# Chapter 12

Dr. Raza Ul Mustafa Khokhar
American University
Computer Science Department - CSC-148

# Dictionaries

- Strings, lists, and tuples — are sequential collections
- This means that the items in the collection are ordered from left to right and they use integers as indices to access the values they contain

- Dictionaries are a different kind of collection. They are Python's built-in mapping type. A map is an unordered, associative collection. The association, or mapping, is from a key, which can be any immutable type, to a value, which can be any Python data object.

# Examples

As an example, we will create a dictionary to translate English words into Spanish. For this dictionary, the keys are strings and the values will also be strings.

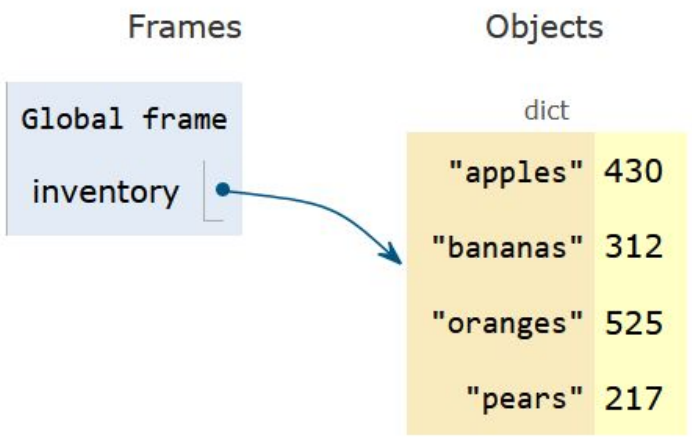
```
eng2sp = {}
eng2sp['one'] = 'uno'
eng2sp['two'] = 'dos'
eng2sp['three'] = 'tres'
```

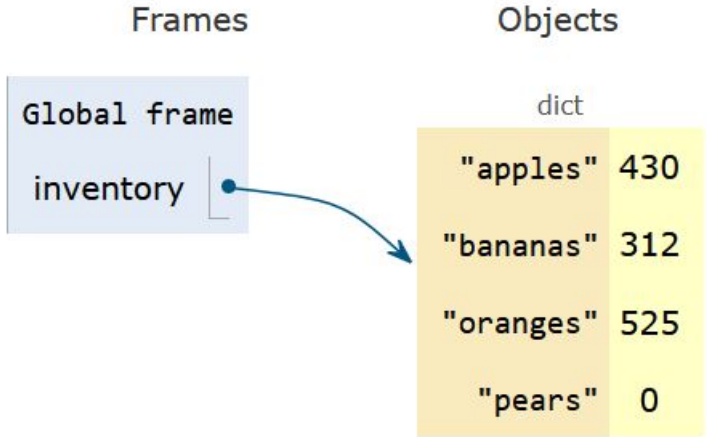
```
eng2sp = {'three': 'tres', 'one': 'uno', 'two': 'dos'}
```

# Mutable

- ***Dictionaries are also mutable***

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525,
'pears': 217}
inventory['pears'] = 0
```



# Dictionary Methods

| Method | Parameters | Description |
|---|---|---|
| keys | none | Returns a view of the keys in the dictionary |
| values | none | Returns a view of the values in the dictionary |
| items | none | Returns a view of the key-value pairs in the dictionary |
| get | key | Returns the value associated with key; None otherwise |
| get | key,alt | Returns the value associated with key; alt otherwise |

# Get keys – Make keys a list - Loop on Keys

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525,
'pears': 217}

for akey in inventory.keys():
   print("Got key", akey, "which maps to value", inventory[akey])

ks = list(inventory.keys())
print(ks)
```

# Values, Items and Keys

- The **values** and **items** methods are similar to **keys**. They return view objects which can be turned into lists or iterated over directly. Note that the items are shown as tuples containing the key and the associated value.

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

print(list(inventory.values()))
print(list(inventory.items()))

for (k,v) in inventory.items():
    print("Got", k, "that maps to", v)

for k in inventory:
    print("Got", k, "that maps to", inventory[k])
```

# The in and not in operators

- The in and not in operators can test if a key is in the dictionary:

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}
print('apples' in inventory)
print('cherries' in inventory)

if 'bananas' in inventory:
    print(inventory['bananas'])
else:
    print("We have no bananas")
```

# Get method - When there is no error if key is not present

- The get method allows us to access the value associated with a key, similar to the [ ] operator.
- The important difference is that get will not cause a runtime error if the key is not present. It will instead return None.
- However, a variation is there for return. In this case, since “cherries” is not a key, return 0 (instead of None).

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

print(inventory.get("apples"))
print(inventory.get("cherries"))

print(inventory.get("cherries", 0))
```

# Slides & Material

razaulmustafa.us/cs148/