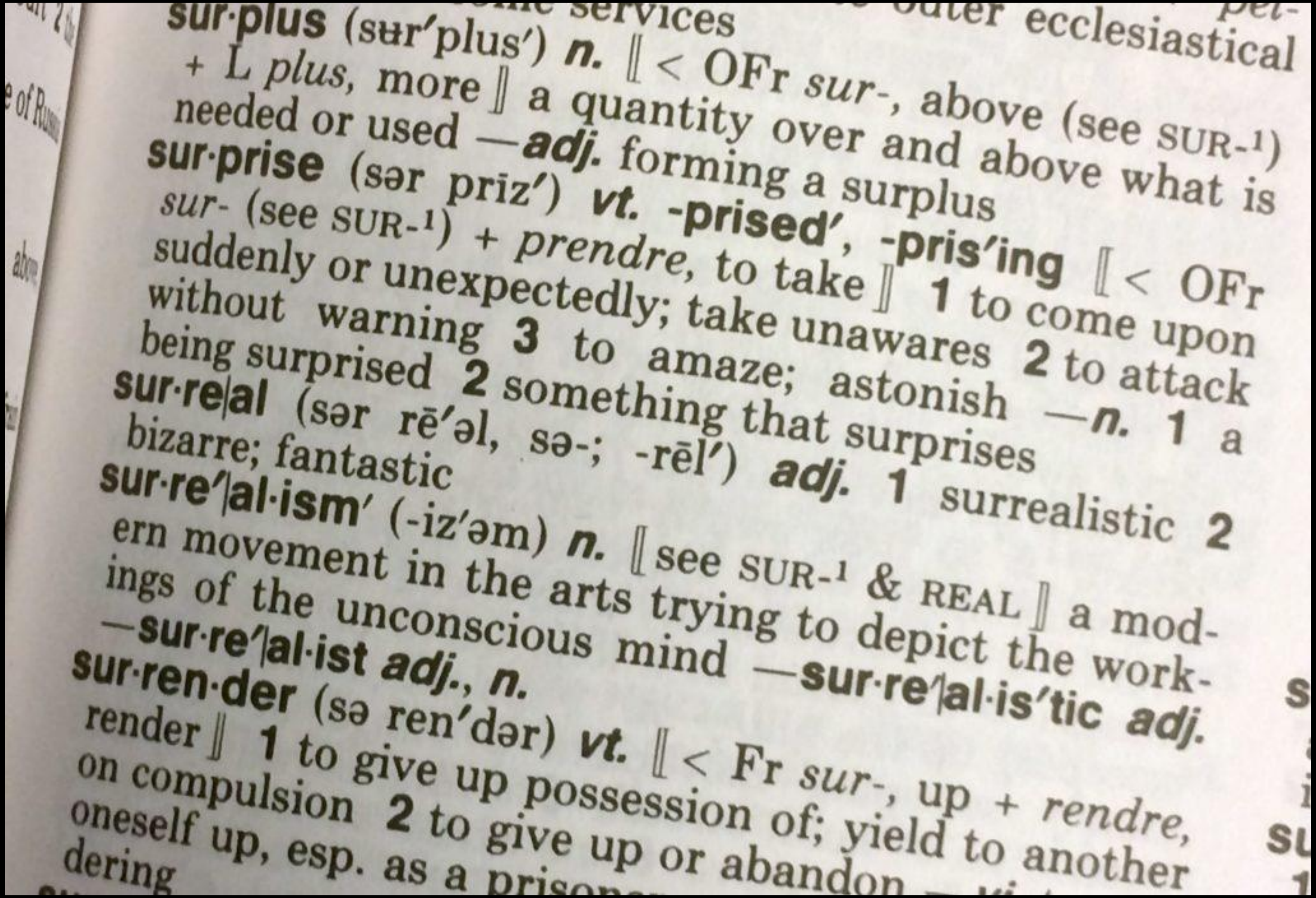
Intro to Creative Computing

Week 9: Dictionaries

Dictionaries:



services … outer ecclesiastical … pet-

sur·plus (sur′plus′) ***n.*** ‖< OFr *sur-*, above (see SUR-¹) + L *plus*, more‖ a quantity over and above what is needed or used —***adj.*** forming a surplus

sur·prise (sər prīz′) ***vt.*** **-prised′**, **-pris′ing** ‖< OFr *sur-* (see SUR-¹) + *prendre*, to take‖ **1** to come upon suddenly or unexpectedly; take unawares **2** to attack without warning **3** to amaze; astonish —***n.*** **1** a being surprised **2** something that surprises

sur·re|al (sər rē′əl, sə-; -rēl′) ***adj.*** **1** surrealistic **2** bizarre; fantastic

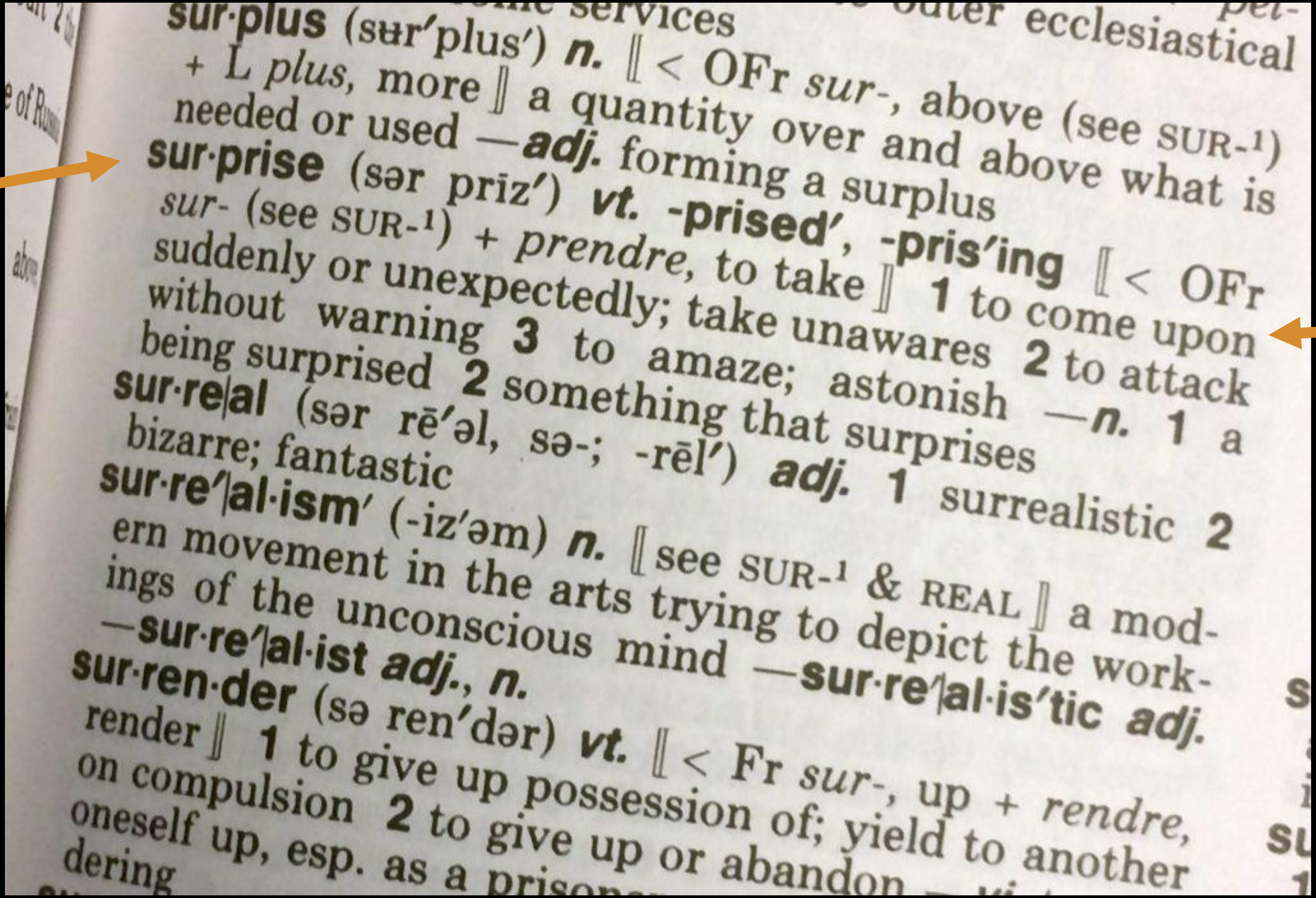
sur·re′|al·ism′ (-iz′əm) ***n.*** ‖see SUR-¹ & REAL‖ a modern movement in the arts trying to depict the workings of the unconscious mind —**sur·re′|al·is′tic** ***adj.*** —**sur·re′|al·ist** ***adj.***, ***n.***

sur·ren·der (sə ren′dər) ***vt.*** ‖< Fr *sur-*, up + *rendre*, render‖ **1** to give up possession of; yield to another on compulsion **2** to give up or abandon … oneself up, esp. as a prison… dering

Dictionaries:

Index

Item



services ... outer ecclesiastical ... per-

sur·plus (sur′plus′) ***n.*** ‖ < OFr *sur-*, above (see SUR-^{1}) + L *plus*, more ‖ a quantity over and above what is needed or used —***adj.*** forming a surplus

sur·prise (sər prīz′) ***vt.*** **-prised′**, **-pris′ing** ‖ < OFr *sur-* (see SUR-^{1}) + *prendre*, to take ‖ **1** to come upon suddenly or unexpectedly; take unawares **2** to attack without warning **3** to amaze; astonish —***n.*** **1** a being surprised **2** something that surprises

sur·re|al (sər rē′əl, sə-; -rēl′) ***adj.*** **1** surrealistic **2** bizarre; fantastic

sur·re′|al·ism′ (-iz′əm) ***n.*** ‖ see SUR-^{1} & REAL ‖ a modern movement in the arts trying to depict the workings of the unconscious mind —**sur·re′|al·is′tic** ***adj.*** —**sur·re′|al·ist** ***adj.***, ***n.***

sur·ren·der (sə ren′dər) ***vt.*** ‖ < Fr *sur-*, up + *rendre*, render ‖ **1** to give up possession of; yield to another on compulsion **2** to give up or abandon ... oneself up, esp. as a prison... dering

Dictionaries:

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

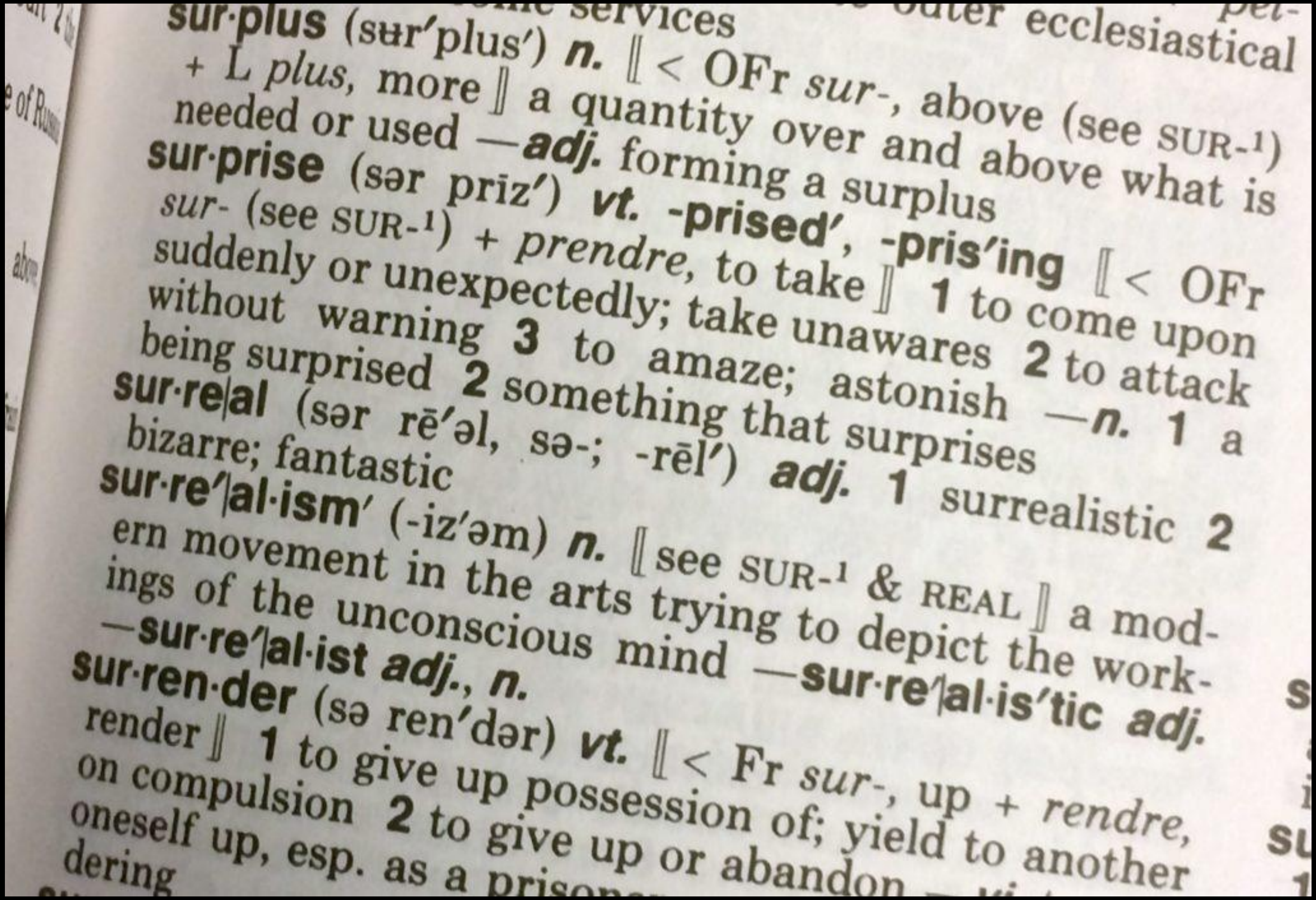
```
my_dictionary = {}

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:

- Deleting an item:

```
my_dictionary = {}

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

del inventory[“index”]

print(my_dictionary)
```

Dictionaries:



services outer ecclesiastical

sur·plus (sur′plus′) *n.* ‖< OFr *sur-*, above (see SUR-¹) + L *plus*, more‖ a quantity over and above what is needed or used —*adj.* forming a surplus

sur·prise (sər prīz′) *vt.* **-prised′**, **-pris′ing** ‖< OFr *sur-* (see SUR-¹) + *prendre*, to take‖ **1** to come upon suddenly or unexpectedly; take unawares **2** to attack without warning **3** to amaze; astonish —*n.* **1** a being surprised **2** something that surprises

sur·re|al (sər rē′əl, sə-; -rēl′) *adj.* **1** surrealistic **2** bizarre; fantastic

sur·re′|al·ism′ (-iz′əm) *n.* ‖see SUR-¹ & REAL‖ a modern movement in the arts trying to depict the workings of the unconscious mind —**sur·re′|al·ist** *adj.*, *n.* —**sur·re′|al·is′tic** *adj.*

sur·ren·der (sə ren′dər) *vt.* ‖< Fr *sur-*, up + *rendre*, render‖ **1** to give up possession of; yield to another on compulsion **2** to give up or abandon oneself up, esp. as a prison

dering

```
d = {}

d["surprise"] = "to come upon
..."
d["surreal"] = "surrealistic,
bizarre, ..."

# etc. etc.

print(d)
```

Dictionaries:

- More sensible dictionary:

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

import datetime                                  # < use case example
d = datetime.date.today()
print(d.month) # 11
print(months_dictionary[d.month]) # November
```

- 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:

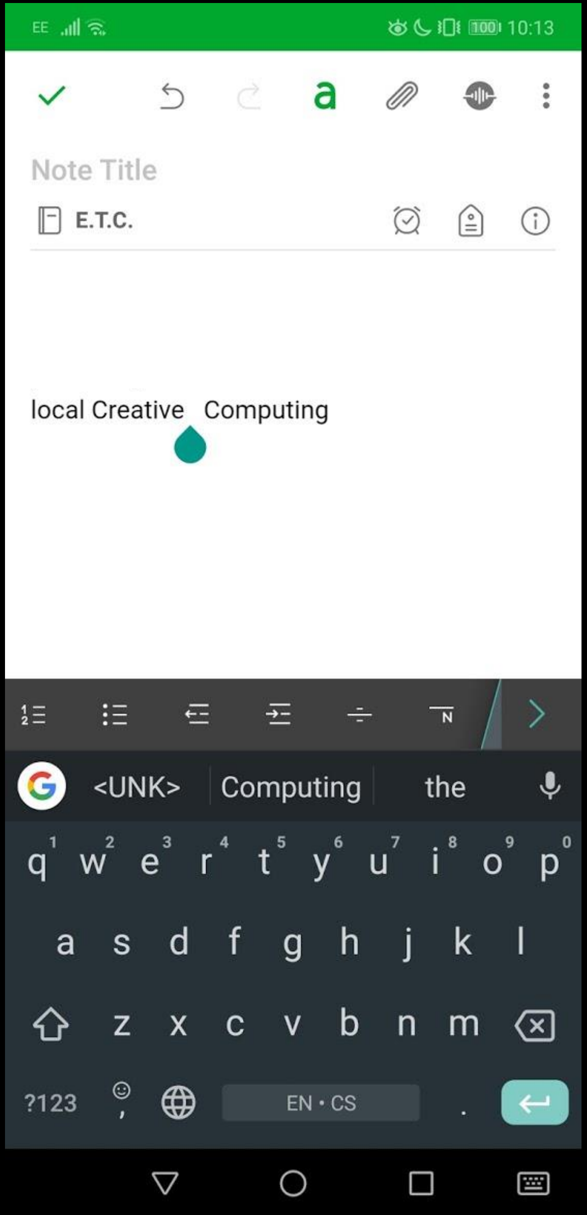
- Check if we have a word in the dictionary:

```
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>")
```



local Creative Computing

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()

```
print(eng2cz.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 aliased copy!
print(opposites)

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

Reading

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

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

Libraries

- What is a library?

Libraries

- What is a library?

```
import os
import math
import sys
import matplotlib

print(math.pi)
print(sys.version)
```

Libraries

- What is a library?
- *(semi-definition) Python library is a collection of functions and methods that allows you to perform many actions without writing your code.*
    - Often community made
    - Often ready made for fast work
    - Often with cool examples!

Versions?

- python,   pip,
  python2, pip2,
  python3, pip3

- conda

```
import sys
print(sys.version)
```

```
# linux + mac

which python
python --version

which pip

conda info
```

```
# windows

where python*
python --version

where pip

conda info
```

Versions?

```
conda info
```

- conda

- *Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.*

Path?

- Example (my pc without python):
- C:\Users\vitek-ntb-win>echo %PATH%
- *C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;D:\Program Files\CalibreEPUB\;C:\Program Files\PuTTY\;D:\Program Files\Git LFS;C:\Users\vitek-ntb-win\AppData\Local\Microsoft\WindowsApps;C:\Users\vitek-ntb-win\AppData\Local\GitHubDesktop\bin*

```
# linux + mac
echo $PATH

# windows
echo %PATH%
```

Libraries … ?

- How does this have anything to do with libraries?

```
import matplotlib
```

- Besides some default libraries which come in all python versions (*standard libraries*), many other require installation and also have versions – you might need to do this first:

```
pip install matplotlib
```

… actually

```
pip install matplotlib
```

- This ^^ only works in a case where someone has released a built code for us
- If it’s a small and obscure code, we might need to

```
pip install git+https://github.com/matplotlib/matplotlib.git
```

… actually

```
pip install matplotlib
```

- This ^^ can also sometimes install a new version of the code, while the rest of our code relies on some older version, in that case …

```
pip install matplotlib==

pip install matplotlib==2.0.2
```

… actually

- All of this is **usually relatively doable** … you just test if it works and if it doesn’t *someone online probably encountered the same problem* and their solution can help …

The task?

- Find a python library of interest
- Install it
- Try it’s examples
- And show it to me!

The task?

- Find a python library of interest
- Install it
- Try it's examples
- And show it to me!

- Some places to start?
  - https://medium.mybridge.co/34-amazing-python-open-source-libraries-for-the-past-year-v-2019-93d6ee11aceb
  - https://github.com/vinta/awesome-python
  - Bokeh (nice graphs), Pyxel and Pygame (games), BeautifulSoup and nltk (natural language processing)

The task?

- Find a python library of interest
- Install it
- Try it’s examples
- And show it to me!

```
# hint:
# if things don’t work on local, use
# google colab and this

!pip install matplotlib
```

- Some places to start?
  - https://medium.mybridge.co/34-amazing-python-open-source-libraries-for-the-past-year-v-2019-93d6ee11aceb
  - https://github.com/vinta/awesome-python
  - Bokeh (nice graphs), Pyxel and Pygame (games), BeautifulSoup and nltk (natural language processing)

Homework

- Check the exercises at https://runestone.academy/runestone/books/published/thinkcspy/Dictionaries/Exercises.html
  - Exercise 2 can be useful to get through the basics
  - 3, 4 might help out with today’s in class exercises
  - And finally 5 is pretty funny