Intro to Creative Computing

Week 9: Dictionaries

Dictionaries:

- What are dictionaries? Collections of items with custom indices pointing at them:

```
my_dictionary = {}

my_dictionary["index"] = "item"
my_dictionary[1] = 32

print(my_dictionary)
print(my_dictionary[1])
print(my_dictionary["index"])
# They store what I placed into it under the index I chose – this even be a
# long string
```

Dictionaries:

- More sensible dictionary:

```
months_dictionary = {}

months_dictionary[1] = “January”
months_dictionary[2] = “February”
# …
months_dictionary[12] = “December”
```

- Translating a number (that I might get from some database) to a human readable form.

Dictionaries:

- How are they different from lists?
- They have no order, they are key/value pairs instead (index/item)

Dictionaries:

- Your own translation dictionary:

```
eng2cz = {}

eng2cz[“hello”] = “ahoj”
eng2cz[“thursday”] = “čtvrtek”
eng2cz[“icecream”] = “zmrzlina”
```

- For this to work nicely we would like to check if we have some words in the dictionary?

Checking dictionaries:

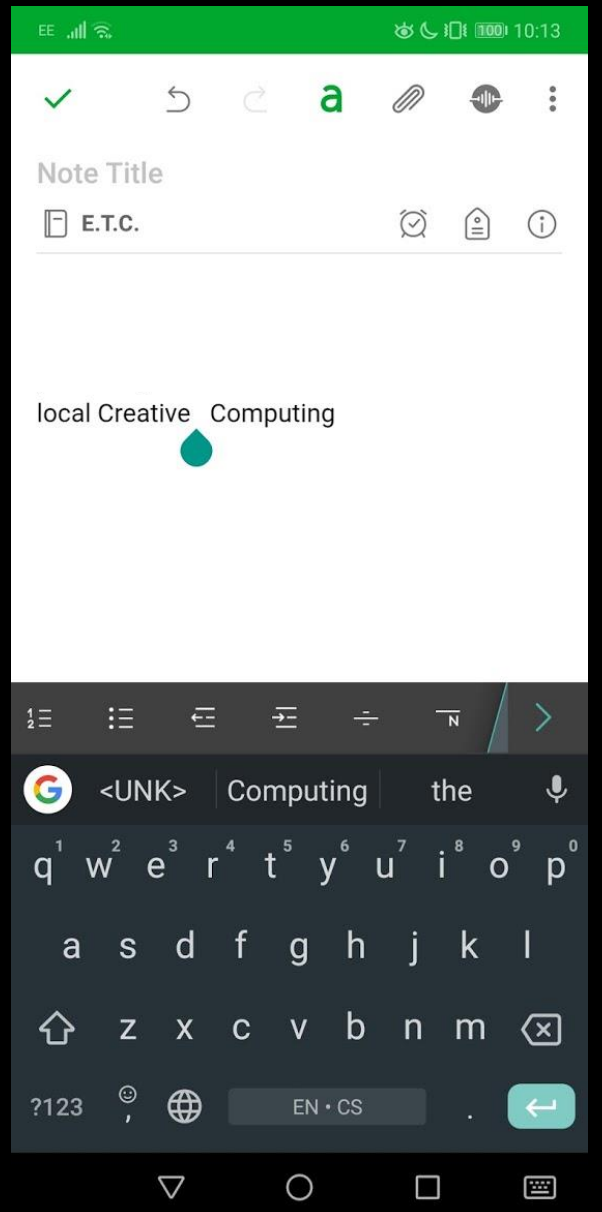
- Check if we have a word in the dictionary:

```
eng2cz = {}

eng2cz[“hello”] = “ahoj”
eng2cz[“thursday”] = “čtvrtek”
eng2cz[“icecream”] = “zmrzlina”

# ... continue

if word in eng2cz:
        print(word, "is", eng2cz[word], "in Czech")
else:
        print("Don't know that word!")
```

<UNK>

- What if we don’t know the word?

```
if word in eng2cz:
        print(word, "is", eng2cz[word], "in Czech")
else:
        print(“<UNK>")
```



Dictionaries:

- Dictionaries can be useful to count word occurrences:

```
occurrences = {}

for word in text.split():
        if word not in occurrences:
                occurrences[word] = 1
        else:
                occurrences[word] = occurrences[word] + 1

print(occurrences)
```

Dictionaries:

- Dictionaries could store entire large strings / objects / images – we only need to wait until we make it to them in this class :)

```
images = {}
images[“cat”] = << … some magic to load an image

books = {}
books[“alice”] = << … whole loaded text from the Alice in Wonderland

machine_learning_models = {}
machine_learning_models[“GAN”] = << … load a ML GAN model

# … etc …
```

Keys and items:

- Keys = list of indices we have (ask with “in” if it contains things)

```
print(eng2cz.keys())

# dict_keys(['hello', 'thursday', 'icecream'])
```

- Items = list of coupled things = list of tuples

```
print(eng2cz.items())

# dict_items([('hello', 'ahoj'), ('thursday', 'čtvrtek'), ('icecream',
'zmrzlina')])
```

- .values()

Tuple:

- Tuple is like a list of items
- **Twist**: It cannot be changed after we make it! << immutable

```
my_tuple = (“apples”, “cinnamon”, “red wine”)

print(my_tuple) # works ok
print(my_tuple[0]) # works ok

my_tuple[2] = “carrots” # will cause an error!
```

- *Ps: usage of tuples is kinda special, can’t think of one … (maybe if you want to tell the user “don’t touch this”)*

Aliasing:

- Let's talk about aliasing – it's not specifically connected to anything, but we can encounter it with lists and dictionaries:

```
opposites = {"up": "down", "right": "wrong", "yes": "no"}
aliasedCopy = opposites

copy = opposites.copy()
# shallow copy

aliasedCopy["up"] = "potato"
# note you just changed both opposites and alised copy!
print(opposites)

# to chanage only one use the copyed one
copy["right"] = "rightMan!"
print(copy)
```

Reading

- Go through the **chapter 12** on dictionaries at:
  https://runestone.academy/runestone/books/published/thinkcspy/

  12.0 some notes ?

Pause 1

Tasks with dictionaries:

- **Task 1**: Using dictionary for a translation of a text. Write your own translation (think of it as not just language to language, but any transformation you want with it).

- Load a text from a file, “translate” it and save the results into another file.

- Hint: If you stumble on a word which is not in the dictionary you can put “<UNK>” (or some other symbol).

Tasks with dictionaries:

- **Task 2**: Read a file and count occurrences of every word. We are finally equipped to do this easily with dictionaries (*do you remember last classes advanced task?*).
- Print the 5 most common words.
  - Bonus: Do this again but don't count the generic words from the original text.

- Bonus (**Advanced**):
  - Use matplotlib to visualize these most common words (x axis = different words, y axis = number of occurences)

Pause 2

Images

- How computer sees images?
- How can we load an image?
- How can we alter an image?
- And then save it?