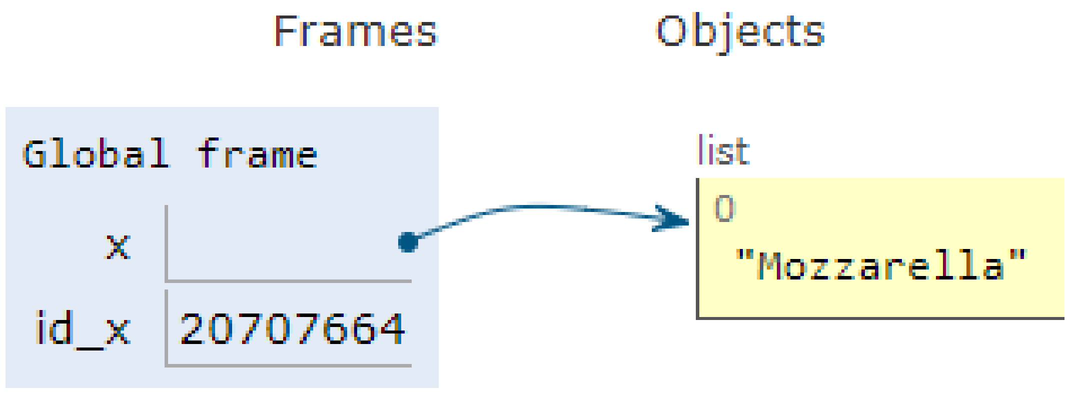
Types and Mutability

Basic Terminology

- *object* – data stored in a program
- each object has three attributes:
    1) identity – id() function
    2) type
    3) value

```
>>> x = [‘Mozzarella’]
>>> id_x = id(x)
>>> type(x)
<type 'list'>
  TypeError: unhashable type: 'list'
```



Standard Data Types

(1) number

```
>>> x = 5
```

(2) string

```
>>> x = “Mozzarella”
```

(3) list

```
>>> x = [“Mozzarella”]
```

(4) dictionary

```
>>> x = {1:”Mozzarella”, 2: “Brie”}
```

(5) set

```
>>> x = {“Mozzarella”}
```

(6) tuple

```
>>> x = (“Mozzarella”, )
```

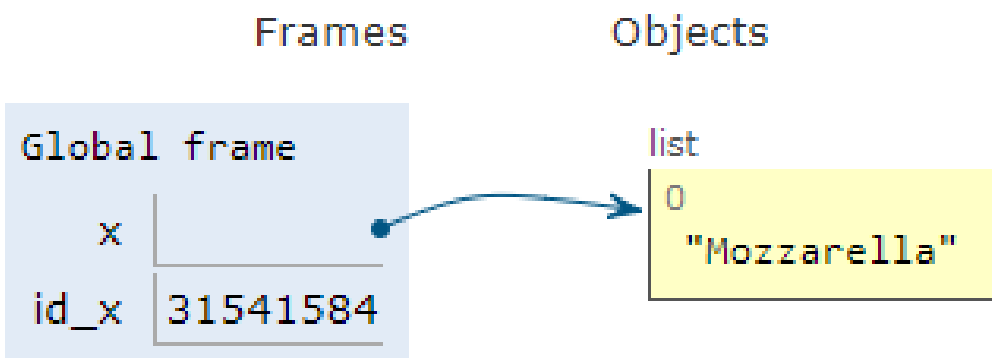
Object Mutability

- mutable objects can change value in-place (same id)
- example(s) of mutable objects: lists and dictionaries
- immutable objects: cannot change value, need to change reference to new value (new id)
- example(s) of immutable objects: strings, tuples, integers

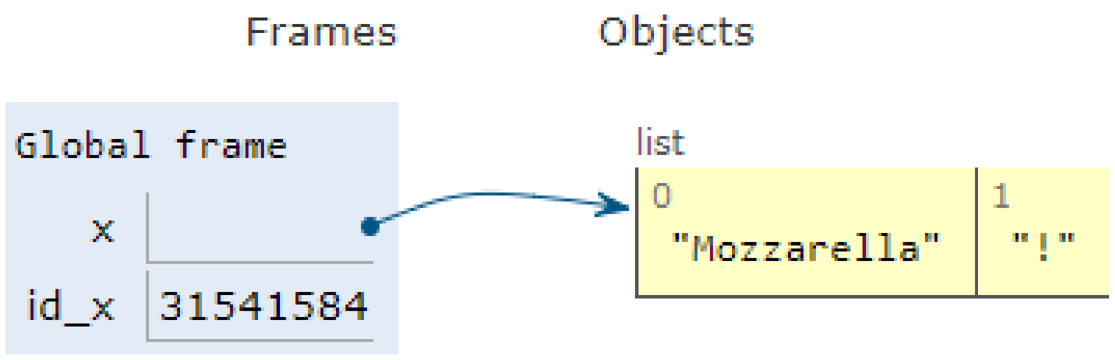
A Mutable Object

- a list is a mutable object

```
>>> x = ["Mozzarella"]
>>> id_x = id(x)
```



```
>>> x.append("!")
>>> id_x = id(x)
```

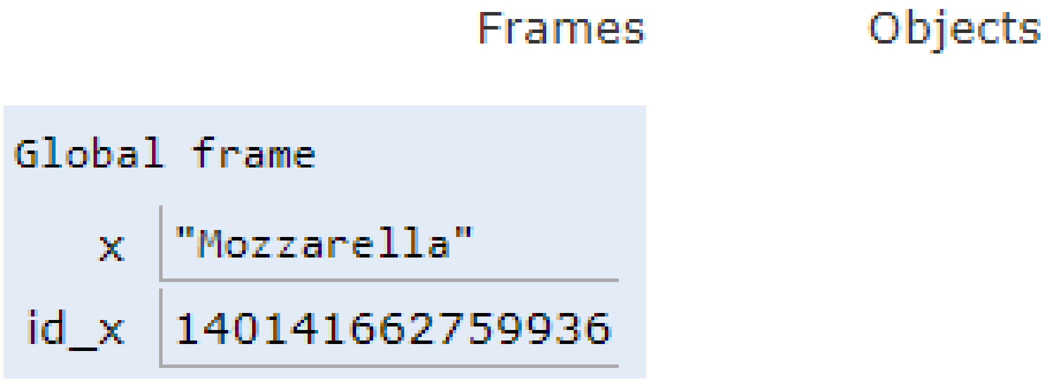


- changed object, same id

An Immutable Object

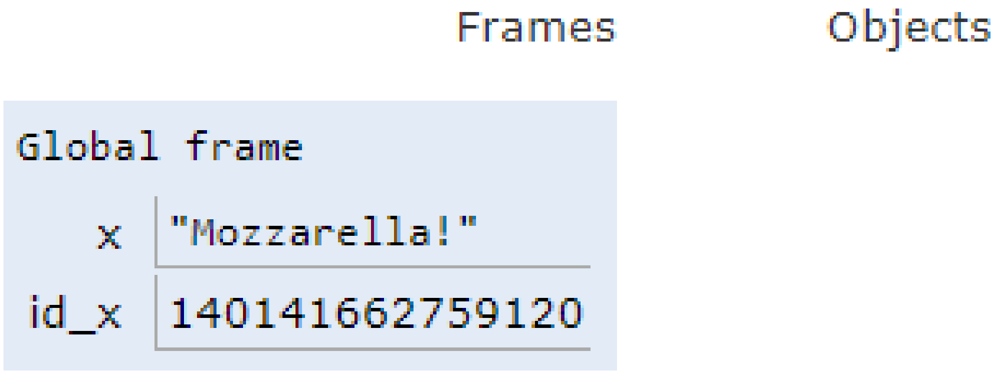
- a string is an immutable object

```
>>> x = "Mozzarella"
>>> id_x = id(x)
```



```
x     "Mozzarella"
id_x  140141662759936
```

```
>>> x.append("!")
>>> id_x = id(x)
```



```
x     "Mozzarella!"
id_x  140141662759120
```

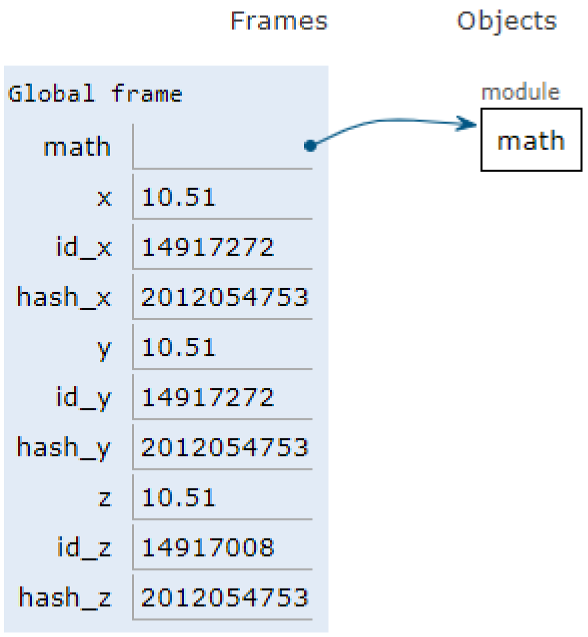
- changed object, different id

hash() **Function**

- map data to a hash value
- used for indexing and detection of duplicates
- same hash for same value (not same object)
- intuitively, want hashing for faster search and comparison of complex objects

Hashing and Object Id

```
>>> import math
>>> x = 10.51
>>> id_x = id(x)
>>> hash_x = hash(x)
>>> y = x
>>> id_y = id(y)
>>> hash_y = hash(y)
>>> z = round(10.51111,2)   # 10.51
>>> id_z = id(z)
>>> hash_z = hash(z)
```

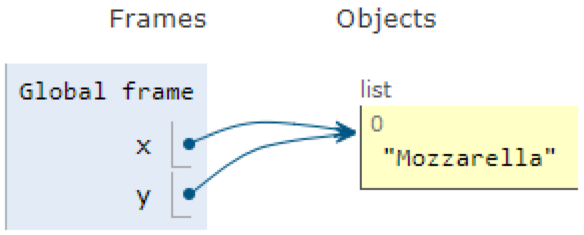


```
math
x       10.51
id_x    14917272
hash_x  2012054753
y       10.51
id_y    14917272
hash_y  2012054753
z       10.51
id_z    14917008
hash_z  2012054753
```

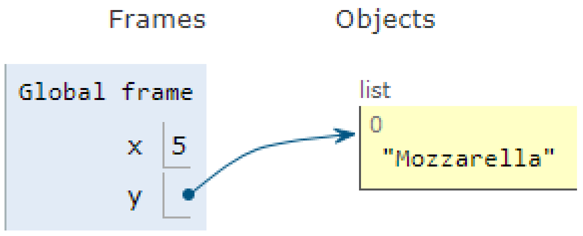
Variable References: *id*()

- check if variables point to same object

```
>>> x = ["Mozzarella"]
>>> y = x
```



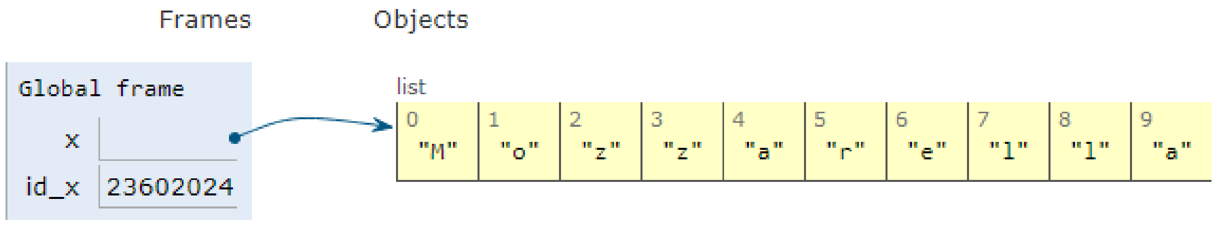
```
>>> id(x), id(y)
>>> (38803792, 38803792)
>>> x = 5
```



```
>>> id(x), id(y)
>>> (37104648, 38803792)
```

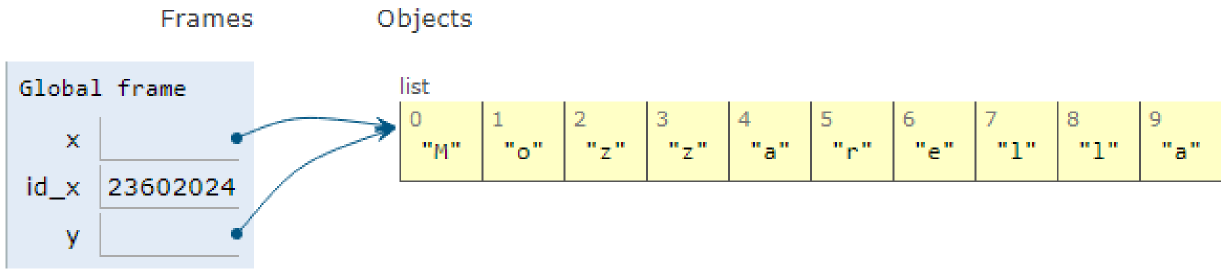
Variable Binding

```
>>> x = ['Mozzarella']
>>> id_x = id(x)
```



```
>>> y = x
>>> id_y = id(y)
```

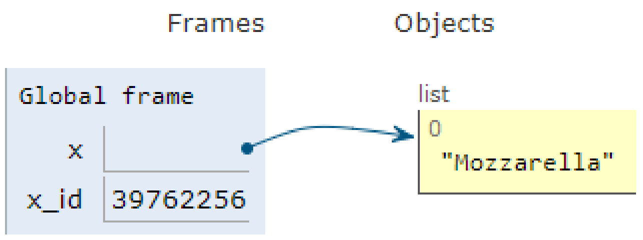
- expect y to have its own copy and id
- not in Python !!!



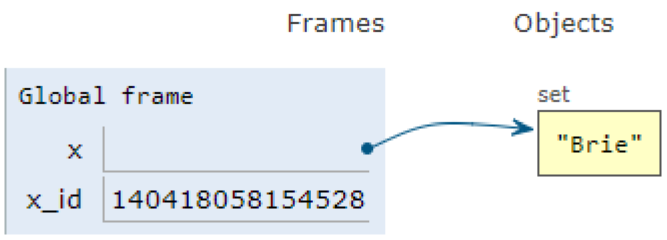
Variable Rebinding

- Python variables are just tags
- not bound to a physical location

```
>>> x = ['Mozzarella'] ; x_id = id(x)
```



```
>>> x = {'Brie'};  x_id = id(x)
```



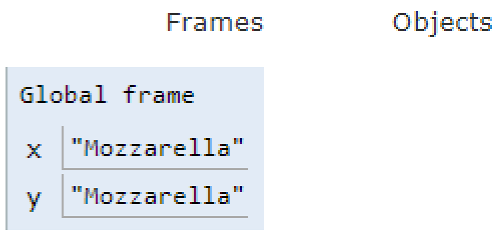
old object is garbage collected X

id: 397622256 [‘Mozzarella’] type: list	id: 140418058154528 {‘Brie’} type: set

Shared References

- immutable objects:

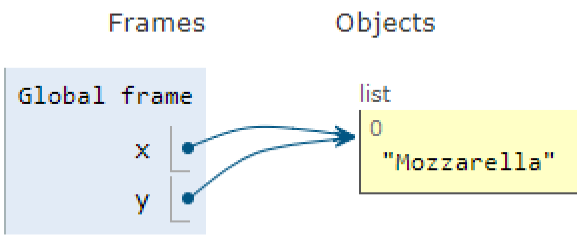
```
>>> x = "Mozzarella"
>>> y = x
```



```
>>> id(x), id(y)
(140413023247360,  140413023247360)
```

- mutable objects:

```
>>> x = ["Mozzarella"]
>>> y = x
```



```
>>> id(x), id(y)
(38803792, 38803792)
```

Hashing and Mutability

- a hashable object has a hash value
- hash value never changes
- immutable objects: bool, bytes, complex, decimal, float, frozenset, int, range, string, tuple – can be hashed

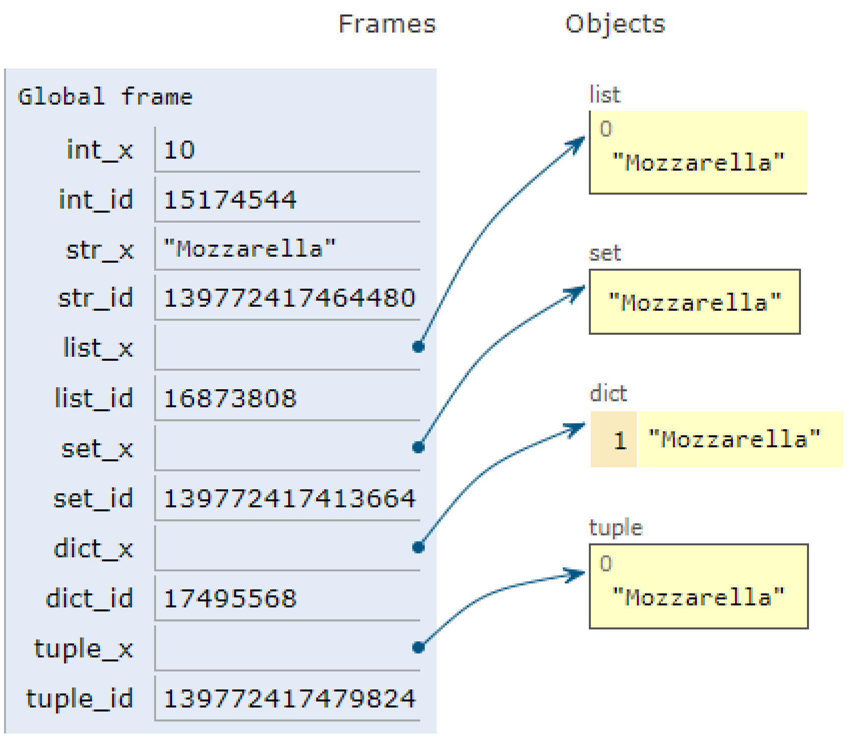
```
>>> x = ‘Mozzarella’
>>> hash(x)
1070568253
```

- mutable objects: bytearray, dict, list, set, user-defined classes

```
>>> x = [‘Mozzarella’]
>>> hash(x)
    TypeError: unhashable type: 'list'
```

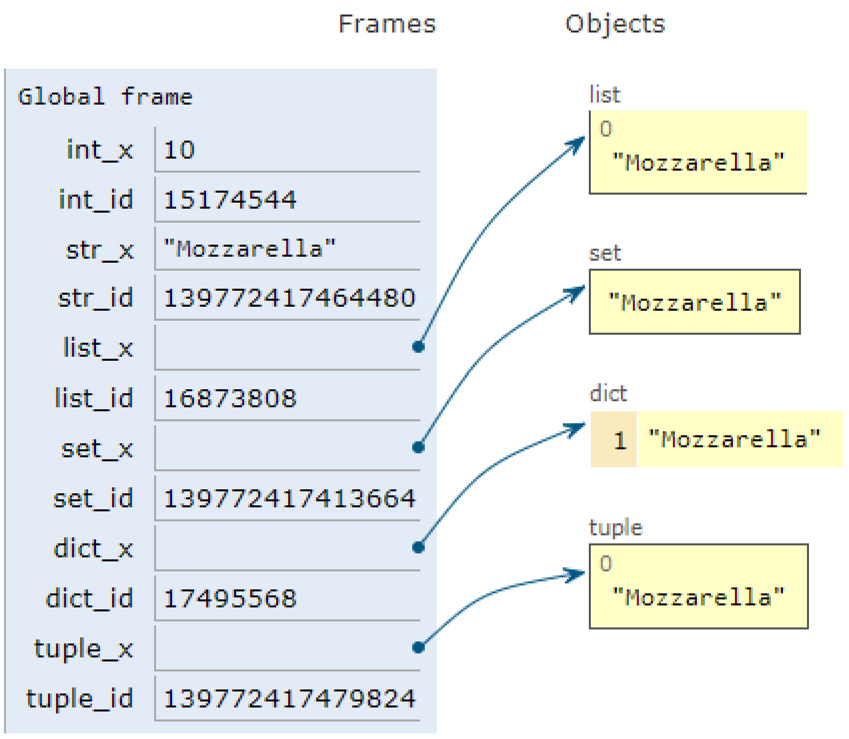
Checking Types Mutability

```
>>> int_x = 5
>>> string_x = “Mozzarella”
>>> list_x = [“Mozzarella”]
>>> set_x = {“Mozzarella”}‘
>>> dict_x = {1:”Mozzarella”}
>>> tuple_x = (“Mozzarella”, )
```



Type Mutability Results

```
>>> int_x = 5
>>> string_x = “Mozzarella”
>>> list_x = [“Mozzarella”]
>>> set_x = {“Mozzarella”}‘
>>> dict_x = {1:”Mozzarella”}
>>> tuple_x = (“Mozzarella”, )
```

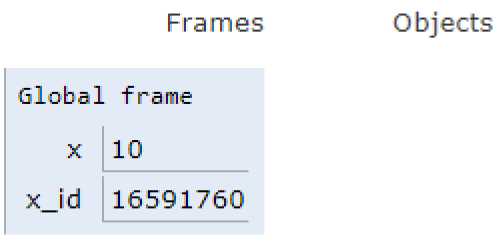


Global frame

int_x	10
int_id	15174544
str_x	"Mozzarella"
str_id	139772417464480
list_x	
list_id	16873808
set_x	
set_id	139772417413664
dict_x	
dict_id	17495568
tuple_x	
tuple_id	139772417479824

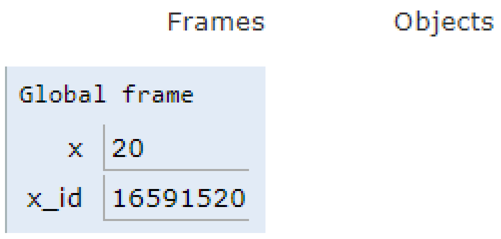
int Type Mutability

```
>>> x = 10
>>> x_id = id(x)
```



```
Global frame
   x    10
x_id    16591760
```

```
>>> x = x + 10
>>> x_id = id(x)
```

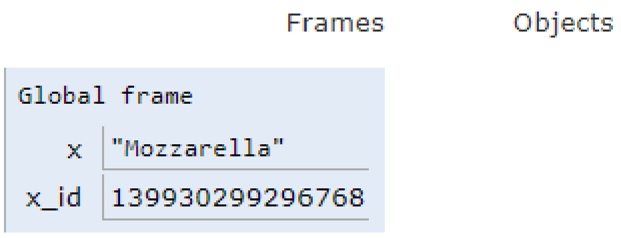


```
Global frame
   x    20
x_id    16591520
```

- integers are immutable

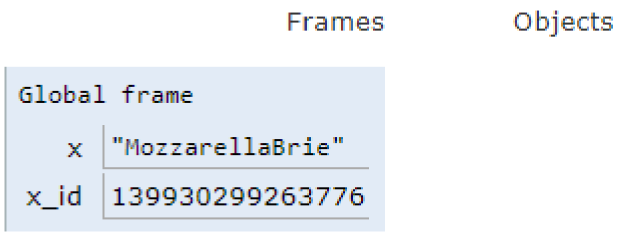
str Type Mutability

```
>>>  x = "Mozzarella"
>>> x_id = id(x)
```



```
x       "Mozzarella"
x_id    139930299296768
```

```
>>> x = x + "Brie"
>>> x_id = id(x)
```

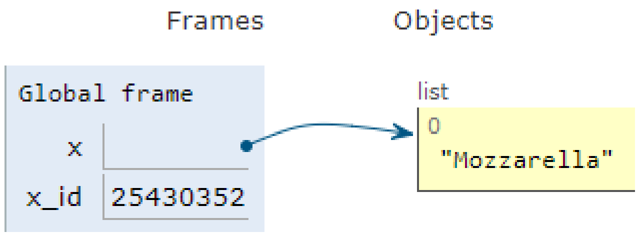


```
x       "MozzarellaBrie"
x_id    139930299263776
```

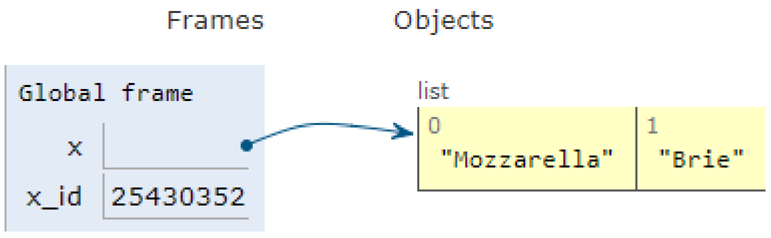
- strings are immutable

list Type Mutability

```
>>> x = [‘Mozzarella’]
>>> x_id = id(x)
```



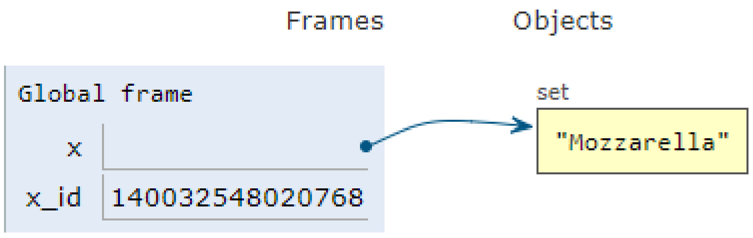
```
>>> x = x.append(‘Brie’)
>>> x_id = id(x)
```



- lists are mutable (same id)

set Type Mutability

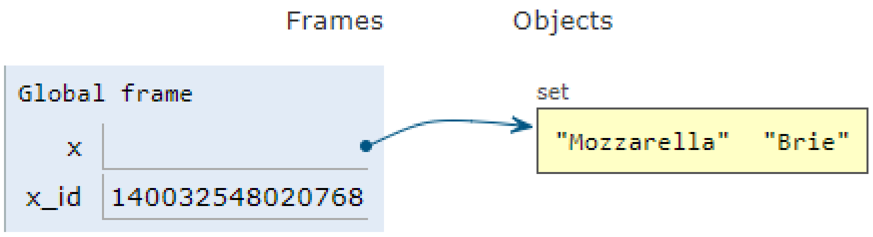
```
>>> x = {"Mozzarella"}
>>> x_id = id(x)
```



x_id 140032548020768

set: "Mozzarella"

```
>>>  x.update({'Brie'}
>>> x_id = id(x)
```



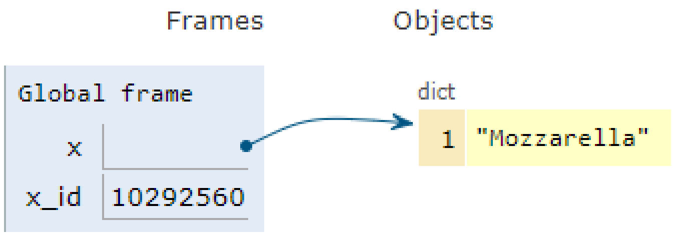
x_id 140032548020768

set: "Mozzarella" "Brie"

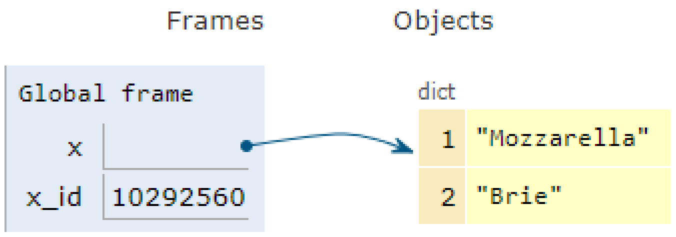
- sets are mutable (same id)

dict Type Mutability

```
>>>x = {1: ‘Mozzarella’}
>>> x_id = id(x)
```



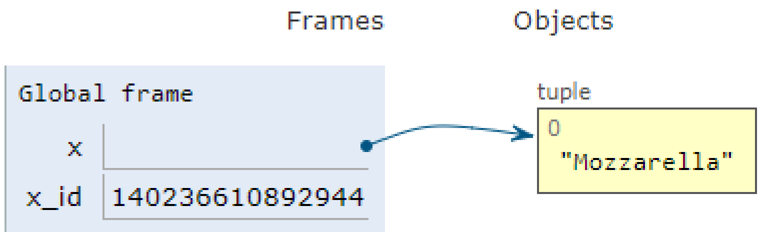
```
>>>x.update({2: ‘Brie’})
>>> x_id = id(x)
```



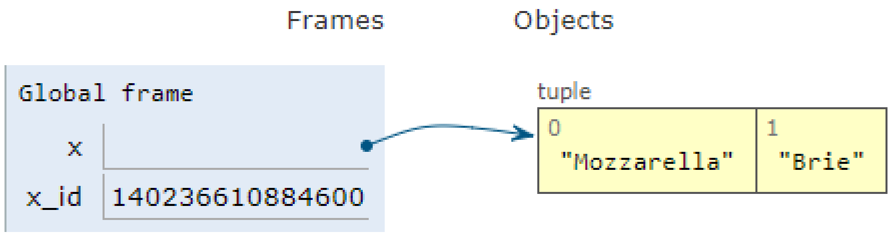
- dictionaries are mutable (same id)

tuple Type Mutability

```
>>> x = ("Mozzarella", )
>>> x_id = id(x)
```



```
>>> x = x + ("Brie" ,)
>>> x_id = id(x)
```



- tuples are immutable

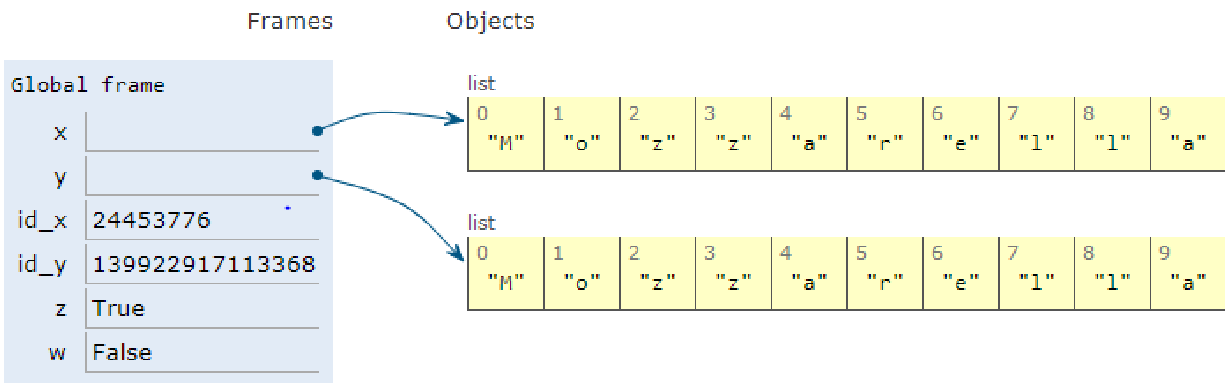
Summary of Type Mutability

type	**initial**	**last**	**mutable**
int	x = 10	x = x + 10	no
str	x = ’Mozzarella’	x = x + ‘Brie’	no
list	x = [‘Mozzarella’]	x.append(‘Brie’)	yes
set	x = [‘Mozzarella’]	x.update(‘Brie’)	yes
dict	x = {1:’Mozzarella’}	x.update({2: ’Brie’})	yes
tuple	x = (‘Mozzarella’,)	x = x + ("Brie’ ,)	no

- polymorphism (“many”, “forms”)
- same method, different types

Identity vs. Equality

- use ‘*is*’ to check for object equality
- use ‘==” for object identity



id_x	24453776
id_y	139922917113368
z	True
w	False

```
>>> x = list(‘Mozzarella’)
>>> y = x[:]
>>> id_x = id(x)
>>> id_y = id(y)
>>> z = (x == y)
>>> w = (y is x)
```

Review Problems

Interview Problem

- what are immutable built-in types?

Interview Problem

- what are mutable built-in types?