Lecture 7

Dictionaries

Burkay Genç, Ahmet Selman Bozkır, and Selma Dilek

19/04/2023

PREVIOUS LECTURE

- tuples
- lists

TODAY

- introduce new compound data types
  - dictionaries

DICTIONARIES

How To Store Student Info

- So far, we can store using separate lists for every info

```
names = ['Ali', 'Ahmet', 'Fatma', 'Kezban']
grades = ['B', 'A+', 'A', 'A']
courses = [101, 102, 101, 201]
```

- a **separate list** for each item
- each list must have the **same length**
- info stored across lists at **same index**, each index refers to a different person

How To Update/Retrieve Student Info

```
names = ['Ali', 'Ahmet', 'Fatma', 'Kezban']
grades = ['B', 'A+', 'A', 'A']
courses = [101, 102, 101, 201]

def get_grade(student):
  i = names.index(student)
  grade = grades[i]
  course = courses[i]
  return (course, grade)


print(get_grade("Fatma"))
```

```
## (101, 'A')
```

- **messy** if have a lot of different info to keep track of
- must maintain **many lists** and pass them as arguments
- must **always index** using integers
- must remember to change multiple lists

Dictionaries

- Objects of type **dict** (short for dictionary) are like lists
  - except that we index them using **keys**
- Think of a dictionary as a **set of key/value pairs**
- Literals of type dict are enclosed in **curly braces**
- Each element is written as a *key* followed by a *colon* followed by a *value*

```
my_dict = {"burkay":41, "ahmet": 23, "ayşe":34}
print(my_dict["ahmet"])
```

```
## 23
```

```
print(my_dict["ayşe"])
```

```
## 34
```

Dictionaries

- The key or the value can be anything

```
months = {1:"January", 2:"February", 3:"March", 4:"April"}
married = {"burkay":True, "ahmet":False, "ayşe":False}

print("The third month is", months[3])
```

```
## The third month is March
```

```
for i in married:
  if married[i]:
    print(i, "is married.")
  else:
    print(i, "is not married.")
```

```
## burkay is married.
## ahmet is not married.
## ayşe is not married.
```

Dictionaries

- If the key is **not found**:

```
married = {"burkay":True, "ahmet":False, "ayşe":False}

print(married["hasan"])
```

```
## KeyError: 'hasan'
```

Mutability

- Dictionaries are mutable.
  - Add, change or delete key/value pairs.

Add Key

- To **add** a new key/value pair, just define it:

```
married["hasan"] = True
print(married)
```

```
## {'burkay': True, 'ahmet': False, 'ayşe': False, 'hasan': True}
```

Test Key Existence

- To **test existence** of a key, use *in*:

```
"burkay" in married
```

```
## True
```

```
"fatma" in married
```

```
## False
```

Delete Key

- To **delete** a key, use *del* function:

```
print(married)
```

```
## {'burkay': True, 'ahmet': False, 'ayşe': False, 'hasan': True}
```

```
del(married["burkay"])
print(married)
```

```
## {'ahmet': False, 'ayşe': False, 'hasan': True}
```

Change Value

- You can change an existing value

```
print(married)
```

```
## {'ahmet': False, 'ayşe': False, 'hasan': True}
```

```
married["hasan"] = False
print(married)
```

```
## {'ahmet': False, 'ayşe': False, 'hasan': False}
```

Get Value

- Use the `dictionary.get(key, val)` function to safely check the value of a key
  - If `key` exists, returns `dictionary[key]`
  - Else, returns `val`

```
testDict = {1:"a", 2:"b", 3:"c", 4:"d"}

testDict.get(2, "does not exist")
```

```
## 'b'
```

```
testDict.get(9, "does not exist")
```

```
## 'does not exist'
```

WARNING!

- `dictionary[i]` does not give you the i^{th} item in the dictionary
- It returns the value corresponding to key `i`
  - And, it can be anywhere

```
squares = {5:25, 4:16, 3:9, 2:4, 1:1}
squares[1]
```

```
## 1
```

```
squares[5]
```

```
## 25
```

Exercise

- Write a function that counts the number of letters in a string.

Exercise

- Write a function that counts the number of letters in a string.
- **strategy**
  - use a dictionary to keep record of letter counts
  - iterate over the string letter by letter
    - add or update letter count

Exercise

- Write a function that counts the number of letters in a string.

```
testString = "this is a long string used for testing purposes"

def letterCount(s):
  d = {}
  for c in s:
    if c in d:
      d[c] += 1   # Same as d[c] = d[c] + 1
    else:
      d[c] = 1
  return d

print(letterCount(testString))
```

```
## {'t': 4, 'h': 1, 'i': 4, 's': 7, ' ': 8, 'a': 1, 'l': 1, 'o': 3, 'n': 3, 'g': 3, 'r': 3, 'u': 2,
'e': 3, 'd': 1, 'f': 1, 'p': 2}
```

Get Items

- You can get all (key, value) pairs as a list of tuples using 'items()`.
- This allows you to convert a dictionary to a list.

```
d = letterCount(testString)
d2list = list(d.items())
print(d2list)
```

```
## [('t', 4), ('h', 1), ('i', 4), ('s', 7), (' ', 8), ('a', 1), ('l', 1), ('o', 3), ('n', 3), ('g', 3),
('r', 3), ('u', 2), ('e', 3), ('d', 1), ('f', 1), ('p', 2)]
```

```
print("Value for key", d2list[3][0], "is", d2list[3][1], ".")
```

```
## Value for key s is 7 .
```

Get Keys

- You can get a list of all keys in a dictionary:

```
d = letterCount(testString)
d.keys()
```

```
## dict_keys(['t', 'h', 'i', 's', ' ', 'a', 'l', 'o', 'n', 'g', 'r', 'u', 'e', 'd', 'f', 'p'])
```

- this returns a `dict_keys` object which is iterable:

```
for key in d.keys():
  print(key)
```

```
## t
## h
## i
## s
##
## a
## l
## o
## n
## g
## r
## u
## e
## d
## f
## p
```

Get Keys

- You can also convert a `dict_keys` to a list:

```
keys_list = list(d.keys())
print(keys_list)
```

```
## ['t', 'h', 'i', 's', ' ', 'a', 'l', 'o', 'n', 'g', 'r', 'u', 'e', 'd', 'f', 'p']
```

Get Values

- You can get a list of all values in a dictionary:

```
d.values()
```

```
## dict_values([4, 1, 4, 7, 8, 1, 1, 3, 3, 3, 3, 2, 3, 1, 1, 2])
```

- this returns a `dict_values` object which is iterable:

```
for value in d.values():
  print(value)
```

```
## 4
## 1
## 4
## 7
## 8
## 1
## 1
## 3
## 3
## 3
## 3
## 2
## 3
## 1
## 1
## 2
```

Get Keys

- You can also convert a `dict_values` to a list:

```
values_list = list(d.values())
print(values_list)
```

```
## [4, 1, 4, 7, 8, 1, 1, 3, 3, 3, 3, 2, 3, 1, 1, 2]
```

Keys and Values

- values
  - any type (**immutable and mutable**)
  - can be **duplicates**
  - dictionary values can be lists, even other dictionaries!
- keys
  - must be **unique**
  - **immutable** type (`int, float, string, tuple, bool`)
  - actually need an object that is *hashable*, but think of it as immutable as all immutable types are hashable in Python
  - careful with **float** type as a key
- **no order** to keys or values!

```
d = {4:{1:0}, (1,3):"twelve", 'const':[3.14,2.7,8.44]}
```

list vs dict

- list
  - **ordered** sequence of elements
  - look up elements by an integer index
  - indices have an **order**
  - index is an **integer**
- dict
  - **matches** “keys” to “values”
  - look up one item by another item
  - **no order** is guaranteed
  - key can be any **immutable** type

Exercise

- Write a function that takes a dictionary of shop items matched to their prices as a parameter, and returns the most expensive item and its price as a tuple `(item, price)`.

Exercise

- Write a function that takes a dictionary of shop items matched to their prices as a parameter, and returns the most expensive item and its price as a tuple `(item, price)`.

```
def find_the_most_expensive(d):
    max_price = -1
    max_item = None
    for key, value in d.items():
        if value > max_price:
            max_price = value
            max_item = key
    return (max_item, max_price)
```

- Test with `items = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}`

```
items = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}
print(find_the_most_expensive(items))
```

```
## ('item4', 55)
```

Copyright Information

These slides are a direct adaptation of the slides used for MIT 6.0001 course present (as of February 2020) on MIT OCW web site.

Original work by:

Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare. License: Creative Commons BY-NC-SA.

Adapted by and for:

Asst. Prof. Dr. Burkay Genç. MUH101 Introduction to Programming, Spring 2020. Hacettepe University, Computer Engineering Department.