Tuples, Dictionaries, Files

Patterns and exercises

Warm-up

Facts and logic exercises

Fact: Tuples are immutable

Select the code fragment(s) that will result in an error if *c* refers to a non-empty tuple

1. print(c[0])
2. c[0] = 5
3. c.pop()
4. c.reverse()
5. c[:1]
6. len(c)

7. for e in c:
   print(e)

Select the code fragment(s) that will result in an error if *c* refers to a non-empty tuple

1. print(c[0])
2. c[0] = 5 ◀
3. c.pop() ◀
4. c.reverse() ◀
5. c[:1]
6. len(c)

7. for e in c:
   print(e)

f.write(s) writes string *s* to a file pointed to by *f*

What will the contents of *cat.txt* look like after the following code is executed?

```
catlines = ["We looked!", "And we saw him!", "The Cat in the Hat!"]
cat_file = open("cat.txt", "w")
for line in catlines:
   cat_file.write(line)
cat_file.close()
```

1. "We looked!"
   "And we saw him!"
   "The Cat in the Hat!"

2. We looked!
   And we saw him!
   The Cat in the Hat!

3. We looked!And we saw him!The Cat in the Hat!

What will the contents of cat.txt look like after the following code is executed?

```
catlines = ["We looked!", "And we saw him!", "The Cat in the Hat!"]
cat_file = open("cat.txt", "w")
for line in catlines:
   cat_file.write(line)
cat_file.close()
```

1. "We looked!"
   "And we saw him!"
   "The Cat in the Hat!"

2. We looked!
   And we saw him!
   The Cat in the Hat!

3. We looked!And we saw him!The Cat in the Hat! ◀

`for line in f:`

at each iteration of the loop,

`line` contains a complete line of text in *`f`*

Will the following code print True or False?

```
def is_in_dictionary(dict_file, w):
    for line in dict_file:
        if w == line:
            return True
    return False

f = open("words.txt")
print (is_in_dictionary (f, "cat"))
```

words.txt

```
ape
apple
bed
cat
dog
mouse
```

Will the following code print True or False?

```
def is_in_dictionary(dict_file, w):
    for line in dict_file:
        if w == line:
            return True
    return False

f = open("words.txt")
print (is_in_dictionary (f, "cat"))
```

words.txt

```
ape
apple
bed
cat
dog
mouse
```

False

Dictionary: keys and values

Select expression(s) that produce a list containing one string.

```
price_to_names = {
   '$': ['Queen St. Cafe', 'Dumplings R Us', 'Deep Fried Everything'],
   '$$': ['Mexican Grill'],
   '$$$': ['Georgie Porgie'],
   '$$$$': []}
```

- price_to_names['$']
- price_to_names['$$']
- price_to_names['$$$']
- price_to_names['$$$$']

Select expression(s) that produce a list containing one string.

```
price_to_names = {
   '$': ['Queen St. Cafe', 'Dumplings R Us', 'Deep Fried Everything'],
   '$$': ['Mexican Grill'],
   '$$$': ['Georgie Porgie'],
   '$$$$': []}
```

- price_to_names['$']
- price_to_names['$$'] ◀
- price_to_names['$$$'] ◀
- price_to_names['$$$$']

What does in operator check when applied to a dictionary?

Select expressions which evaluate to True

d = {"cherry": "red", "pomegranate": "red", "blueberry": "blue"}

- "blueberry" in d
- "apple" in d
- "red" in d
- "r" in d["cherry"]
- "r" in d["blueberry"]

Select expressions which evaluate to True

d = {"cherry": "red", "pomegranate": "red", "blueberry": "blue"}

- "blueberry" in d ◀
- "apple" in d
- "red" in d
- "r" in d["cherry"] ◀
- "r" in d["blueberry"]

In a dictionary: keys are always immutable, values can be of any type

Which are valid dictionaries?

- d1 = {"1-3": [1,2,3], "3-5": [3,4,5]}
- d2 = {[1,2,3]: "1-3", [3,4,5]: "3-5"}
- d3 = {(1,2,3): "1-3", (3,4,5): "3-5"}
- d4 = {(1,2,3): "1-3", (1,2,3): "3-5"}
- d5 = {(1,2,3): (1,2,3), (3,4,5): (3,4,5)}

Which are valid dictionaries?

- d1 = {"1-3": [1,2,3], "3-5": [3,4,5]} ◀
- d2 = {[1,2,3]: "1-3", [3,4,5]: "3-5"}
- d3 = {(1,2,3): "1-3", (3,4,5): "3-5"} ◀
- d4 = {(1,2,3): "1-3", (1,2,3): "3-5"}
- d5 = {(1,2,3): (1,2,3), (3,4,5): (3,4,5)} ◀

Prerequisites for the next lesson

- To learn how to write and run real programs we need:
- A decent text editor to write Python code.
  - Text editor is an app that allows you to easily edit and save simple text files.
  - Recommended editors:
    - Notepad++
    - Sublime Text

- Python 3X interpreter properly installed
  - Here are instructions for windows and for mac
  - To check: open a command prompt/terminal and type *`python`* (or *`python3`* for mac) – this should open Python shell. To exit shell type `exit()`
  - If you are unable to do this, please use help from tutors or ITS

Dictionary patterns

Main tasks performed using dictionaries

I. Loop over a dictionary

d = {"Ann":88, "David":66, "Cat":77}

- Find max student grade
- Find the name of a student with the lowest grade

II. Reverse dictionary: keys ↔ values

```
d = {"Ann":88, "David":66, "Cat":77}


r = {}
for key, val in d.items():
    r[val] = key
```

II. Reverse dictionary: problem

```
d = {"Ann":88, "David":66, "Cat":77, "Carl":66}


r = {}
for key, val in d.items():
    r[val] = key
```

II. Reverse dictionary: solution

```
d = {"Ann":88, "David":66, "Cat":77, "Carl":66}
```

- The values are not guaranteed to be unique
- Then for each non-unique value store a list of keys

```
r = {}
for name, grade in d.items():
    names_list = [ ]
    If grade in r:
        names_list = r[grade]
    names_list.append(name)
    r[grade] = names_list
```

II. Reverse dictionary: solution

```
d = {"Ann":88, "David":66, "Cat":77, "Carl":66}
```

- Or using dictionary.get() with default

```
r = {}
for key, val in d.items():
    new_val = r.get(val, [])
    new_val.append(key)
    r[val] = new_val
```

III. Sort dictionary by keys

```
d = {"Ann":88, "David":66, "Cat":77, "Carl":66}

key_val_list = []
for key,val in d.items():
    key_val_list.append((key,val))
key_val_list.sort()
```

- Or with comprehensions:

```
key_val_sorted = sorted([(key,val) for key,val in d.items()])
```

IV. Sort dictionary by values

```
d = {"Ann":88, "David":66, "Cat":77, "Carl":66}

val_key_list = []
for key,val in d.items():
    val_key_list.append((val,key))
val_key_list.sort()
```

- Or with comprehensions:

```
key_val_sorted = sorted([(val,key) for key,val in d.items()])
```

V. Remove duplicates from a list

```
t = [2, 3, 4, 2, 2, 3]
d = { }
for x in t:
    d[x] = 1

unique_list = list(d.keys())
```

Main dictionary patterns

- Loops over dictionary (max grade)
- Reverse dictionary (grades to names)
- Sort (key, value) pairs by keys (names sorted)
- Sort (key, value) pairs by value (top performers)
- Removing duplicates from a list of values

If you know how to do these 5 tasks –
you have learned dictionaries