and values set to value.
type()	Returns the type of the passed variable.
cmp()	Compares elements of both dict.

https://www.geeksforgeeks.org/python-dictionary/

Convertion between list & tuples

Exist and can be useful



Python | Convert a list of Tuples into Dictionary

Python | Convert a list of Tuples into Dictionary

Avengers Endgame and Deep learning | Image Caption Generation using the Avengers EndGames Characters

Why is Python the Best-Suited Programming Language for Machine Learning?

How to Start Learning Machine Learning?

12 Reasons Why You Should Learn Python in 2019

Python | Django News App

on large audio files

Python | Convert a list of Tuples into Dictionary

Sometimes you might need to convert a tuple to dict object to make it more readable.
In this article, we will try to learn how to convert a list of tuples into a dictionary. Here we will find two methods of doing this.

Examples:

```
Input : [("akash", 10), ("gaura
         ("suraj", 20), ("akhil
Output : {'akash': [10], 'gaura
          'ashish': [30], 'akhi

Input : [('A', 1), ('B', 2), ('
Output : {'B': [2], 'A': [1], '
```

Recommended: Please try your the solution.

Method

Recommended Posts:

Python | Convert dictionary to list of tuples
Python | Convert list of tuples to dictionary value lists
Python | Convert Tuples to Dictionary
Python | Convert string tuples to list tuples
Python | List of tuples to dictionary conversion
Python | Convert list of strings to list of tuples
Python | Convert list of tuples to list of strings
Python | Convert list of tuples into digits
Python | Convert list of tuples into list
Python | Convert list of tuples to list of list
Python | Convert a list to dictionary
Python | Convert list of tuple into dictionary
Python | Convert list of nested dictionary into Pandas dataframe
Python | Remove duplicate tuples from list of tuples

https://www.geeksforgeeks.org/python-conv

The END