Course	
Term	
Week	
Date	
Chapter. Topic	9.2. Dictionaries and Sets

Dictionaries

Siva R Jasthi

Computer Science and Cybersecurity

Metropolitan State University

Outline

- Lists
- Tuples
- Sets

are all collections

We covered “Lists”, “Tuples” and “Sets” so far.

We will cover “Dictionaries” today.

Lists vs Tuples vs Sets vs Dictionary

	Lists	Tuples	Set	Dictionary
Ordered	✓	✓	✗	✗
Indexed	✓	✓	✗	✗
Add or Update items	✓	✗	✓	✓
Can contain duplicates	✓	✓	✗	✗
Supports Keys (Name: Values)	✗	✗	✗	✓
Uses	Square Brackets	Round Brackets	Curly Brackets	Curly Brackets
	[ ], list()	**( ), tuple()**	~~{ }~~ **set()**	**{ }, dict()**

Dictionary

A dictionary is a collection which is **unordered**, **changeable,** does not allow duplicates and supports the concept of keys

Dictionary: An introduction
https://www.w3schools.com/python/python_dictionaries.asp

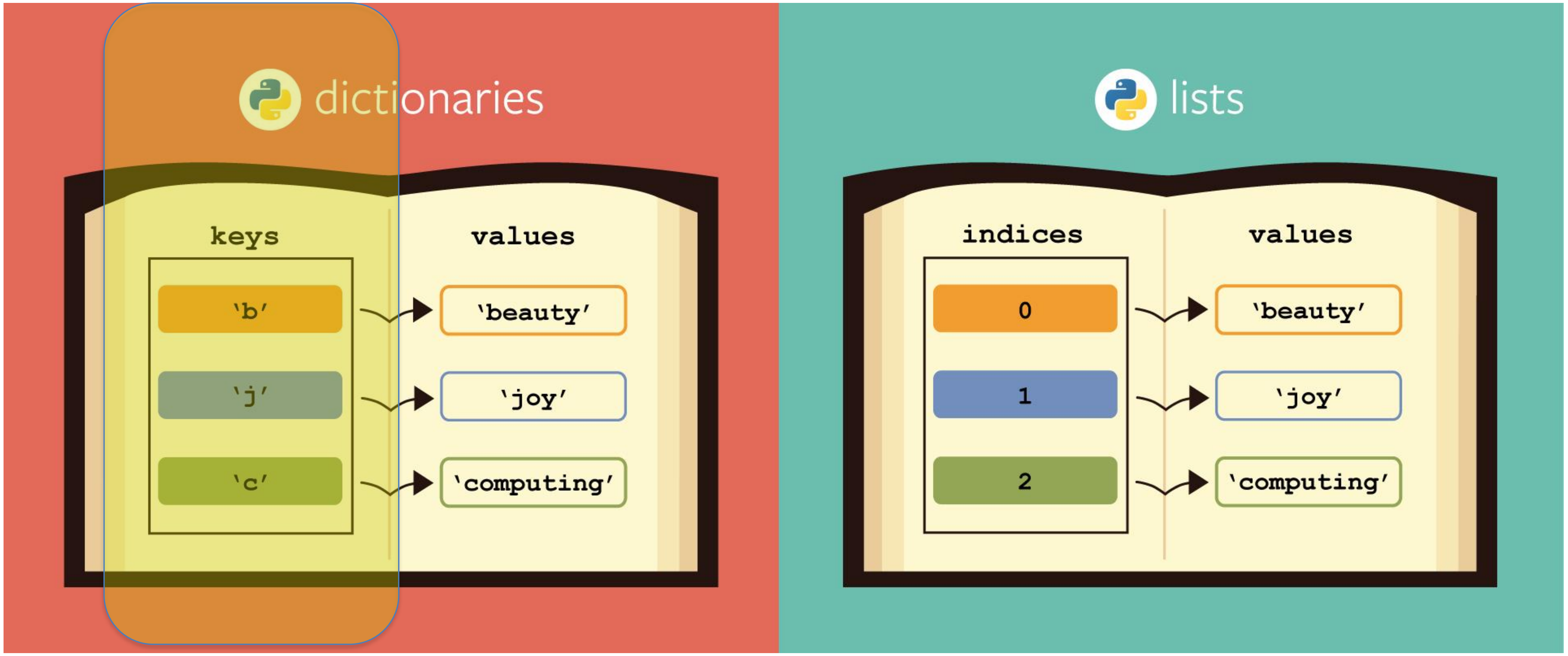
Dictionary Methods
https://www.w3schools.com/python/python_ref_dictionary.asp

Built-in Functions
https://docs.python.org/3/library/functions.html

Dictionary vs Lists

nums1 = [‘b’, ‘joy’, ‘c’]   nums[1]   nums_dict ={} nums[‘j’]

In lists, we have a “numerical index”. We do not have any control on it.
In dictionaries, we have a “named index”. We call it as “Key”. We have better control on it. We can change it as needed.



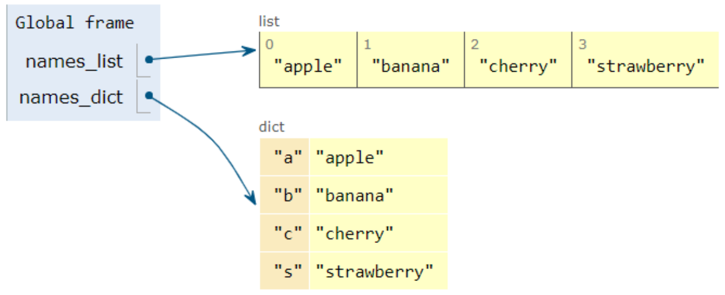
Dictionary vs Lists: An analogy

Lists are linear. Dictionary is not linear.

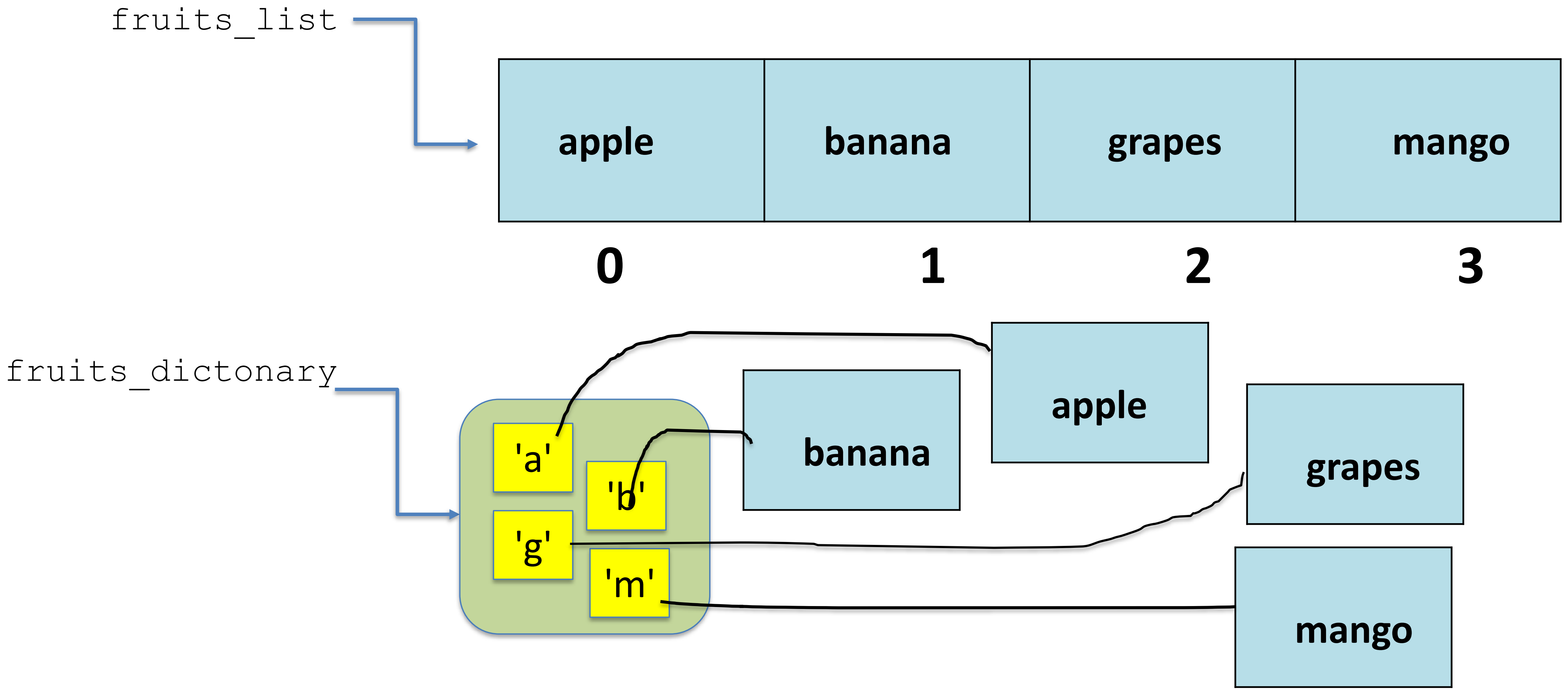
List is ordered. Dictionary is not ordered.

Items in List are accessed by the index (numerical number)
Items in Dictionary are accessed by the named index (a KEY)

Dictionary vs Lists: An analogy



Lists are linear. Dictionary is not linear.
List is ordered. Dictionary is not ordered.

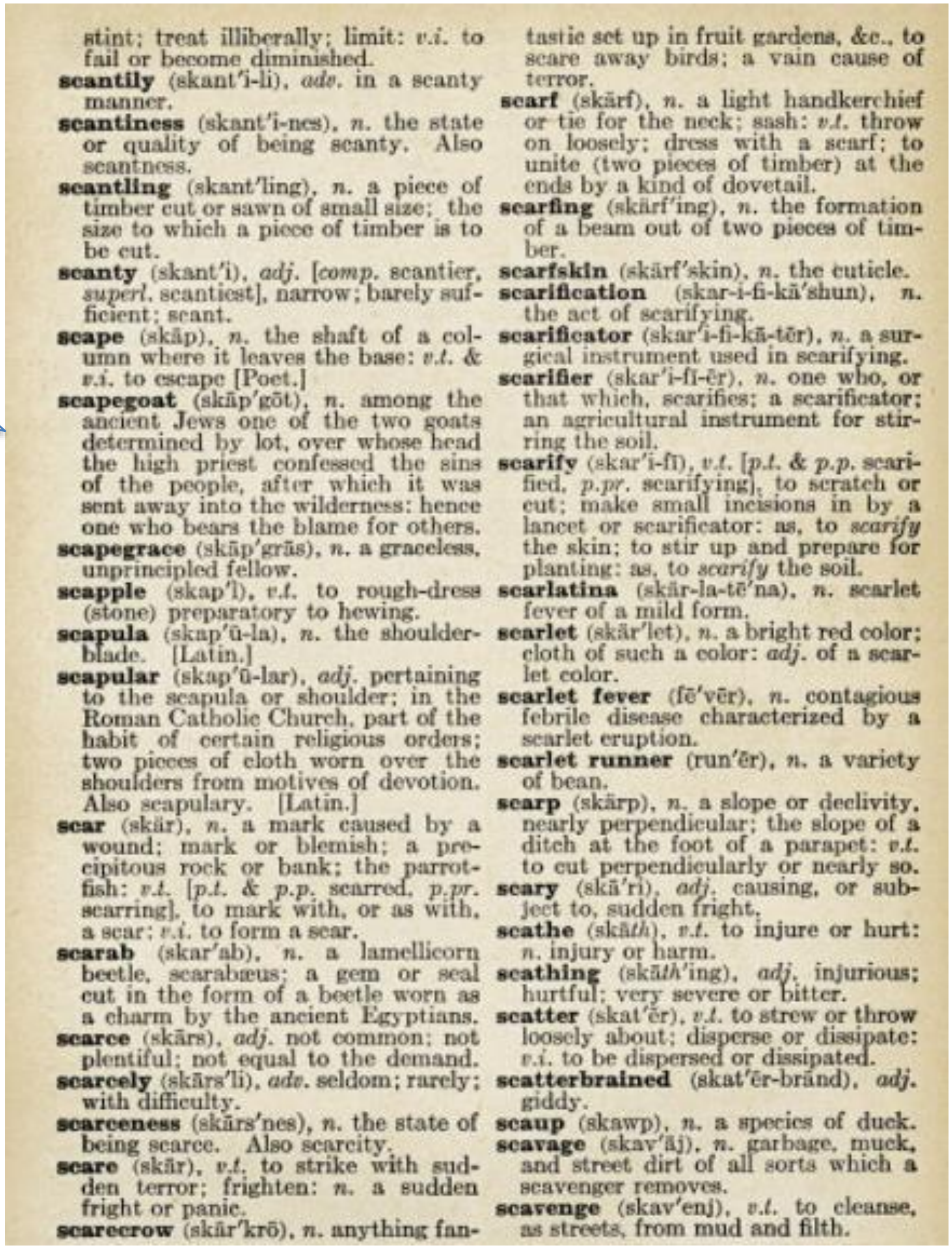


Dictionary: Terminology

Dictionary Terms:

- Key
- Value
- Key-Value Pair
- Name-Value Pairs
- Map
- Mapping
- Association

Note: Dictonary reflects KEY-VALUE pairs.
The keys in real dictionary (book) are ordered.
However, python dictionary is NOT ordered



stint; treat illiberally; limit: *v.i.* to fail or become diminished.

scantily (skant′i-li), *adv.* in a scanty manner.

scantiness (skant′i-nes), *n.* the state or quality of being scanty. Also scantness.

scantling (skant′ling), *n.* a piece of timber cut or sawn of small size; the size to which a piece of timber is to be cut.

scanty (skant′i), *adj.* [*comp.* scantier, *superl.* scantiest], narrow; barely sufficient; scant.

scape (skāp), *n.* the shaft of a column where it leaves the base: *v.t.* & *v.i.* to escape [Poet.]

scapegoat (skāp′gōt), *n.* among the ancient Jews one of the two goats determined by lot, over whose head the high priest confessed the sins of the people, after which it was sent away into the wilderness: hence one who bears the blame for others.

scapegrace (skāp′grās), *n.* a graceless, unprincipled fellow.

scapple (skap′l), *v.t.* to rough-dress (stone) preparatory to hewing.

scapula (skap′ū-la), *n.* the shoulder-blade. [Latin.]

scapular (skap′ū-lar), *adj.* pertaining to the scapula or shoulder; in the Roman Catholic Church, part of the habit of certain religious orders; two pieces of cloth worn over the shoulders from motives of devotion. Also scapulary. [Latin.]

scar (skär), *n.* a mark caused by a wound; mark or blemish; a precipitous rock or bank; the parrot-fish: *v.t.* [*p.t.* & *p.p.* scarred, *p.pr.* scarring], to mark with, or as with, a scar; *v.i.* to form a scar.

scarab (skar′ab), *n.* a lamellicorn beetle, scarabæus; a gem or seal cut in the form of a beetle worn as a charm by the ancient Egyptians.

scarce (skārs), *adj.* not common; not plentiful; not equal to the demand.

scarcely (skārs′li), *adv.* seldom; rarely; with difficulty.

scarceness (skārs′nes), *n.* the state of being scarce. Also scarcity.

scare (skār), *v.t.* to strike with sudden terror; frighten: *n.* a sudden fright or panic.

scarecrow (skār′krō), *n.* anything fantastic set up in fruit gardens, &c., to scare away birds; a vain cause of terror.

scarf (skärf), *n.* a light handkerchief or tie for the neck; sash: *v.t.* throw on loosely; dress with a scarf; to unite (two pieces of timber) at the ends by a kind of dovetail.

scarfing (skärf′ing), *n.* the formation of a beam out of two pieces of timber.

scarfskin (skärf′skin), *n.* the cuticle.

scarification (skar-i-fi-kā′shun), *n.* the act of scarifying.

scarificator (skar′i-fi-kā-tēr), *n.* a surgical instrument used in scarifying.

scarifier (skar′i-fī-ēr), *n.* one who, or that which, scarifies; a scarificator; an agricultural instrument for stirring the soil.

scarify (skar′i-fī), *v.t.* [*p.t.* & *p.p.* scarified, *p.pr.* scarifying], to scratch or cut; make small incisions in by a lancet or scarificator: as, to *scarify* the skin; to stir up and prepare for planting: as, to *scarify* the soil.

scarlatina (skär-la-tē′na), *n.* scarlet fever of a mild form.

scarlet (skär′let), *n.* a bright red color; cloth of such a color: *adj.* of a scarlet color.

scarlet fever (fē′vēr), *n.* contagious febrile disease characterized by a scarlet eruption.

scarlet runner (run′ēr), *n.* a variety of bean.

scarp (skärp), *n.* a slope or declivity, nearly perpendicular; the slope of a ditch at the foot of a parapet: *v.t.* to cut perpendicularly or nearly so.

scary (skā′ri), *adj.* causing, or subject to, sudden fright.

scathe (skāth), *v.t.* to injure or hurt: *n.* injury or harm.

scathing (skāth′ing), *adj.* injurious; hurtful; very severe or bitter.

scatter (skat′ēr), *v.t.* to strew or throw loosely about; disperse or dissipate: *v.i.* to be dispersed or dissipated.

scatterbrained (skat′ēr-brānd), *adj.* giddy.

scaup (skawp), *n.* a species of duck.

scavage (skav′āj), *n.* garbage, muck, and street dirt of all sorts which a scavenger removes.

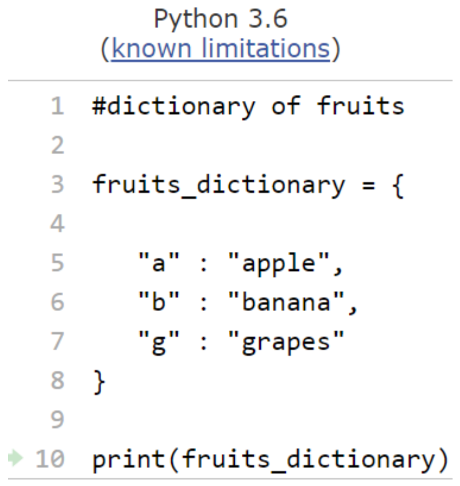
scavenge (skav′enj), *v.t.* to cleanse, as streets, from mud and filth.

Dictionaries have different names in different languages

- Associative Arrays - Perl / Php
- Properties or Map or HashMap - Java
- Property Bag - C# / .NET

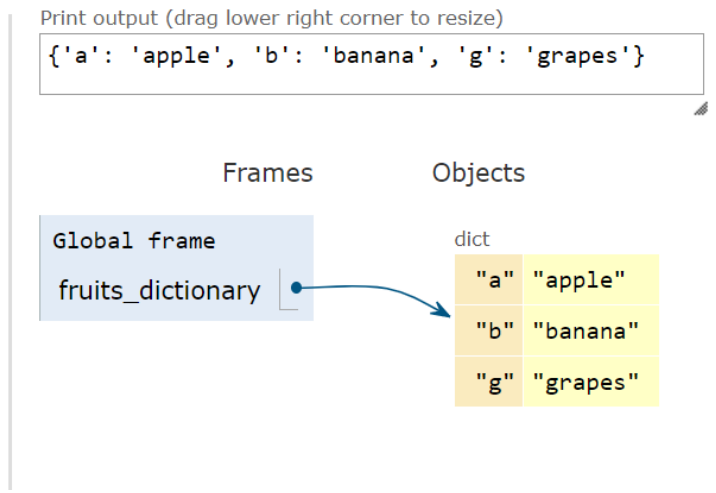
An example of a dictionary (link)

We use CURLY Brackets to indicate a dictionary. And we separate the keys and values using : (colon)



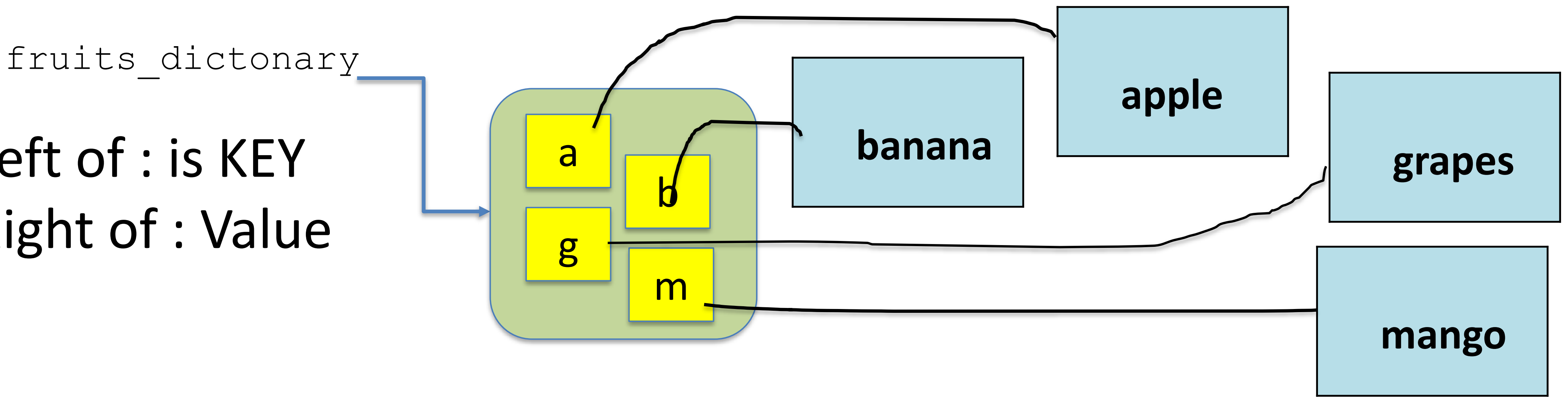
Python 3.6
(known limitations)

```
#dictionary of fruits

fruits_dictionary = {

    "a" : "apple",
    "b" : "banana",
    "g" : "grapes"
}

print(fruits_dictionary)
```



Print output (drag lower right corner to resize)

```
{'a': 'apple', 'b': 'banana', 'g': 'grapes'}
```



Left of : is KEY
Right of : Value

What can be used as a KEY?

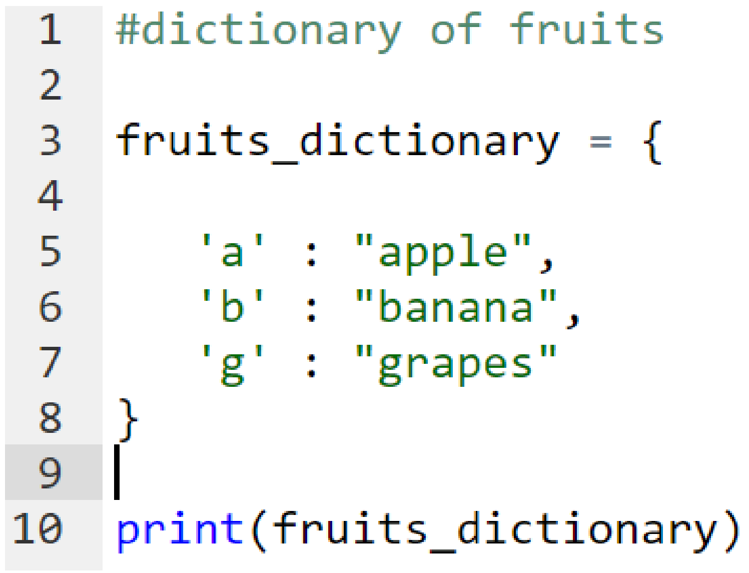
We can use the following data types as KEYs:

- Strings
- Characters
- Integers
- Floats
- Booleans
- Other Objects (Tuples, Sets, User Defined Objects)

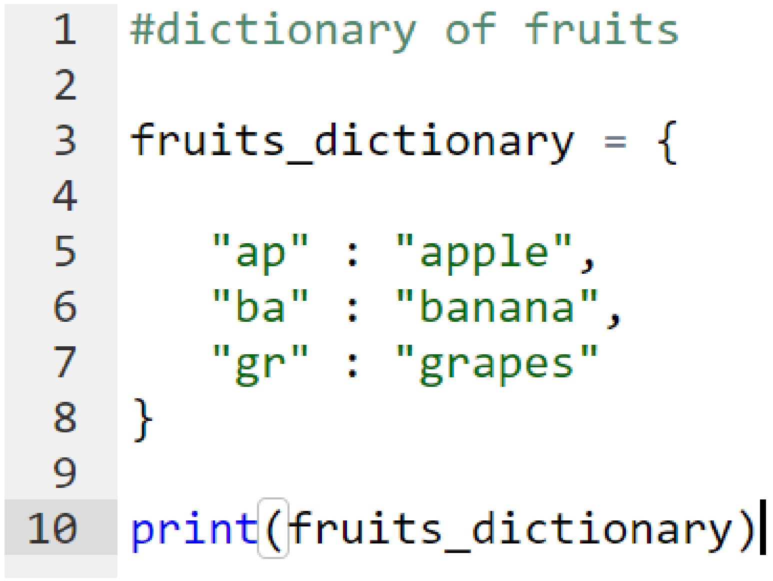
However, the following can not be used as KEYs

- Lists
- Dictionaries

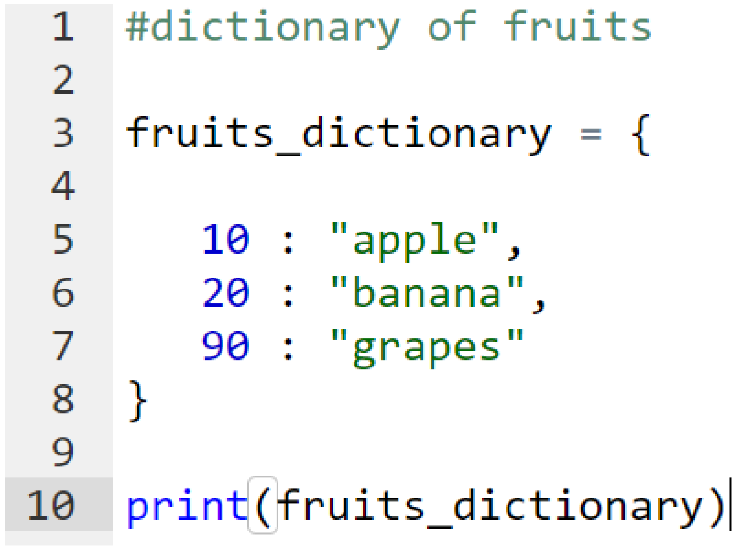
What can be used as a KEY?



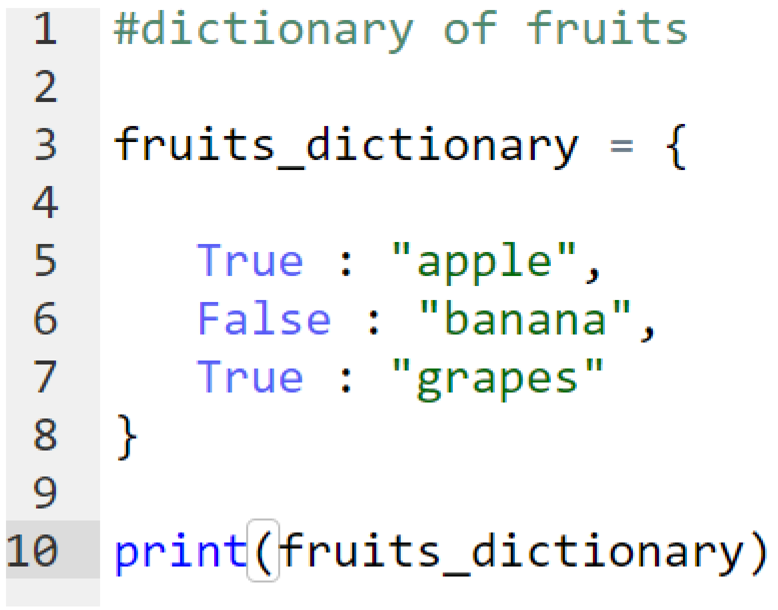
```python
#dictionary of fruits

fruits_dictionary = {

    'a' : "apple",
    'b' : "banana",
    'g' : "grapes"
}

print(fruits_dictionary)
```



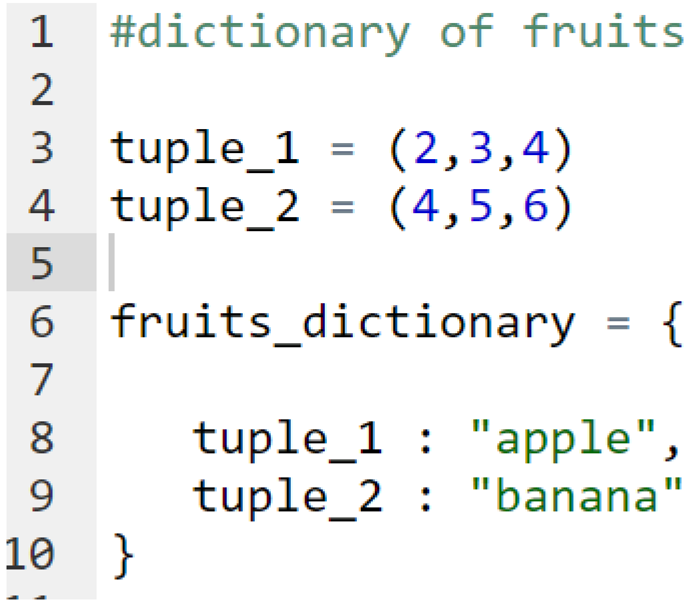
```python
#dictionary of fruits

fruits_dictionary = {

    "ap" : "apple",
    "ba" : "banana",
    "gr" : "grapes"
}

print(fruits_dictionary)
```



```python
#dictionary of fruits

fruits_dictionary = {

    10 : "apple",
    20 : "banana",
    90 : "grapes"
}

print(fruits_dictionary)
```



```python
#dictionary of fruits

fruits_dictionary = {

    True : "apple",
    False : "banana",
    True : "grapes"
}

print(fruits_dictionary)
```



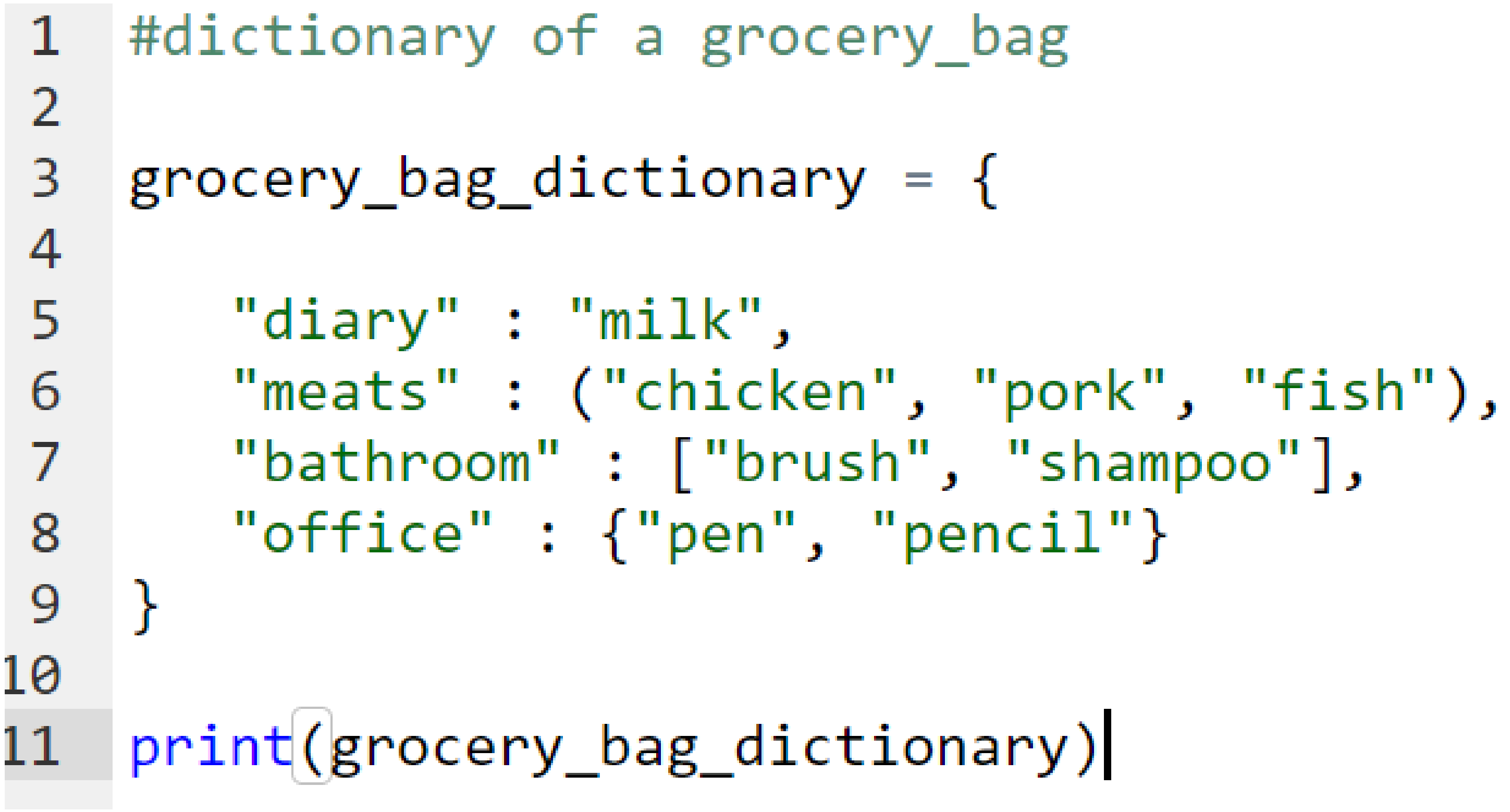
```python
#dictionary of fruits

tuple_1 = (2,3,4)
tuple_2 = (4,5,6)

fruits_dictionary = {

    tuple_1 : "apple",
    tuple_2 : "banana"
}
```

What can be used as a VALUE?

What can be used as a VALUE? (link)

ANYTHING

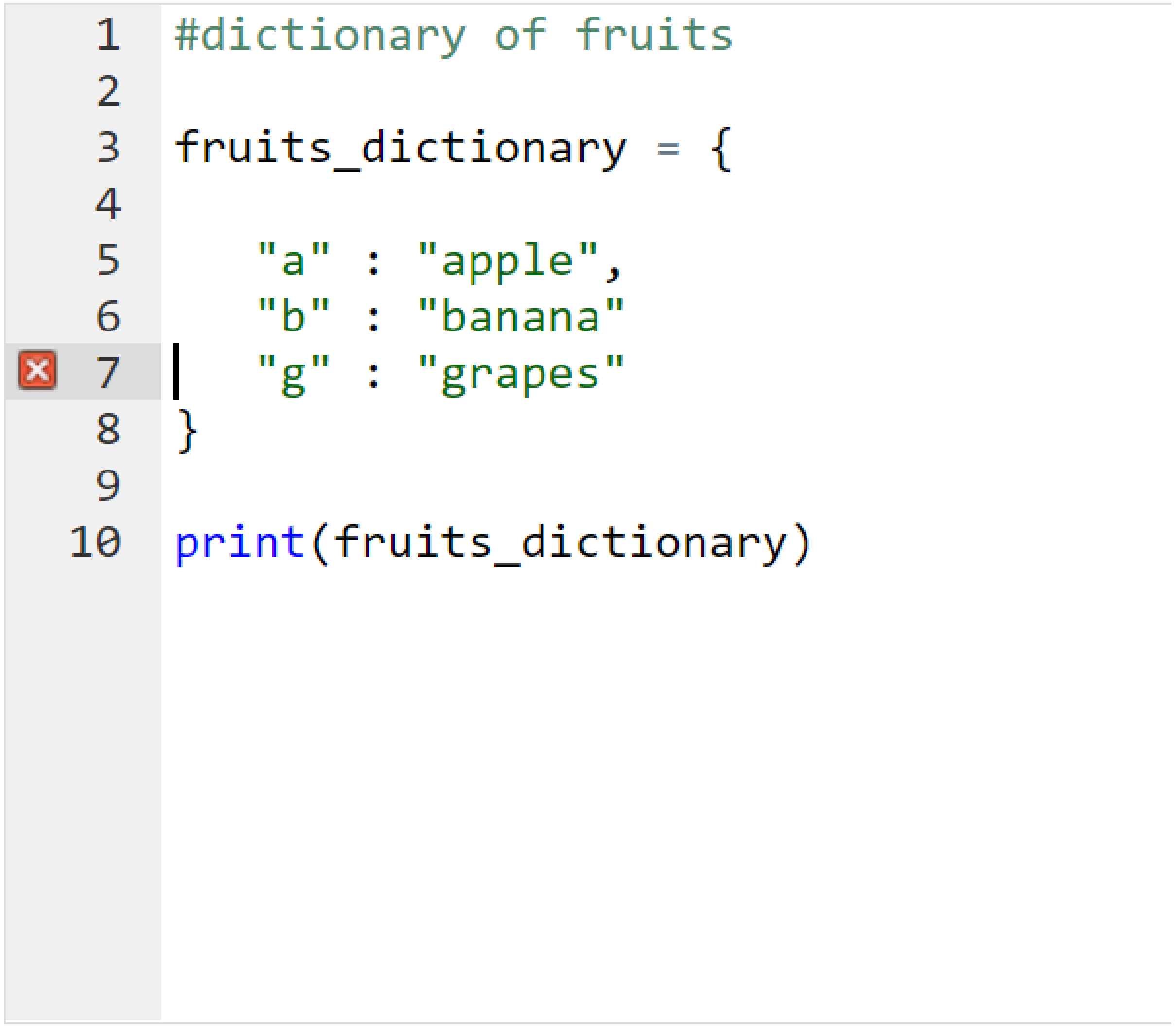
And you can also have different data types for different keys



```python
#dictionary of a grocery_bag

grocery_bag_dictionary = {

    "diary" : "milk",
    "meats" : ("chicken", "pork", "fish"),
    "bathroom" : ["brush", "shampoo"],
    "office" : {"pen", "pencil"}
}

print(grocery_bag_dictionary)
```

Common Mistakes #1

What is wrong with this code?

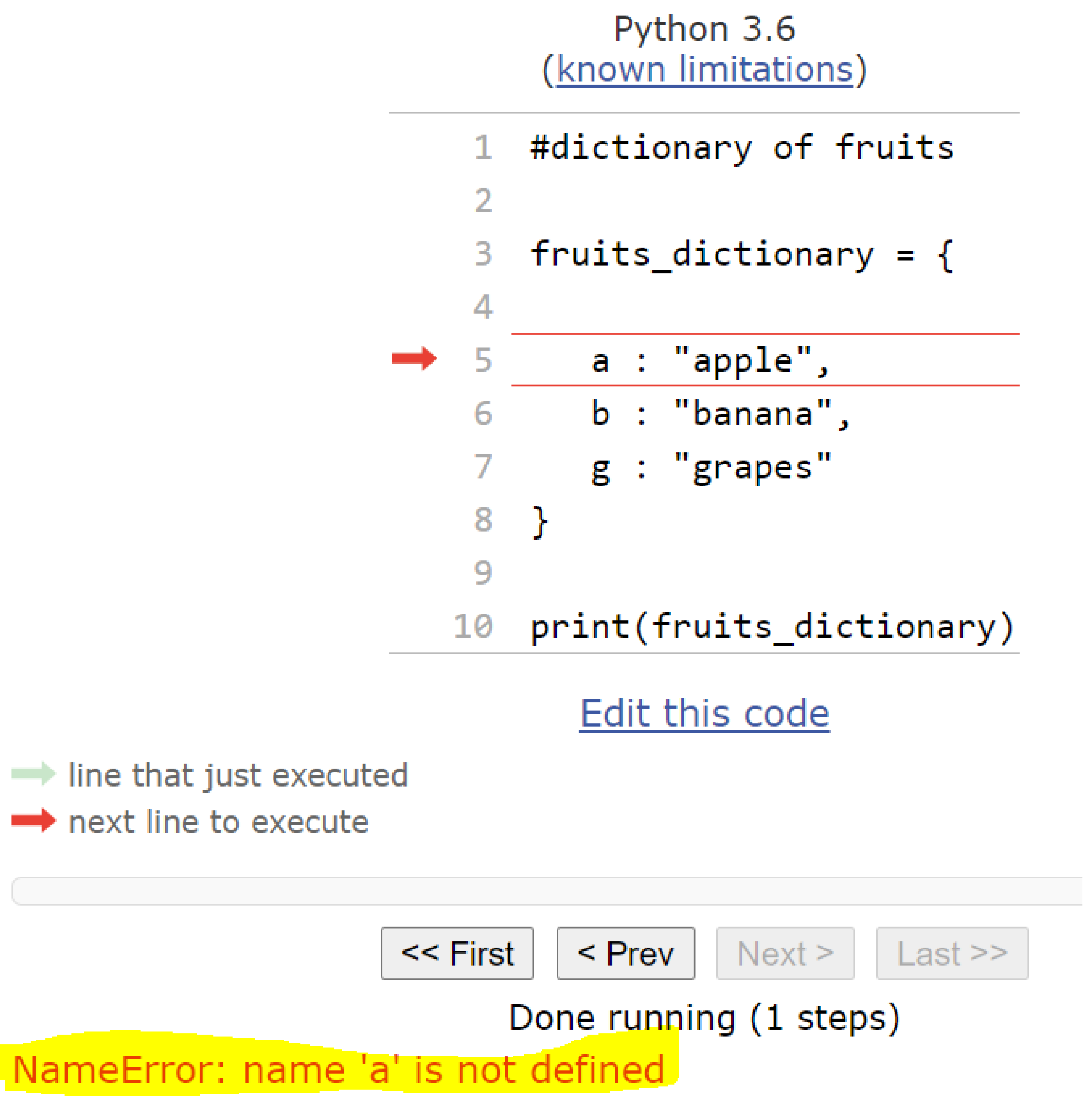


```
 1  #dictionary of fruits
 2
 3  fruits_dictionary = {
 4
 5      "a" : "apple",
 6      "b" : "banana"
 7      "g" : "grapes"
 8  }
 9
10  print(fruits_dictionary)
```

```
SyntaxError: invalid syntax (<string>, line 7)
```

Common Mistakes #2

What is wrong with this code?



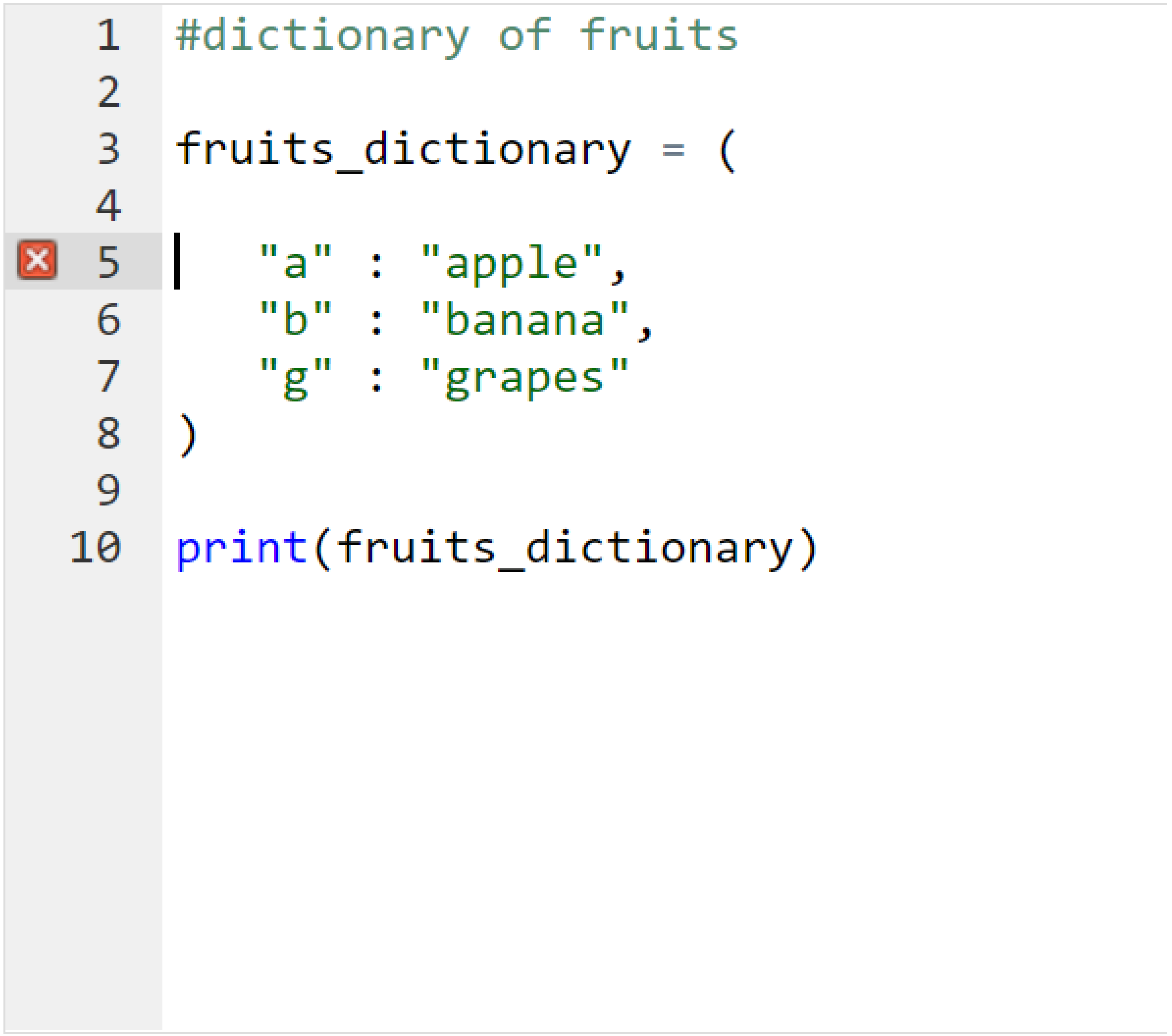
```
#dictionary of fruits

fruits_dictionary = {

    a : "apple",
    b : "banana",
    g : "grapes"
}

print(fruits_dictionary)
```

Done running (1 steps)

NameError: name 'a' is not defined

Common Mistakes #3

What is wrong with this code?

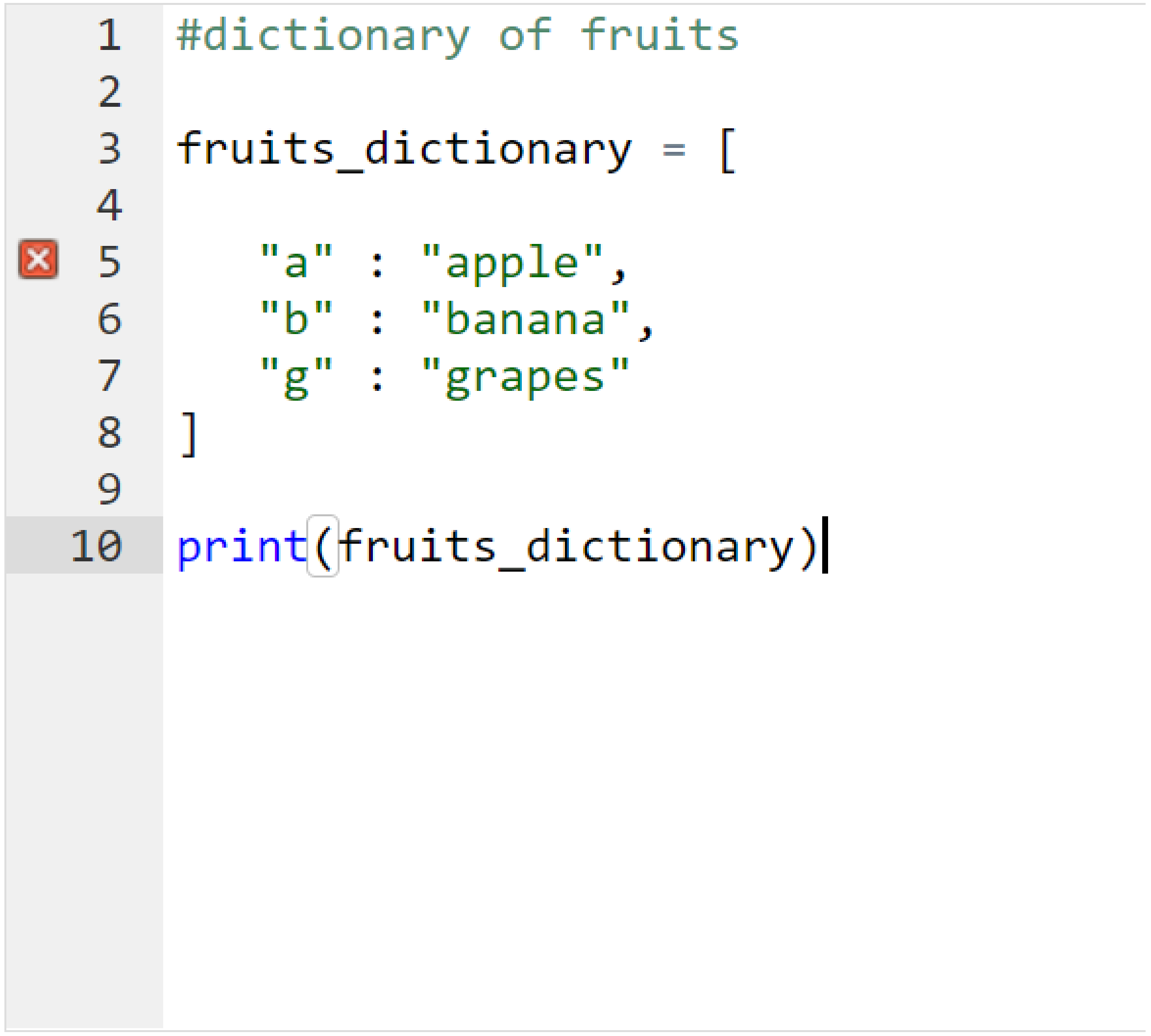


```
 1  #dictionary of fruits
 2
 3  fruits_dictionary = (
 4
 5      "a" : "apple",
 6      "b" : "banana",
 7      "g" : "grapes"
 8  )
 9
10  print(fruits_dictionary)
```

SyntaxError: invalid syntax (<string>, line 5)

Common Mistakes #4

What is wrong with this code?



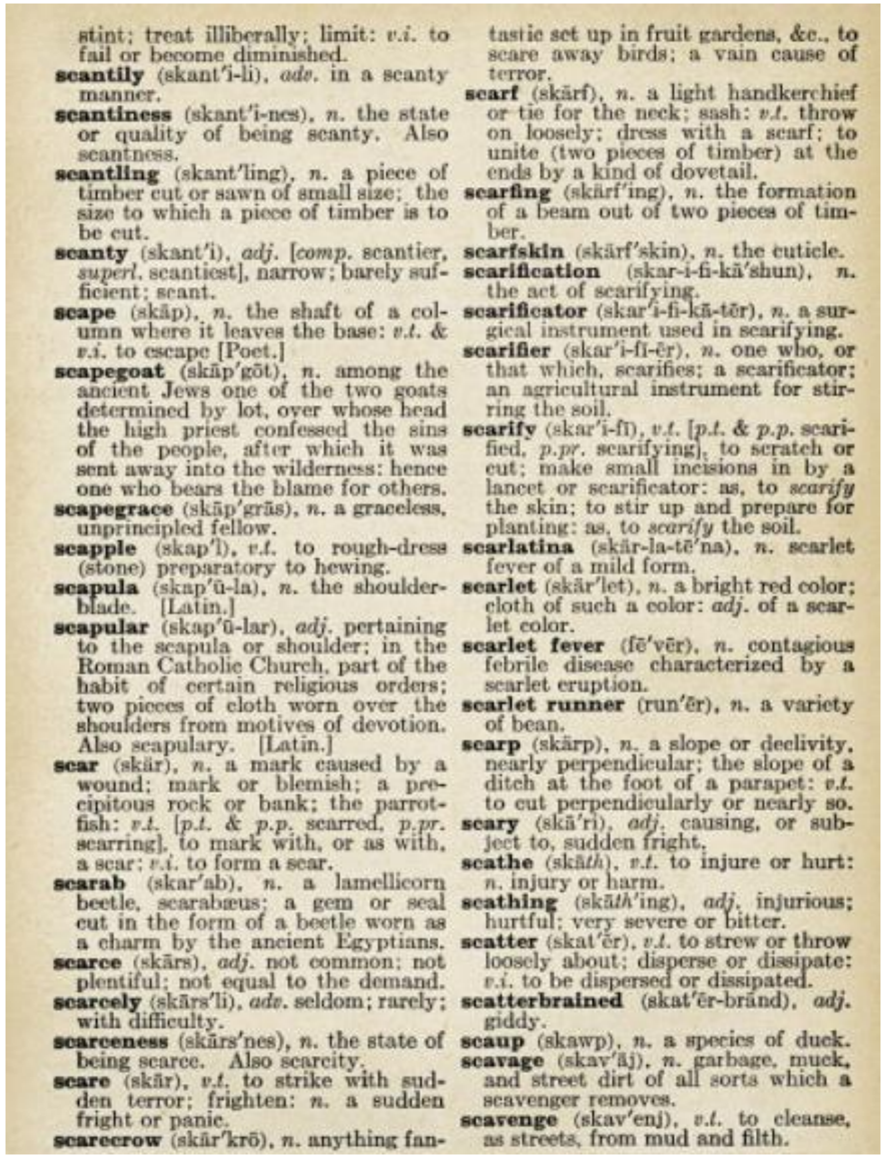
```
#dictionary of fruits

fruits_dictionary = [

    "a" : "apple",
    "b" : "banana",
    "g" : "grapes"
]

print(fruits_dictionary)
```

SyntaxError: invalid syntax (<string>, line 5)

Duplicates KEYs are not allowed

We can not have two KEYs.

(Analogy: You can not have two BOLD words in a dictionary). represent this collection as a “set”.

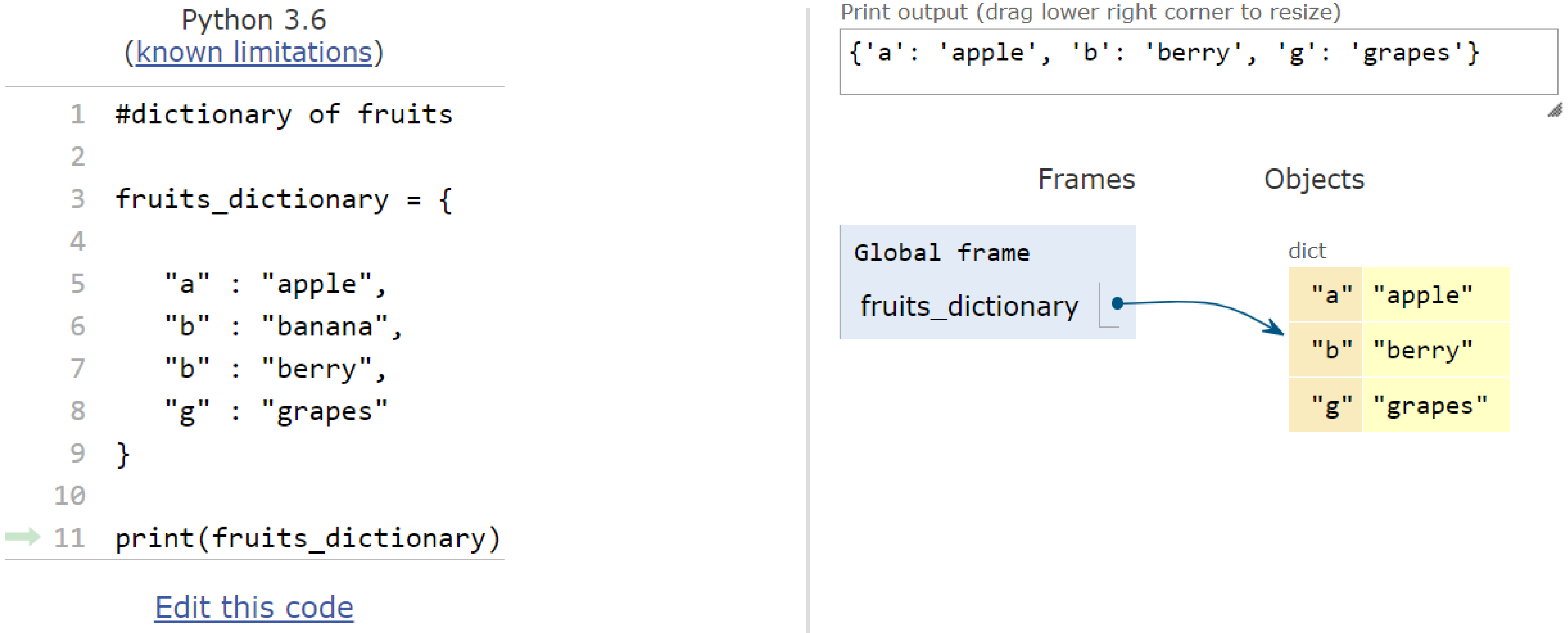


stint; treat illiberally; limit: *v.i.* to fail or become diminished.
scantily (skant′i-li), *adv.* in a scanty manner.
scantiness (skant′i-nes), *n.* the state or quality of being scanty. Also scantness.
scantling (skant′ling), *n.* a piece of timber cut or sawn of small size; the size to which a piece of timber is to be cut.
scanty (skant′i), *adj.* [*comp.* scantier, *superl.* scantiest], narrow; barely sufficient; scant.
scape (skāp), *n.* the shaft of a column where it leaves the base: *v.t.* & *v.i.* to escape [Poet.]
scapegoat (skāp′gōt), *n.* among the ancient Jews one of the two goats determined by lot, over whose head the high priest confessed the sins of the people, after which it was sent away into the wilderness: hence one who bears the blame for others.
scapegrace (skāp′grās), *n.* a graceless, unprincipled fellow.
scapple (skap′l), *v.t.* to rough-dress (stone) preparatory to hewing.
scapula (skap′ū-la), *n.* the shoulder-blade. [Latin.]
scapular (skap′ū-lar), *adj.* pertaining to the scapula or shoulder; in the Roman Catholic Church, part of the habit of certain religious orders; two pieces of cloth worn over the shoulders from motives of devotion. Also scapulary. [Latin.]
scar (skär), *n.* a mark caused by a wound; mark or blemish; a precipitous rock or bank; the parrot-fish: *v.t.* [*p.t.* & *p.p.* scarred, *p.pr.* scarring], to mark with, or as with, a scar; *v.i.* to form a scar.
scarab (skar′ab), *n.* a lamellicorn beetle, scarabæus; a gem or seal cut in the form of a beetle worn as a charm by the ancient Egyptians.
scarce (skārs), *adj.* not common; not plentiful; not equal to the demand.
scarcely (skārs′li), *adv.* seldom; rarely; with difficulty.
scarceness (skārs′nes), *n.* the state of being scarce. Also scarcity.
scare (skār), *v.t.* to strike with sudden terror; frighten: *n.* a sudden fright or panic.
scarecrow (skār′krō), *n.* anything fantastic set up in fruit gardens, &c., to scare away birds; a vain cause of terror.
scarf (skärf), *n.* a light handkerchief or tie for the neck; sash: *v.t.* throw on loosely; dress with a scarf; to unite (two pieces of timber) at the ends by a kind of dovetail.
scarfing (skärf′ing), *n.* the formation of a beam out of two pieces of timber.
scarfskin (skärf′skin), *n.* the cuticle.
scarification (skar-i-fi-kā′shun), *n.* the act of scarifying.
scarificator (skar′i-fi-kā-tēr), *n.* a surgical instrument used in scarifying.
scarifier (skar′i-fī-ēr), *n.* one who, or that which, scarifies; a scarificator; an agricultural instrument for stirring the soil.
scarify (skar′i-fī), *v.t.* [*p.t.* & *p.p.* scarified, *p.pr.* scarifying], to scratch or cut; make small incisions in by a lancet or scarificator: as, to *scarify* the skin; to stir up and prepare for planting: as, to *scarify* the soil.
scarlatina (skär-la-tē′na), *n.* scarlet fever of a mild form.
scarlet (skär′let), *n.* a bright red color; cloth of such a color: *adj.* of a scarlet color.
scarlet fever (fē′vēr), *n.* contagious febrile disease characterized by a scarlet eruption.
scarlet runner (run′ēr), *n.* a variety of bean.
scarp (skärp), *n.* a slope or declivity, nearly perpendicular; the slope of a ditch at the foot of a parapet: *v.t.* to cut perpendicularly or nearly so.
scary (skā′ri), *adj.* causing, or subject to, sudden fright.
scathe (skāth), *v.t.* to injure or hurt: *n.* injury or harm.
scathing (skāth′ing), *adj.* injurious; hurtful; very severe or bitter.
scatter (skat′ēr), *v.t.* to strew or throw loosely about; disperse or dissipate: *v.i.* to be dispersed or dissipated.
scatterbrained (skat′ēr-brānd), *adj.* giddy.
scaup (skawp), *n.* a species of duck.
scavage (skav′āj), *n.* garbage, muck, and street dirt of all sorts which a scavenger removes.
scavenge (skav′enj), *v.t.* to cleanse, as streets, from mud and filth.

Duplicates KEYs are not allowed

The latest / last KEY wins. (Link)

It will overwrite the previous KEY and VALUE pair



Python 3.6
(known limitations)

```
#dictionary of fruits

fruits_dictionary = {

    "a" : "apple",
    "b" : "banana",
    "b" : "berry",
    "g" : "grapes"
}

print(fruits_dictionary)
```

Edit this code

Print output (drag lower right corner to resize)

```
{'a': 'apple', 'b': 'berry', 'g': 'grapes'}
```

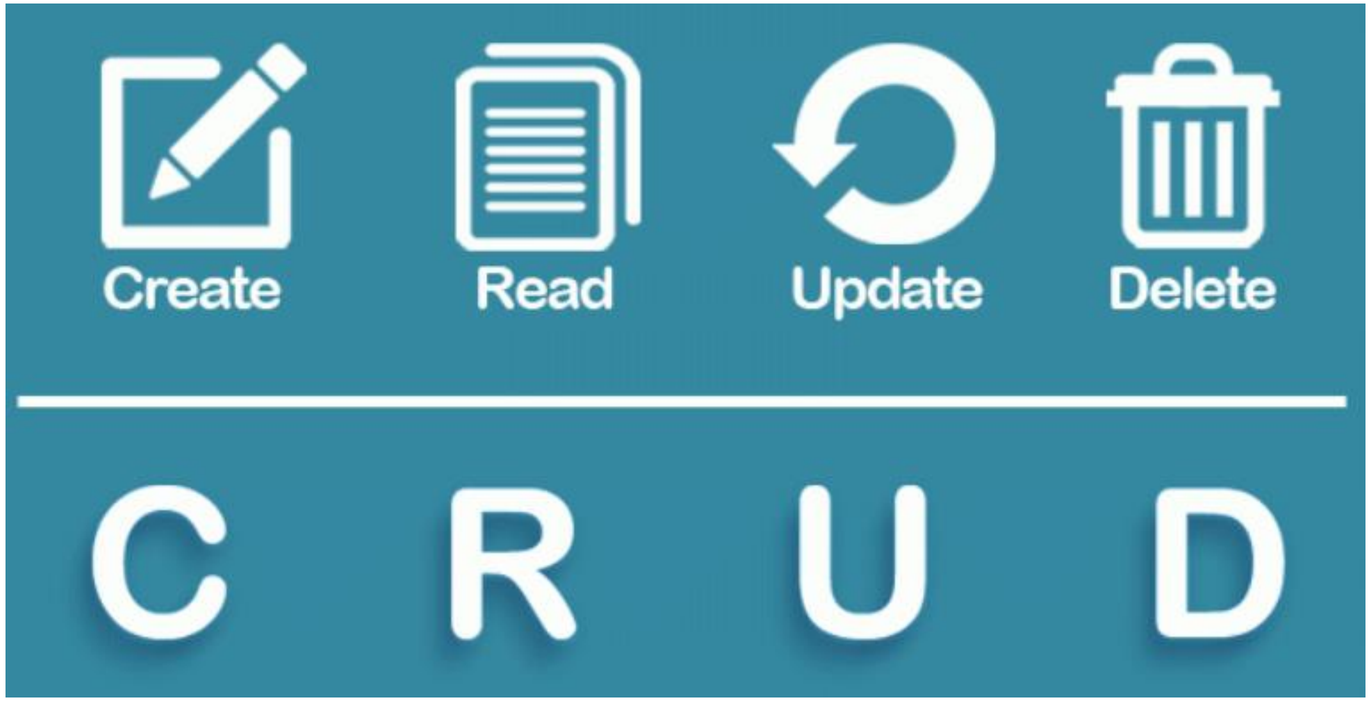
CRUD of Dictionary

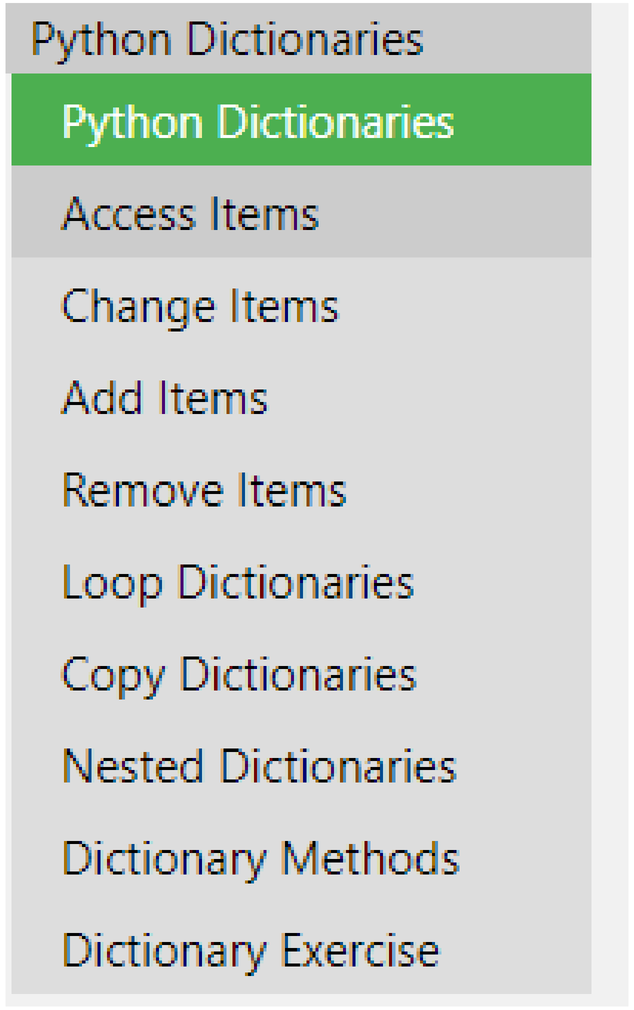
C – Create (Add, Insert, Append, Extend, Copy)

R – Read (Query, Traversal, Find, Search)

U – Update (Modify, Change, Edit)

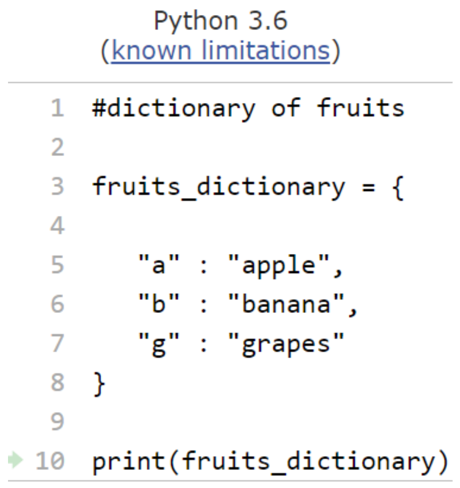
D – Delete (Remove, Empty)





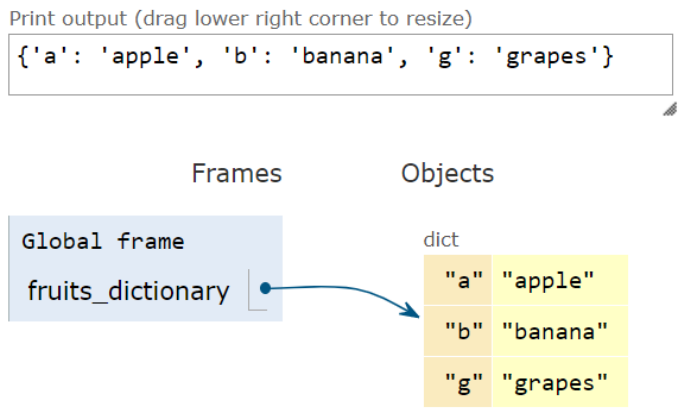
Creating a dictionary (link)

We use CURLY Brackets to indicate a dictionary. And we separate the keys and values using : (colon)



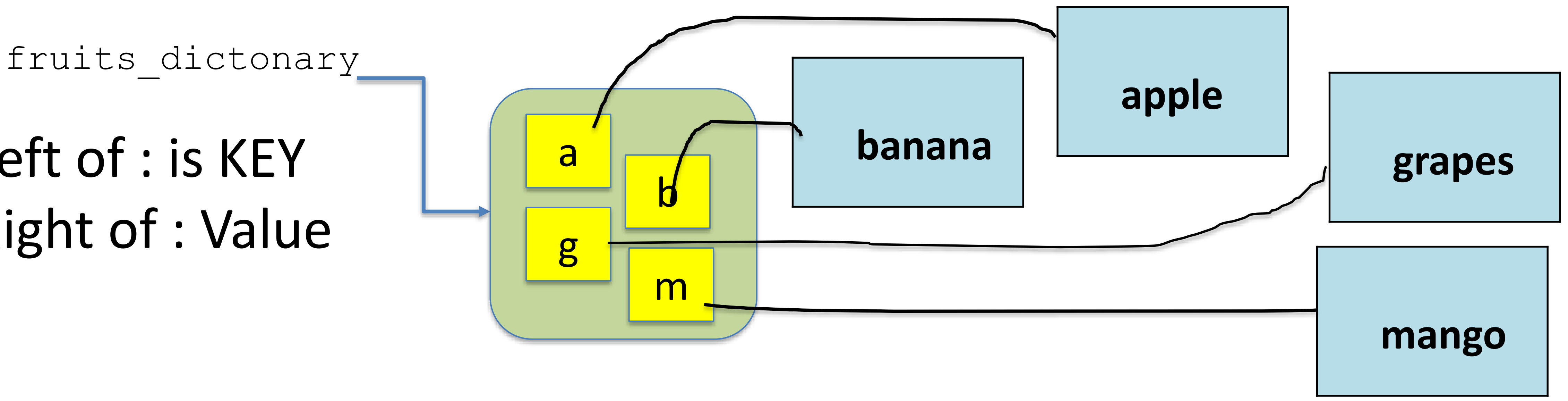
Python 3.6
(known limitations)

```
#dictionary of fruits

fruits_dictionary = {

    "a" : "apple",
    "b" : "banana",
    "g" : "grapes"
}

print(fruits_dictionary)
```



Print output (drag lower right corner to resize)

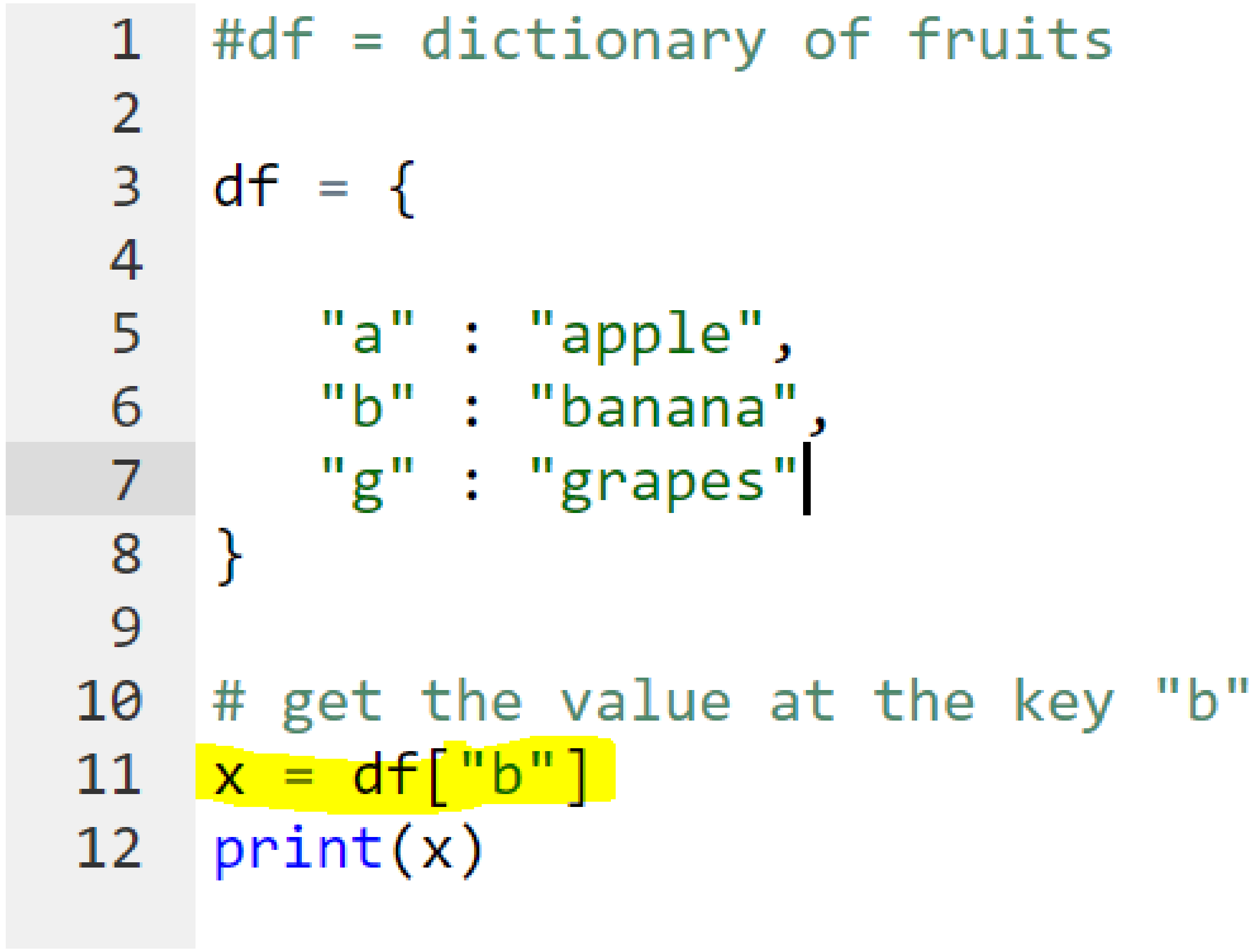
```
{'a': 'apple', 'b': 'banana', 'g': 'grapes'}
```



Left of : is KEY
Right of : Value

Reading/Accessing an item from a dictionary (link)

We use **subscript** notation and use the **KEY** to access the value



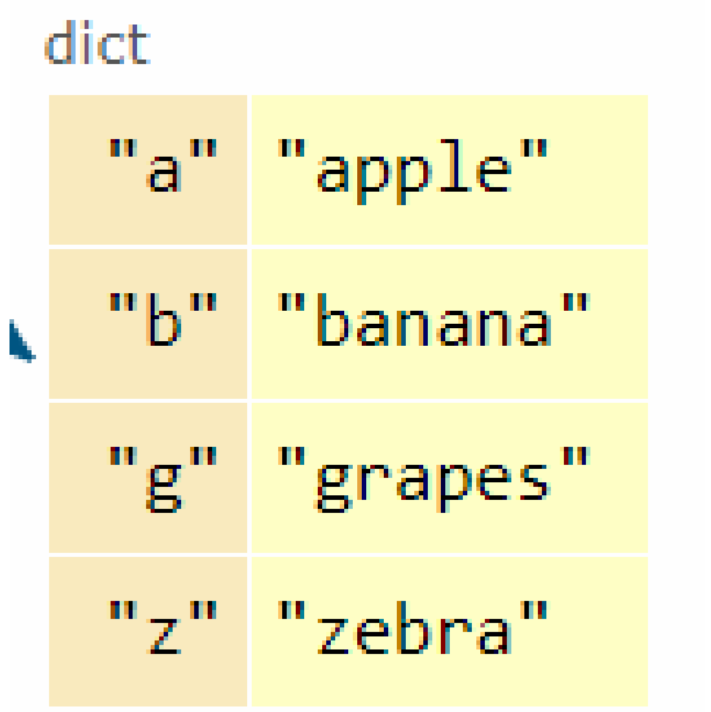
```
#df = dictionary of fruits

df = {

    "a" : "apple",
    "b" : "banana",
    "g" : "grapes"
}

# get the value at the key "b"
x = df["b"]
print(x)
```

Accessing all KEYS, VALUES and ITEMS (link)

https://www.w3schools.com/python/python_dictionaries_access.asp

keys( ) returns all the keys
values( ) method returns all the values
items( ) method returns all (key-value) pairs

d.keys() → [‘a’, ‘b’, ‘g’, ‘z’]
d.values() → [‘apple’, banana’, ‘grapes’, ‘zebra’]
d.items() →[(‘a’, ‘apple’), (‘b’, ‘banana’…]



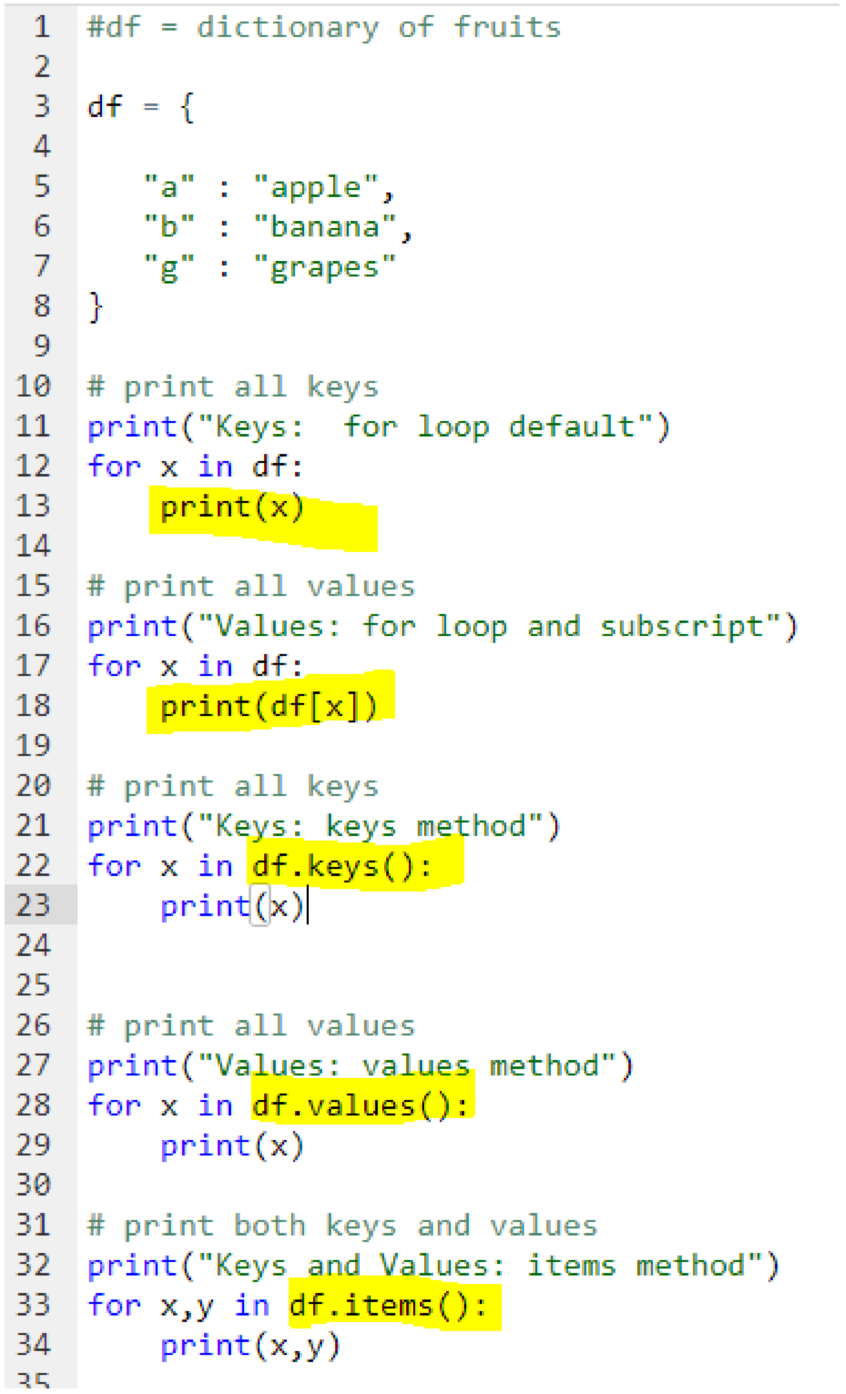
dict

"a"	"apple"
"b"	"banana"
"g"	"grapes"
"z"	"zebra"

Iterating the dictionary (link)

https://www.w3schools.com/python/python_dictionaries_loop.asp

1. Read all KEYS
2. Read all VALUES
3. Read both KEYS and VALUES

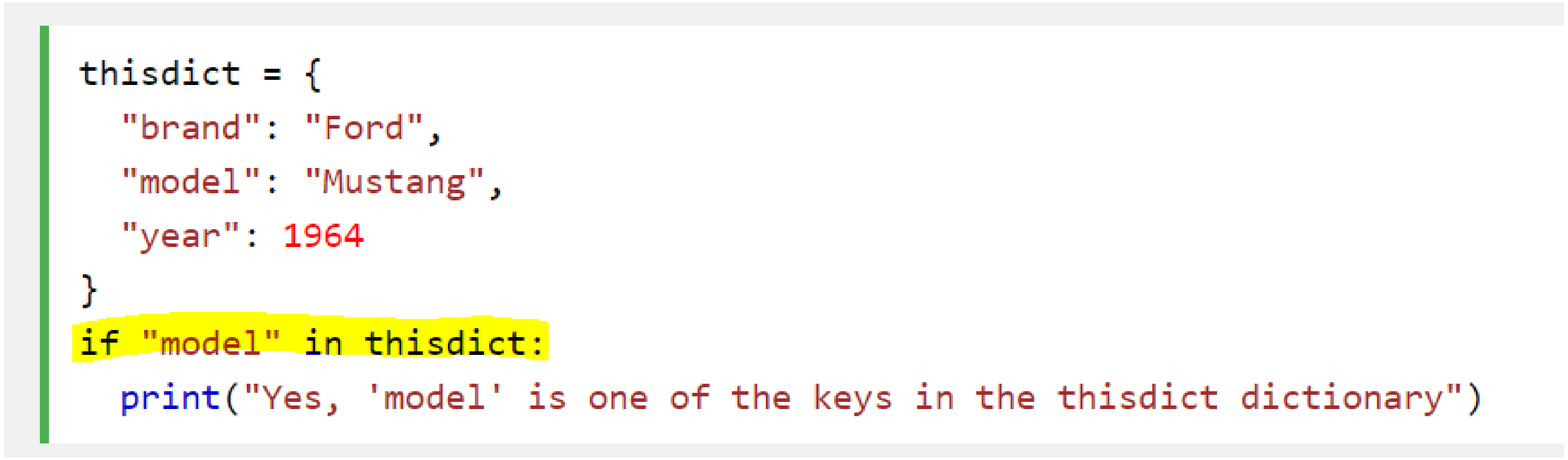


```python
#df = dictionary of fruits

df = {

   "a" : "apple",
   "b" : "banana",
   "g" : "grapes"
}

# print all keys
print("Keys:  for loop default")
for x in df:
    print(x)

# print all values
print("Values: for loop and subscript")
for x in df:
    print(df[x])

# print all keys
print("Keys: keys method")
for x in df.keys():
    print(x)


# print all values
print("Values: values method")
for x in df.values():
    print(x)

# print both keys and values
print("Keys and Values: items method")
for x,y in df.items():
    print(x,y)
```

Checking for the memberships? **in**

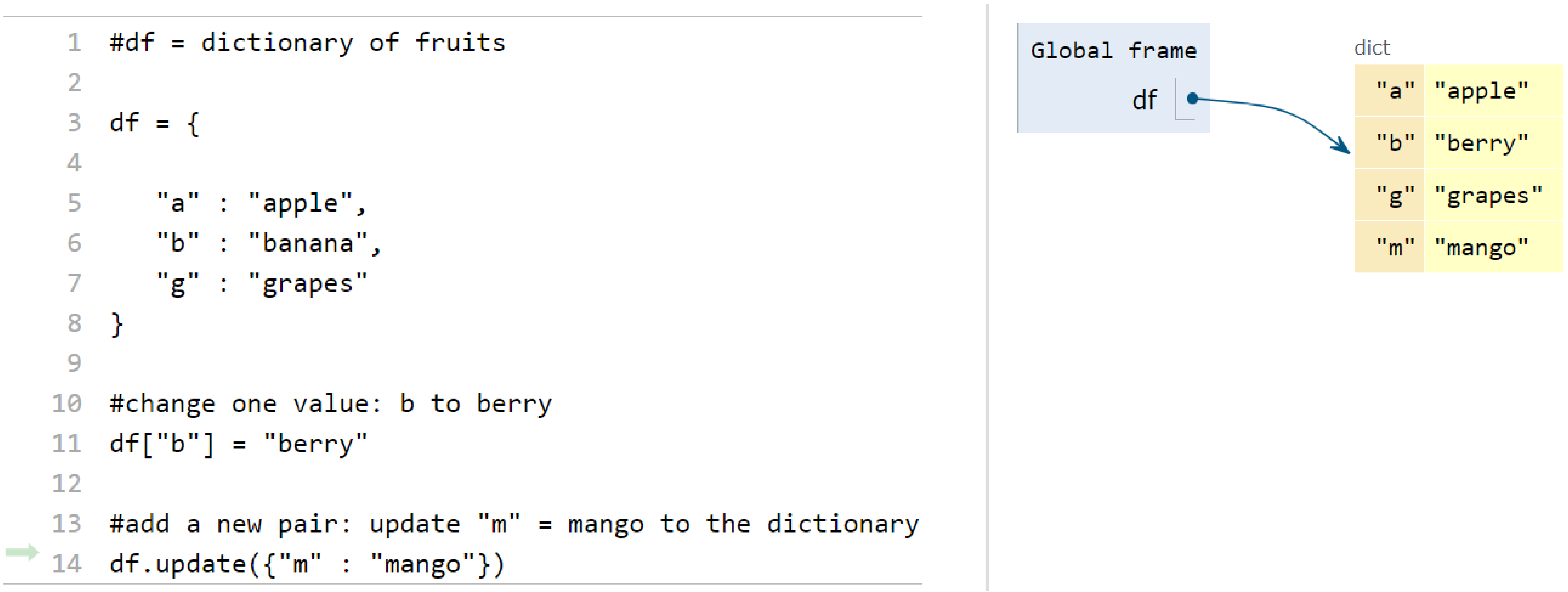
The dictionary uses the KEYS for checking the membership (in) (not in)

Is KEY there? Update it.
Is KEY not there? Insert it.



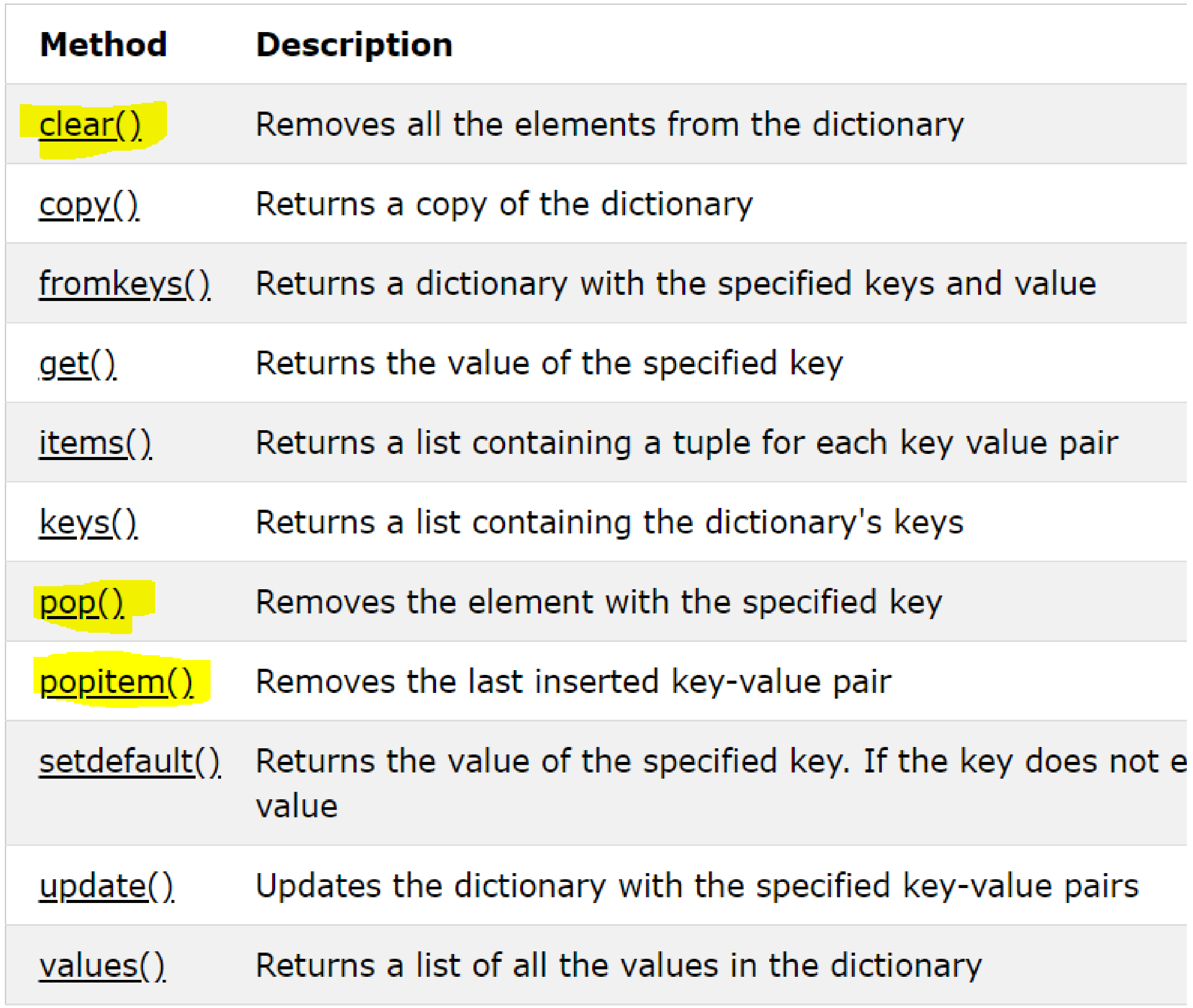
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Updating the dictionary (link)



```python
#df = dictionary of fruits

df = {

    "a" : "apple",
    "b" : "banana",
    "g" : "grapes"
}

#change one value: b to berry
df["b"] = "berry"

#add a new pair: update "m" = mango to the dictionary
df.update({"m" : "mango"})
```

Deleting items from the dictionary

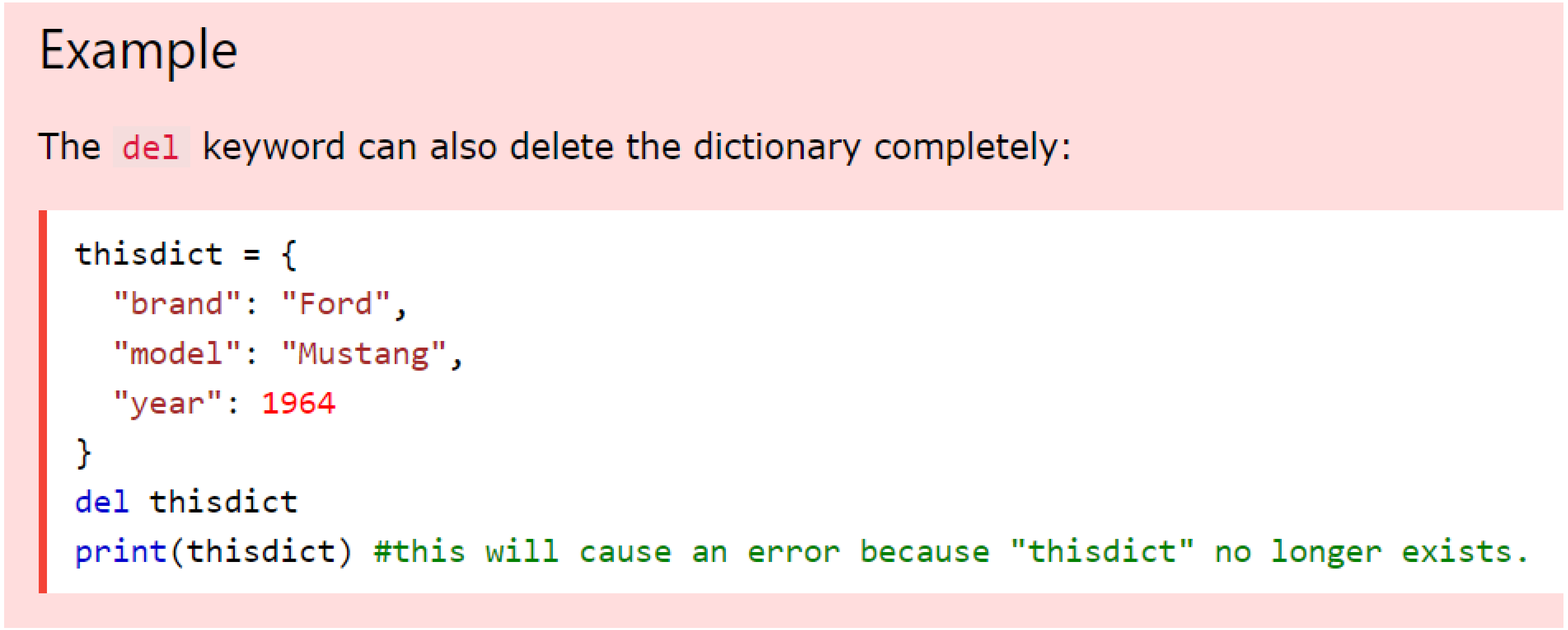


Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not e value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

https://www.w3schools.com/python/python_dictionaries_methods.asp

Deleting the entire dictionary

We use the python built in function to delete the entire dictionary.

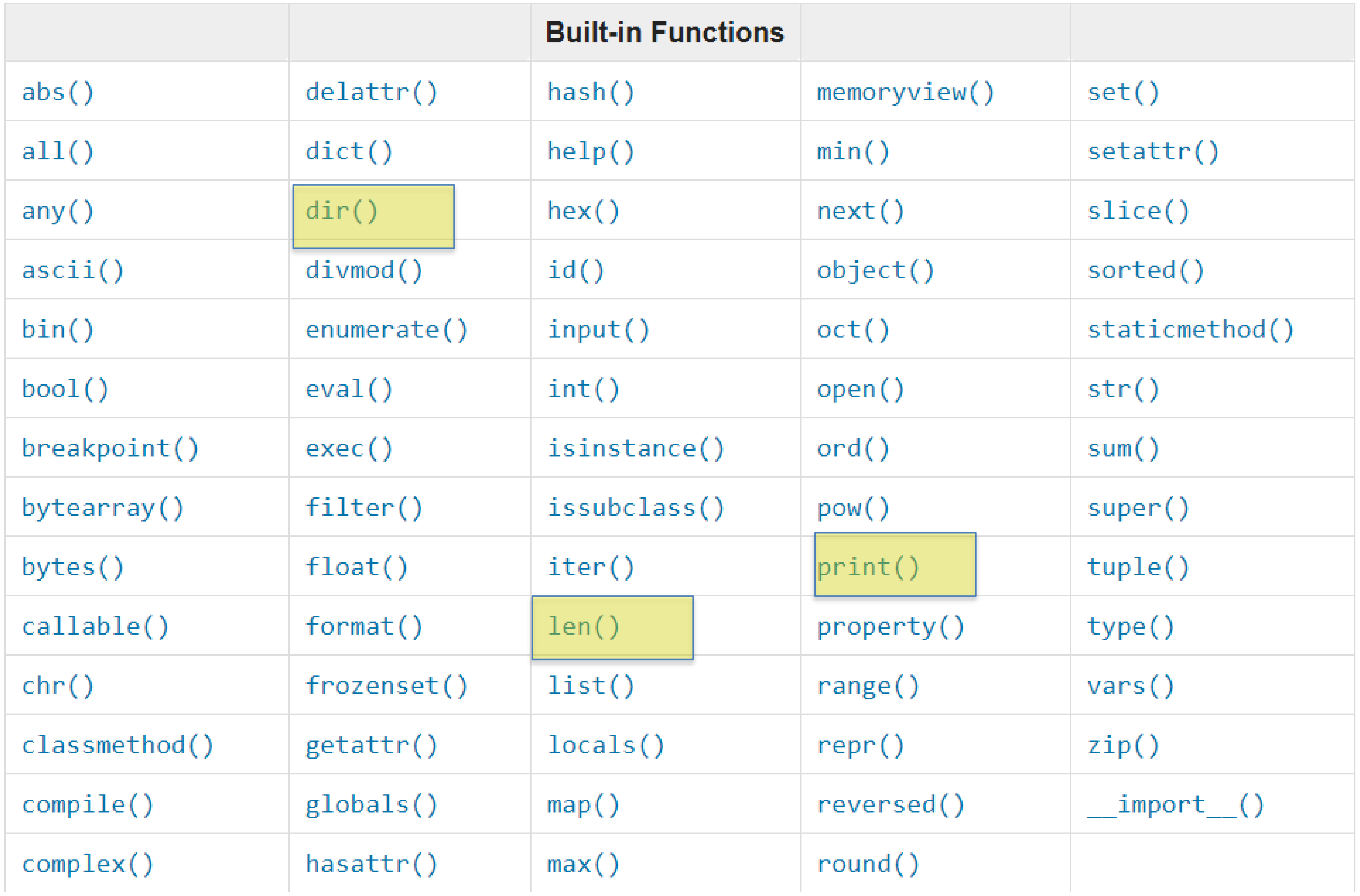


Example

The `del` keyword can also delete the dictionary completely:

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

Python’s Built-in Functions



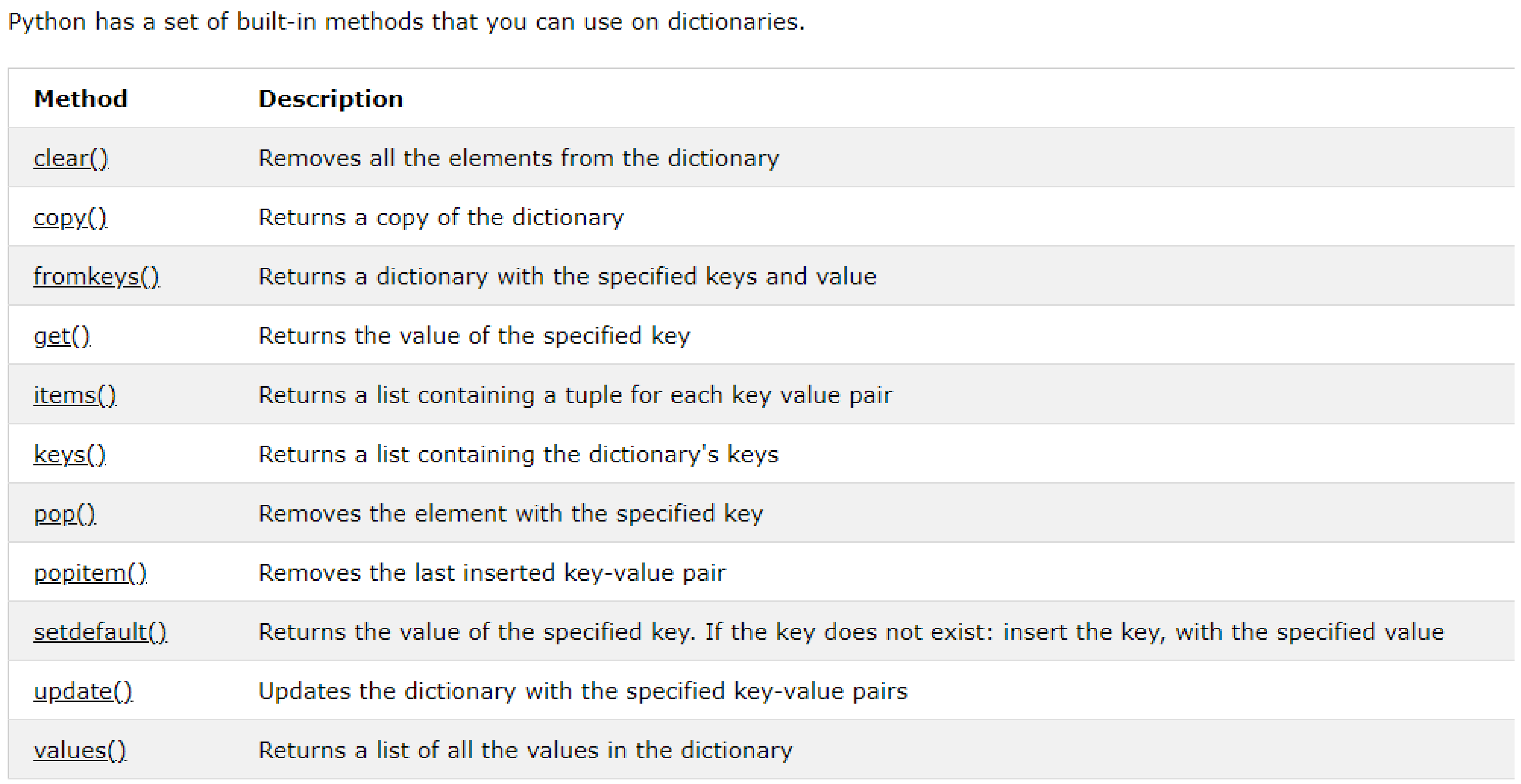
		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

What functions are valid for dictionaries?

https://docs.python.org/3/library/functions.html

dictionary methods

https://www.w3schools.com/python/python_ref_dictionary.asp



Python has a set of built-in methods that you can use on dictionaries.

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
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Dictionary summary

Unordered collection.
Does not contain duplicates.
Supports mapping (key-value pairs) (name-value pairs)

Except LIST and DICTIONARY, all others can be used as a KEY.
Any data type can be used as a VALUE.

Very efficient in quick looks ups through easy to remember keys.

Lists vs Tuples vs Sets vs Dictionary

	Lists	Tuples	Set	Dictionary
Ordered	✓	✓	✗	✗
Indexed	✓	✓	✗	✗
Add or Update items	✓	✗	✓	✓
Can contain duplicates	✓	✓	✗	✗
Supports Keys (Name: Values)	✗	✗	✗	✓
Uses	Square Brackets	Round Brackets	Curly Brackets	Curly Brackets
	[ ] list()	**( ) tuple()**	**{ } set ()**	**{ } dict( )**