finding your own role in the artistic field

bis dahin kannst du anschauen terminal:
- was bedeutet cd oder ls, (+)
- was ist brew install, (+)
- was ist der unterschied zu pip install, (+)
- wie führt man ein python-skript über commandline aus (+)
- espeak (+)

- andre emails Explination

alex

bash / shell +
git (git for beginners) tutorial
https://www.youtube.com/watch?v=8KCQe9Pm1kg
tesseract +
n+7
python 3
7 nächste wort
nltk library

ocr +
dot files (wichtig fur git -werden versteckt - könnte man anzeigen lassen) tutorials
curl command ?
rsync files
chmod +
imagemagick +

nachstes mal (tomorrow)
git
make files

miert kell ezt a nyelvet megtanulni

- alles viel einfacher zugänglicher , Sachen automatisiert (bash) machen, voorloop, ffm pack

milyen cellal tudom ezt kesobb hasznalni
programme nutzen die nicht zuganligich sind
kleine programme schreiben

mi a cel ezt a nyelvet megtanulni

milyen kontextusban tudom ezt hasznalni
allen möglichen
video Daten visualisieren

schnell Klips zusammenschneiden
automatisieren - interessante programme

youtube video subscription transcription michael

How To Use The find Command
https://www.youtube.com/watch?v=KCVaNb_zOuw

output header

https://www.youtube.com/watch?v=KvHCQmfrmdg&list=PLmM6cQ3o6oQ9RJdFD9cgvRYKFiaCpwYvh&index=11

Terminal (Command Line)
https://www.youtube.com/watch?v=I65C4ZXK4ek

https://www.youtube.com/watch?v=jDINUSK7rXE

Shell Scripting
https://www.youtube.com/watch?v=hwrnmQumtPw

Make files Shell Scripting
https://unix.stackexchange.com/questions/270778/how-to-write-exactly-bash-scripts-into-makefiles

Tesseract (OCR)
https://www.youtube.com/watch?v=QhJiOCwz-_I

Natural Language Toolkit
https://www.nltk.org

Git with Terminal
https://github.com/codepath/ios_guides/wiki/Using-Git-with-Terminal

Python

https://howtothink.readthedocs.io

(I think this is one of the most recent iteration)

Its content/approach may feel disconnected from your interest at first, but it will get you through all the programming patterns you will eventually encounter, and once you know the basics, it becomes easier to dive into specific modules and libraries to work with more specific things (network, audio, video, statistics, etc)

You want to be in a situation where you have enough understanding of the core functionality of Python, and understand what is programming and scripting, so that you can start explore resources like https://pypi.python.org/ and be able to read the technical documentation on your own to start building upon other people tools, and/or combining

them.

_____

26.02.

terminal

cd = change directory
ls = listing all the files of the current directory
ls -l = passing parameters
clear = clean up
alias = shortcut for running 1 or more terminal commands
control X = restart terminal

creating temporary alias = alias test="echo 'test'"

creating permanent alias =
1. command line sudo nano .bash_profile
2. saving the alias such as
alias dt= ‘cd ~/Desktop/‘
3. exit= control X
4. restart terminal

show all hidden files on desktop=
1.Desktop SZAKACS$ defaults write com.apple.finder AppleShowAllFiles YES
2. killall Finder

reverse command=
1.Desktop SZAKACS$ defaults write com.apple.finder AppleShowAllFiles NO
2. killall Finder

creating an alias from it=
1. open sudo nano .bash_profile
2. alias showFiles=‘defaults write com.apple.finder AppleShowAllFiles YES;
#semmicolon because the it needs an other command line to execute the command
alias showFiles='defaults write com.apple.finder AppleShowAllFiles YES; killall
Finder'

hiding
alias hideFiles='defaults write com.apple.finder AppleShowAllFiles NO; killall Finder'

control X

restart terminal

new alias commands

- hiding the dock

- changing the clock
- emptying the trash bin

alias tutMode='defaults write com.apple.docks autohide -bool true;
killall Dock; #executing the command line
defaults write com.apple.menuextra.clock IsAnalog -bool true;
killall SystemUIServer; #executing the command line
rm -rf ~/Trash/*;' (why is this command not working?!)
No matching processes belonging to you were found
???

- setting back to regular mode

alias regMode='defaults write com.apple.docks autohide -bool false;
killall Docks;
defaults write com.apple.menuextra.clock IsAnalog -bool false;
killall SystemUIServer;'

typing the word alias and it will pop up the the saved alias list

mkdir = making a directory
touch (test.html)= creating a file in the directory

mv main.css css/main.css = moving a file into a directory
mv test.html #file css #directory /test.html #file
open index.html = opening a file
open . -a “Sublime Text” = opening the file in “Sublime Text”
opening a file in a different programme = open index.html -a “Sublime Text”
rm index.html = deleting a file
mv index.html index2.html = renaming a folder
open . = opening the current folder where I am
cd .. = going outside of the folder
rm -r (reculsiv) (putting the directory after) = deleting a directory
clear = clearing the line
top = performance of the system
ls = list of all the files in the folder
cd = change directory
cd .. = going up the directory
command K = clears the directory
cd / = takes you to the root directory
pwd = print working directory
cd ../(zalantest3) = moving to an other directory
rmdir (name of the directory) = deleting the directory
nano index.html = creating a file
cp (file that you want to copy) (new file) = copy

moving file up to 2 directory and changing the name of the file at the same time =
mv orig_files.txt #filename ../test_file.txt #newfilename

screencapture -x -t jpg Code.jpg = creating a screenshot
find . -type f
(find) (current directory) (type of the file) (file) (name of the file)

find . -type f (“name”) * (shows all the options)

find . -type f (“iname”) * (shows all the options)
(not key sensitive)

find . -type f -name “*.py”

#finding python files

find . -type f -mmin -10
#finding files modified less then 10min ago

find . -type f -mmin +1 -mmin -5

find . -type f -mtime 20
#days

find . -size +5M
#finding files over 5MG

homebrew= software package management system OSX
brew install (write the software name)

installing things with brew will install them into /usr/local/

pip= packager for the python world
home directory (e.g. ~/.local/lib/python2.7/site-packages/) or in some global search-path of your python interpreter (e.g. /usr/local/lib/python2.7/dist-packages/)

Standard Input/Output, Writing to Files (Redirection & Pipelines)

echo Hello WORLD

echo hello world > example.txt

less example.txt #reading this text line

| pip

a | b = run command a and then pass the result to command b

wc (word count)

more command line
less command line

cat -n shoppinglist.txt = numbering the lines

> overwrites the file: The left contents will overwrite the contents of the right

>> appends to the end the file: The left contents will be added at the end of the contents of the right

cat > shoppinglist.txt

ctrl + d = to go back to the terminal

cat >> shoppinglist.txt = appends the output

quotes use in terminal

variable=dollar
ZALANs-MacBook-Pro:Test Folder SZAKACS$ echo "$variable"
dollar
ZALANs-MacBook-Pro:Test Folder SZAKACS$ echo '$variable'
$variable
ZALANs-MacBook-Pro:Test Folder SZAKACS$ number=5
ZALANs-MacBook-Pro:Test Folder SZAKACS$ text="give me a dollar"
ZALANs-MacBook-Pro:Test Folder SZAKACS$ echo "$text $number"
give me a dollar 5
ZALANs-MacBook-Pro:Test Folder SZAKACS$ echo '$text $number'
$text $number

‘#2 words name’ = has to be put always in ‘’

‘’ is not equal to “”
when special characters (i.e. $ #…) are involved

‘’ singe quotes = preserves the literal meaning of the string. special character functions (i.e. $ #…) will not be called while inside single quotes.

“” double quotes = special character functions (i.e. $ #…) are called when used inside double quotes.

variable=dollar
ZALANs-MacBook-Pro:Desktop SZAKACS$ echo "$variable"
dollar

mkdir -p = parent directory

head command in terminal

The head command reads the first few lines of any text given to it as an input and writes them to standard output (which, by default, is the display screen).
head - n 7 (name of the file after the number) = shows only 7 lines

changing permissions with chmod

file types to recognise

-rw-rw-r- - = file
-=file
r=readable
w=writable
-= is not runable
rw- = access for the group
r - - = permission for others

drwxr-xr-x=directory

chmod
parts of the command that needs to be run:
1. programme
2. option
3. arguments that might go with it

Bash
“Bourne-Again”, default shell for Mac OS X

vim

27.02
bash / shell +
git (git for beginners) tutorial
https://www.youtube.com/watch?v=8KCQe9Pm1kg
tesseract +
n+7
python 3
7 nächste wort
nltk library

ocr +
dot files (wichtig fur git -werden versteckt - könnte man anzeigen lassen) tutorials
curl command ?
rsync files
chmod +
imagemagick +

nachstes mal (tomorrow)
git

make files

alex
vereinfachung
Zeile fur zeile command line befehle

bash
.sh

#!/bin/bash immer beginnen
___

tesseract

foto reinzeihen directory
alt taste command line clicken

double tab

tesseract - png - name of the txt file

tesseracttest SZAKACS$ tesseract Screen\ Shot\ 2018-01-19\ at\ 15.26.52\ 2.png text2.txt

chmod 777 (name von der folder)

leserrechte

imagemagick

convert anstatt imagemagick

converter jpg to pdf

python

natural language tool kit

extension

text2.txt

~/ homefolder kopieren

cp tesseracttest/text2.txt.txt ~/xpub

aymeric

tesseract png - txt
python script

import
std in std output #libraries (nur fur terminal - python) output fur command line
from sys import stdin,stdout
stdin
stdout
erweitern python dass ich lesen und schreiben kann

open offnen das Datei
read den ganzen text rausnehmen
speichern als varaible (content) - beinhaltet das ganze text

saving as the variable (platzhalter) the text to be able to work with it
text = open ("example.txt")

text.read() - holt den ganzen text aus der text document raus
() braucht kein input fur die Funktion den ich ausfuhre
print(text.read()) - output in terminal

was ist ein variable
kann etwas beinhalten (zahl, text, liste, form annehmen)
mach man um später ausrufen
(content)

variablen
funktionen
(print) (echo)

welche teil Prozesse
der computer weiss nicht was Wörter sind
weiss nicht wie er kann zahlen

was ist die 1 aufgäbe

1. text auf Wörter aufteilen
wenn Leerzeichen kommt speichere alles bis dorthin aus
nltk
import nltk - Funktion raussuchen

aufgabe mehrere aufgaben aufteilen

splitt funktion python

zuweisen

alles immer in der variablen speichern
weiter mit der variable weiterarbeiten
if

for loop 10

x = ‘blue,red,green’
x.split(“,”)

[‘blue’, ‘red’, ‘green’]
>>>

>>> a,b,c = x.split(“,”)

>>> a
‘blue’

>>> b
‘red’

>>> c
‘green’

len (words) list

python
“w” = write a new file

strings = textual data
test1.py

https://www.youtube.com/watch?v=W8KRzm-HUcc&list=PL-osiE80TeTskrapNbzXhwoFUiLCjGgY7&index=4

message = ‘Bobby\’s World’

or
message = “Bobby’s World”

message = """Bobby's World was a good
cartoon in the 1990s"""

print(message)

len = length (counting the letters)

[ ] going to the position
starts always with 0
[0:5] start and the end of the word

message = 'Hello World'
                  01234
print(message[0:5])

it is also possible to write

print(message[:5])

if you want only the end part
print(message[6:])

slicing
https://www.youtube.com/watch?v=ajrtAuDg3yw

lower case
print(message.lower())

upper
print(message.upper())

counting
print(message.count('l'))

find
print(message.find('World'))
6
because it starts the 6th variable

replacing parts=
message = 'Hello World'

new_message = message.replace('World','Universe')

print(new_message)

greeting = 'Hello'
name = 'Michael'

message = greeting +', ' + name
print(message)

better way of writing (because of the +)
message = '{}, {}. Welcome!'.format(greeting, name)

f strings (works from python 3)

message = f'{greeting}, {name}. Welcome!'

fstrings makes possible to give functions

message = f'{greeting}, {name.upper()}. Welcome!'

dir function

print(dir(name))

shows all the attributes and methods

seeing more information

print (help(str))

message = f'{greeting}, {name}. Welcome!'

Integers and Floats - Working with Numeric Data
test2.py

integer = whole number (3)
float = des number (3.12)

num = 3

print(type(num))
class 'int'>

num = 3

print(type(num))

class ‘float’>

print(3 * (2 + 1))

num = 1

num = num + 1

print(num)

2

easier way of writing it
num = 1

num += 1

print(num)

abs
absolute value
print(abs(-3))
3

print(round(3.75))
4

print(round(3.75, 1))
8

num_1 = 3
num_2 = 2

print(num_1 == num_2)
False

casting
num_1 = '100'
num_2 = '200'

num_1 = int(num_1)
num_2 = int(num_2)

print(num_1 + num_2)
300

lists, tuples, sets

courses = ['History', 'Math', 'Physics', 'CompSci']

print(len(courses))
4

courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses[0])
History

negative indexes will start from the end of the list
-1 is always the last one

courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses[-1])
CompSci

courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses[0:2])
['History', 'Math']

courses = ['History', 'Math', 'Physics', 'CompSci']

print(courses[0:])
['History', 'Math', 'Physics', 'CompSci']

.append = adding

courses = ['History', 'Math', 'Physics', 'CompSci']

courses.append('Art')

print(courses)
['History', 'Math', 'Physics', 'CompSci', 'Art']

if you want to add to certain place you have to use the insert methods

courses = ['History', 'Math', 'Physics', 'CompSci']

courses.insert(0,'Art')

print(courses)
['Art', 'History', 'Math', 'Physics', 'CompSci']

extend methode
courses = ['History', 'Math', 'Physics', 'CompSci']

courses_2 = ['Art', 'Education']

courses.extend(courses_2)

print(courses)
['History', 'Math', 'Physics', 'CompSci', 'Art', 'Education']

remove

.pop #removing the last one

popping and dropping back

courses = ['History', 'Math', 'Physics', 'CompSci']

popped = courses.pop()

print(popped)
print(courses)

CompSci
['History', 'Math', 'Physics']

reverse

courses = ['History', 'Math', 'Physics', 'CompSci']

```
courses.reverse()
print(courses)
```

sorting list

sort

```
courses = ['History', 'Math', 'Physics', 'CompSci']

nums = [1, 5, 2, 4, 3]

courses.sort()
nums.sort()

print(courses)
print(nums)
```

```
['CompSci', 'History', 'Math', 'Physics']
[1, 2, 3, 4, 5]
```

sorting in reversing order

```
courses = ['History', 'Math', 'Physics', 'CompSci']

nums = [1, 5, 2, 4, 3]

courses.sort(reverse=True)
nums.sort(reverse=True)

print(courses)
print(nums)
```

sorted function

```
courses = ['History', 'Math', 'Physics', 'CompSci']

#nums = [1, 5, 2, 4, 3]

sorted_courses = sorted(courses)

print(sorted_courses)
#print(nums)
```

```
['CompSci', 'History', 'Math', 'Physics']
```

min and max

```
courses = ['History', 'Math', 'Physics', 'CompSci']
```

```
nums = [1, 5, 2, 4, 3]

sorted_courses = sorted(courses)

#print(sorted_courses)
print(min(nums))
```
1

printing the min + max

```
courses = ['History', 'Math', 'Physics', 'CompSci']

nums = [1, 5, 2, 4, 3]

sorted_courses = sorted(courses)

#print(sorted_courses)
print (f'{min(nums)},{max(nums)}')
```
1, 5

sum

```
courses = ['History', 'Math', 'Physics', 'CompSci']

nums = [1, 5, 2, 4, 3]

sorted_courses = sorted(courses)

#print(sorted_courses)
print(sum (nums))
```
15

```
courses = ['History', 'Math', 'Physics', 'CompSci']


print(courses.index('CompSci'))
```
3

in - operator

```
courses = ['History', 'Math', 'Physics', 'CompSci']


print('Art' in courses)
```

False

for loop

```
courses = ['History', 'Math', 'Physics', 'CompSci']
```

```
for item in courses:
        print(item)
```

```
History
Math
Physics
CompSci
```

enumerate function

```
courses = ['History', 'Math', 'Physics', 'CompSci']

for index, course in enumerate(courses):
        print(index, course)
```

```
0 History
1 Math
2 Physics
3 CompSci
```

adding a start value such as 1

```
courses = ['History', 'Math', 'Physics', 'CompSci']

for index, course in enumerate(courses, start=1):
        print(index, course)
```

```
1 History
2 Math
3 Physics
4 CompSci
```

joining

```
courses = ['History', 'Math', 'Physics', 'CompSci']

course_str = ','.join(courses)

print(course_str)
```

```
History,Math,Physics,CompSci
```

```
courses = ['History', 'Math', 'Physics', 'CompSci']

course_str = '-'.join(courses)

new_list = course_str.split(' - ')

print(course_str)
print(new_list)
```

```
History-Math-Physics-CompSci
```

['History-Math-Physics-CompSci']

mutable and immutable

#Mutable

```
list_1 = ['History', 'Math', 'Physics', 'CompSci']
list_2 = list_1

print(list_1)
print(list_2)

list_1[0] = 'Art'

print(list_1)
print(list_2)
```

```
['History', 'Math', 'Physics', 'CompSci']
['History', 'Math', 'Physics', 'CompSci']
['Art', 'Math', 'Physics', 'CompSci']
['Art', 'Math', 'Physics', 'CompSci']
```

#Immutable

```
tuple_1 = ('History', 'Math', 'Physics', 'CompSci')
tuple_2 = tuple_1

print(tuple_1)
print(tuple_2)

#tuple_1[0]='Art'

#print(tuple_1)
#print(tuple_2)
```

```
('History', 'Math', 'Physics', 'CompSci')
('History', 'Math', 'Physics', 'CompSci')
```

list = modification
tuple = look through

sets

sets don’t really care about order

#sets

```
cs_courses = {'History', 'Math', 'Physics,' 'CompSci'}

print(cs_courses)
```

```
{'Math', 'Physics,CompSci', 'History'}
```

```
cs_courses = {'History', 'Math', 'Physics,' 'CompSci', 'Math'}

print('Math' in cs_courses)
True
```

finding out what courses do they have in common

#sets

intersections

```
cs_courses = {'History', 'Math', 'Physics', 'CompSci', 'Art'}
art_courses = {'History', 'Math', 'Art,' 'Design'}

print(cs_courses.intersection(art_courses))
```

{'History', 'Math'}

differences
#sets

```
cs_courses = {'History', 'Math', 'Physics', 'CompSci', 'Art'}
art_courses = {'History', 'Math', 'Art,' 'Design'}

print(cs_courses.difference(art_courses))
{'Physics', 'Art', 'CompSci'}
```

union
#sets

```
cs_courses = {'History', 'Math', 'Physics', 'CompSci', 'Art'}
art_courses = {'History', 'Math', 'Art,' 'Design'}

print(cs_courses.union(art_courses))
{'Art', 'CompSci', 'Art,Design', 'Math', 'Physics', 'History'}
```

empty lists, tuples, sets

```
# Empty lists
empty_list = []
empty_list = list()

#Empty Tuples
empty_tuple = ()
empty_tuple = tuple()

#Empty Sets
empty_set = {} #This isn't right ! It's a dict
empty_set = set ()
```

Dictionaries - Working with Key-Value Pairs
https://www.youtube.com/watch?v=daefaLgNkw0&list=PL-osiE80TeTskrapNbzXhwoFUiLCjGgY7&index=5

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}
print(student)
```

```
{'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}
```

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}
print(student.get('name'))
```
John

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}


student['phone'] = '55 555 555'
student['name'] = 'Jane'

print(student)
```
```
{'name': 'Jane', 'age': 25, 'courses': ['Math', 'CompSci'], 'phone': '55 555 555'}
```

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}

student.update({'name': 'Jane', 'age': 26, 'phone': '4444555 444'})
print(student)
```

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}

del student['age']

print(student)
```
```
{'name': 'John', 'courses': ['Math', 'CompSci']}
```

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}

age = student.pop('age')

print(student)
print(age)
```
```
{'name': 'John', 'courses': ['Math', 'CompSci']}
25
```

len
student.keys
student.values
student.items

```
student = {'name': 'John', 'age': 25, 'courses': ['Math', 'CompSci']}
```

```
for key, value in student.items():
        print(key, value)
name John
age 25
courses ['Math', 'CompSci']
```

Conditionals and Booleans
If, Else, and Elif statements

comparisons:
equal ==
not equal !=
greater than >
less than <
greater or equal <=
object identity is

```
# and
# or
# not

if True:
        print('Conditional was True')
Conditional was True

language = 'Python'

if language == 'Python':
        print('language is Python')
else:
        print('No match')
language is Python

language = 'Java'

if language == 'Python':
        print('language is Python')
elif language == 'Java':
        print('language is Java')

else:
        print('No match')

language is Java

user = 'Admin'
logged_in = True

if user == 'Admin' and logged_in:
        print('Admin Page')
```

```
else:
    print('Bad Creds')
```

Admin Page

```
user = 'Admin'
logged_in = False

if user == 'Admin' or logged_in:
    print('Admin Page')
else:
    print('Bad Creds')
```

Admin Page

thinking of if not false = if not logged_in

```
user = 'Admin'
logged_in = False

if not logged_in:
    print('Please Log In')
else:
    print('Welcome')
```

Please Log In

two same objects but two different objects in memory

```
a = [1, 2, 3]
b = [1, 2, 3]

print(a == b)
```

True

```
a = [1, 2, 3]
b = [1, 2, 3]

print(a is b)
```

False

```
a = [1, 2, 3]
b = [1, 2, 3]

print(id(a))
print(id(b))
print(a is b)
```

4492254984
4495181384

```
False

#False Values:
#False
#None
#Zero of any numeric type
#Any empty sequence. For example, ‘’. (), [].
#Any empty mapping. For example, {}.

condition = False

if condition:
        print('Evaluated to True')
else:
        print('Evaluated to False')
Evaluated to False
```

Loops and Iterations-For/While Loops

```
nums = [1, 2, 3, 4, 5]

for num in nums:
        print(num)
1
2
3
4
5

nums = [1, 2, 3, 4, 5]

for num in nums:
        if num ==3:
                print('Found!')
                break
        print(num)
1
2
Found!

nums = [1, 2, 3, 4, 5]

for num in nums:
        if num ==3:
                print('Found!')
                continue
        print(num)

1
2
```

```
Found!
4
5

for num in nums:
        for letter in 'abc':
                print(num, letter)
1 a
1 b
1 c
2 a
2 b
2 c
3 a
3 b
3 c
4 a
4 b
4 c
5 a
5 b
5 c

for i in range(1,11):
        print(i)
1
2
3
4
5
6
7
8
9
10

x = 0

while x < 10:
        print(x)
        x +=1
0
1
2
3
4
5
6
7
```

```
8
9


x = 0

while x < 10:
        if x == 5:
                break
        print(x)
        x +=1
0
1
2
3
4
```

Functions
https://www.youtube.com/watch?v=9Os0o3wzS_I&list=PL-osiE80TeTskrapNbzXhwoFUiLCjGgY7&index=8

specific tasks

def = definition

#pass = we won't do anything for now

```
def hello_func():
        print('Hello Function!')

print(hello_func())
Hello Function!
None

def hello_func():
        print('Hello Function.')

hello_func()
hello_func()
hello_func()
hello_func()
hello_func()
hello_func()
hello_func()
hello_func()
hello_func()

Hello Function.
Hello Function.
Hello Function.
```

Hello Function.
Hello Function.
Hello Function.
Hello Function.
Hello Function.
Hello Function.

easily changeable in the print section

```
def hello_func(greeting):
        return '{} Function.'.format(greeting)


print(hello_func('Hi'))
```

Hi Function.

!!! check scope video
https://www.youtube.com/watch?v=QVdf0LgmICw

```
def hello_func(greeting, name='You'):
        return '{}, {}'.format(greeting, name)


print(hello_func('Hi', name= 'Zalan'))
```
Hi, Zalan

```
def student_info(*args, **kwargs):
        print(args)
        print(kwargs)

student_info('Math', 'Art', name='John', age=22)
```
('Math', 'Art')
{'name': 'John', 'age': 22}

```
def student_info(*args, **kwargs):
        print(args)
        print(kwargs)

courses = ['Math', 'Art']
info = {'name':'John', 'age':22}

student_info(*courses, **info)
```

('Math', 'Art')
{'name': 'John', 'age': 22}

```
# Number of days per month. First value placeholder for indexing purposes.
```

```
month_days = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

def is_leap(year):
    """Return True for leap years, False for non-leap years."""
    return year % 4 ==0 and (year % 100 != 0 or year % 400 == 0)

def days_in_month(year, month):
    """Return number of days in that month in that year."""

    if not 1 <= month <= 12:
        return 'Invalid Month'

    if month == 2 and is_leap(year):
        return 29

        return month_days[month]

print(is_leap(2017))
```
False

```
print(is_leap(2020))
```

True

```
# Number of days per month. First value placeholder for indexing purposes.

month_days = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

def is_leap(year):
    """Return True for leap years, False for non-leap years."""
    return year % 4 ==0 and (year % 100 != 0 or year % 400 == 0)

def days_in_month(year, month):
    """Return number of days in that month in that year."""

    if not 1 <= month <= 12:
        return 'Invalid Month'

    if month == 2 and is_leap(year):
        return 29

        return month_days[month]

print(days_in_month(2017, 2))
```
28

_____

01.03.

list methods

Method
Parameters
Result
Description
append
item
mutator
Adds a new item to the end of a list
insert
position, item
mutator
Inserts a new item at the position given
pop
none
hybrid
Removes and returns the last item
pop
position
hybrid
Removes and returns the item at position
sort
none
mutator
Modifies a list to be sorted
reverse
none
mutator
Modifies a list to be in reverse order
index
item
return idx
Returns the position of first occurrence of item
count
item
return ct
Returns the number of occurrences of item
remove
item
mutator
Removes the first occurrence of item

doubleStuff

pure function
modifiers
functional programming style

list comprehensions

general syntax
[<expression> for <item> in <sequence> if <condition>]

```
alist = [4,2,8,6,5]
blist = [num*2 for num in alist if num%2==1]
print(blist)
```
[10]

```
alist = [ [4, [True, False], 6, 8], [888, 999] ]
if alist[0][1][0]:
   print(alist[1][0])
else:
   print(alist[1][1])
```
888

```
myname = "Edgar Allan Poe"
namelist = myname.split()
init = ""
for aname in namelist:
    init = init + aname[0]
print(init)
```
EAP

Whereas split will break a string into a list of “words”, list will always break it into a list of characters

functions
= problem solving strategy coming from smaller chunks

= a function is a named sequence of statements

http://interactivepython.org/courselib/static/thinkcspy/Functions/Functionsthatreturnvalues.html

```
def function_name(arg1 #arguments, arg2, …):
        statement 1
        statement 2
```

```
def bitcoin_to_usd(btc):
        amout = btc * 527
```

```
        print(amout)

bitcoin_to_usd(3.85)
2028.95

#define a baic function
def func1():
        print("I am a function")

#function that takes arguments

def func2(arg1, arg2):
        print (arg1, " ", arg2)

#function that returns a value

def cube(x):
        return x*x*x

#functions with default value for an argument

def power(num, x=1):
        result = 1
        for i in range(x):
                result = result * num
        return result

#functions with variable number of arguments

def multi_add (*args):
        result = 0
        for x in args:
                result = result + x
        return result

"""func1 ()
print (func1())
print (func1)"""

"""func2(10,20)
print (func2(10,20))
print (cube (3))"""
"""print (power (2))
print (power (2,3))
print (power (x=3, num=2))"""
print (multi_add(4, 5, 10, 4))

the accumulator pattern
http://interactivepython.org/courselib/static/thinkcspy/Functions/
TheAccumulatorPattern.html
```

———

```
def function (n):
    print(n)

function(47)
47

def function (n = 1):
    print(n)
    return n * 2

x = function(42)
print(x)
42
84

def isprime(n):
    if n <= 1:
        return False
    for x in range(2, n):
        if n % x == 0:
            return False
    else:
        return True

n = 6
if isprime(n):
    print(f'{n} is prime')
else:
    print(f'{n} not prime')

6 not prime

def isprime(n):
    if n <= 1:
        return False
    for x in range(2, n):
        if n % x == 0:
            return False
    else:
        return True

def list_primes():
    for n in range(100):
        if isprime(n):
            print(n, end= ' ', flush=True)
    print()
```

list_primes()
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Objects

https://www.lynda.com/Python-tutorials/Objects/614299/687487-4.html

```
class Duck:
	def quack(self):
		print('Quaaack!')

	def walk(self):
		print('Walks like a duck.')

def main():
	donald = Duck()
	donald.quack()
	donald.walk()

if __name__ == '__main__': main()
```

Quaaack!
Walks like a duck.

```
class Duck:
	sound = 'Quaaack'
	walking = 'Walks like a duck.'

	def quack(self):
		print(self.sound)

	def walk(self):
		print(self.walking)

def main():
	donald = Duck()
	donald.quack()
	donald.walk()

if __name__ == '__main__': main()
```

Quaaack
Walks like a duck.

the string type

conditional structures

https://www.lynda.com/Python-tutorials/Conditional-structures/661773/707225-4.html

```
#example file for working with conditional statements

def main():
        x, y = 1000, 100

        # conditional flow uses if, elif, else

        if (x < y):
                st = "x is less than y"
        elif (x == y):
                st = "x is the same as y"
        else:
                st = "x is greater than y"
    print (st)

# conditional statements let you use "a if C else b"

st= "x is less than y" if (x<y) else "x is greater than or the same as "
    print (st)

if __name__ == '__main__':
        main()
```

https://www.lynda.com/Python-tutorials/Conditionals/614299/687484-4.html

```
x = 42
y = 73

if x < y :
        print('x < y: x is {} and y is {}' .format(x,y))

x < y: x is 42 and y is 73

x = 42
y = 73

if x > y :
        print('x < y: x is {} and y is {}' .format(x,y))
else:
        print('do something else')
```

```
do something else

x = 42
y = 73

if x > y :
        print('x > y: x is {} and y is {}' .format(x,y))
elif x < y:
        print('x < y: x is {} and y is {}' .format(x,y))

else:
        print('do something else')
```

x < y: x is 42 and y is 73

loops

while:

```
words = ['one', 'two', 'three', 'four', 'five']

n = 0
while(n<5):
        print (words [n])
        n+= 1
```

one
two
three
four
five

for:

```
words = ['one', 'two', 'three', 'four', 'five']

for i in words:
        print(i)
```

one
two
three
four
five

string type
https://www.lynda.com/Python-tutorials/string-type/614299/687490-4.html

```
x = "seven" .upper()
print('x is {}'.format(x))
print(type (x))
```

```
x is SEVEN
<class 'str'>

numeric types

x is 21
<class 'int'>

sequence types

test15.py
x = [1, 2, 3, 4, 5]
x[2] = 42

for i in x:
        print('i is {}' .format(i))

i is 1
i is 2
i is 42
i is 4
i is 5

x = range(10)


for i in x:
        print('i is {}' .format(i))
i is 0
i is 1
i is 2
i is 3
i is 4
i is 5
i is 6
i is 7
i is 8
i is 9


x = range(1,10)


for i in x:
        print('i is {}' .format(i))
i is 1
i is 2
i is 3
i is 4
```

```
i is 5
i is 6
i is 7
i is 8
i is 9

dictionary {}

x = {'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
for k, v in x.items():
        print('k: {}, v: {}' .format(k,v))

k: one, v: 1
k: two, v: 2
k: three, v: 3
k: four, v: 4
k: five, v: 5

Type and id
test15.py


02.03.
import modules and exploring the standard library
https://www.youtube.com/watch?v=CqvZ3vGoGs0&index=9&list=PL-
osiE80TeTskrapNbzXhwoFUiLCjGgY7

my_module.py

print('Imported my_module...')

test = 'Test String'

def find_index(to_search, target):
        '''Find the index of a value in a sequence'''
        for i, value in enumerate(to_search):
                if value == target:
                        return i

        return -1

intro.py
from my_module import find_index
#shortage of the name - my_module mm

courses=['History', 'Math', 'Physics', 'CompSci']

index = find_index(courses, 'Math')
print(index)
```

1

import sys
module path (python looks for it in the computer)
directories - python path environment

from my_module import find_index, test

import sys

courses=['History', 'Math', 'Physics', 'CompSci']

index = find_index(courses, 'Math')
#print(index)
#print(test)
print (sys.path)
Imported my_module...
['/Users/SZAKACS/Desktop/python_exercises', '/Library/Python/2.7/site-packages/pip-9.0.1-py2.7.egg', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python27.zip', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-darwin', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac/lib-scriptpackages', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-tk', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-old', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-dynload', '/Library/Python/2.7/site-packages', '/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python', '/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python/PyObjC']
ZALANs-MacBook-Pro:python_exercises SZAKACS$

importing the module from a different directory

import sys

sys.path.append('/Users/SZAKACS/Desktop/My-Modules')

from my_module import find_index, test

courses=['History', 'Math', 'Physics', 'CompSci']

index = find_index(courses, 'Math')
#print(index)
#print(test)
print (sys.path)
['/Users/SZAKACS/Desktop/python_exercises', '/Library/Python/2.7/site-packages/pip-9.0.1-py2.7.egg', '/System/Library/Frameworks/Python.framework/Versions/2.7/

lib/python27.zip', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-darwin', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac/lib-scriptpackages', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-tk', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-old', '/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-dynload', '/Library/Python/2.7/site-packages', '/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python', '/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python/PyObjC', '/Users/SZAKACS/Desktop/My-Modules']

changing environment variables
terminal
nano ~/.bash_profile
in nano
export PYTHONPATH="/Users/SZAKACS/Desktop/My-Modules"

in terminal
open py3

import my_module
import sys
sys.path

next steps

standard library imports

intro2.py

```
import random

courses = ['History', 'Math', 'Physics', 'CompSci']

random_course = random.choice(courses)

print(random_course)
```
Physics

```
import math

courses = ['History', 'Math', 'Physics', 'CompSci']

rads = math.radians(90)

print(rads)
```
1.5707963267948966

```
import datetime
import calendar

courses = ['History', 'Math', 'Physics', 'CompSci']

today = datetime.date.today()
print(today)
2018-03-02

import datetime
import calendar

courses = ['History', 'Math', 'Physics', 'CompSci']

today = datetime.date.today()
#print(today)

print(calendar.isleap(2018))
False

os = underlining operating system

os.getcwd = get current working directory

import os

courses = ['History', 'Math', 'Physics', 'CompSci']

print(os.getcwd())
/Users/SZAKACS/Desktop/python_exercises

import os

courses = ['History', 'Math', 'Physics', 'CompSci']

print(os.__file__)

/usr/local/Cellar/python3/3.6.3/Frameworks/Python.framework/Versions/3.6/lib/
python3.6/os.py


sys module
stdin() and stdout()
test17.py

stdin = input
stderr = in-between
stdout = output
```

OS Modules _ Use underlying operating system functionality

____

Bash
Bourne Again Shell
(Shell)

nano #name of the bash file after nano

nano opens up

#!/usr/bin/env bash

echo “HELLO!”

close nano

run bash script less #name of the file

make a script executable you have to change the permission of the script

chmod +x myscript2

./myscript2
HELLO!

back in nano
NAME=${1?Error: no name given}
echo "HELLO! $NAME"

./myscript2: line 3: 1: Error: no name given

./myscript2 Zalan #writing the name after name of the script

HELLO! Zalan

back in nano
#!/usr/bin/env bash

NAME=${1?Error: no name given}
NAME2=${2:-friend}
echo "HELLO! $NAME and $NAME2"

myscript2 Zalan

HELLO! Zalan and friend

myscript2 Zalan Peter
HELLO! Zalan and Peter

saving output / input to a file

echo “hello world” > temp.txt
nano temp.txt

> erase the file
>> add to the botton of the file

bash shell putting commands inside of the text file - having bash shell to read the commands out of the text files

vi start

.sh #bash

basicscript
#!/bin/bash

chmod 775 basicscript.sh

executing the shell script
./basicscript.sh

https://gist.github.com/megawertz/7017879

#!/bin/bash

viewing environment variables

echo "The value of the home variable is: "

echo $HOME

issue a command

echo "The output of the pwd command is: "

```
pwd


# that's boring, grab output and make it readable
echo "The value of the pwd command is $(pwd)"


# assign command output to a variable
output=$(pwd)
echo "The value of the output variable is ${output}"


# view data on the command line
echo "I saw $@ on the command line"


# read data from the user
echo "Enter a value: "
read userInput
echo "You just entered $userInput"


# concatenate userinput with command output
echo "Enter a file extension: "
read ext
touch "yourfile.${ext}"
```

```
# check to see if a file exists

if [ -d /etc/sysconfig ]; then

        echo "That file is there and a directory"

else

        echo "Not there or not a directory"

fi

775 BashSnippets.sh

 echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

The value of the home variable is
/Users/SZAKACS

echo "The output of the pwd command is: "
pwd

The output of the pwd command is:
/Users/SZAKACS/xpub


The value of the pwd command is /Users/SZAKACS/xpub

# assign command output to a variable

output=$(pwd)

echo "The value of the output variable is ${output}"

The value of the output variable is /Users/SZAKACS/xpub

./BashSnippets.sh option
I saw option on the command line

$ is a variable - easy way of access components

$@ gives more options

text files in bash

#!/bin/bash

i=1
while read f; do
```

```
        echo #Line $i $f"
        ((i++))
done < file.txt
```

tokenising
nltk
bash shell to translate / make file
git
functions

pos_tag

http://www.pythonforbeginners.com/dictionary/python-split

http://benjamindalton.com/extracting-nouns-with-python/

https://www.youtube.com/watch?v=AKcxEfz-EoI

https://www.youtube.com/watch?v=l_dIleafLZ8

https://stackoverflow.com/questions/18647707/count-letters-in-a-text-file

https://www.tutorialspoint.com/How-to-detect-vowels-vs-consonants-in-Python

http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python

http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

https://stackoverflow.com/questions/35086440/python-how-to-compute-the-top-x-most-frequently-used-words-in-an-nltk-corpus

find out the list (tutorial)
vorloop

```
for in in word_tokenize(text):
        print (i)
```

https://www.youtube.com/watch?v=Flpj_D8b1Vg

bash
basic conditional example if..then..else
http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-6.html

```
 #!/bin/bash
        if [ "foo" = "foo" ]; then
          echo expression evaluated as true
        fi
```

expression evaluated as true

```
 #!/bin/bash
        if [ "foo" = "foo" ]; then
          echo expression evaluated as true
        else
          echo expression evaluated as false
        fi
```

expression evaluated as true

```
  #!/bin/bash
        T1="foo"
        T2="bar"
        if [ "$T1" = "$T2" ]; then
           echo expression evaluated as true
        else
           echo expression evaluated as false
        fi
```

expression evaluated as false

Loops for, while and until

```
#!/bin/bash
      for i in $( ls ); do
          echo item: $i
      done
```

```
item: Applications
item: Desktop
item: Documents
item: Downloads
item: Dropbox
item: Library
item: Movies
item: Music
item: Pictures
item: Public
item: Zotero
item: ee
```

item: hs_err_pid2571.log
item: nltk_data
item: scan-utils
item: tmphCS_Fw.plist
item: xpub

```
#!/bin/bash
        for i in `seq 1 10`;
        do
                echo $i
        done
```

1
2
3
4
5
6
7
8
9
10

```
 #!/bin/bash

COUNTER=0
        while [  $COUNTER -lt 10 ]; do
           echo The counter is $COUNTER
           let COUNTER=COUNTER+1
        done
```

The counter is 0
The counter is 1
The counter is 2
The counter is 3
The counter is 4
The counter is 5
The counter is 6
The counter is 7
The counter is 8
The counter is 9

```
 #!/bin/bash
        COUNTER=20
        until [  $COUNTER -lt 10 ]; do
           echo COUNTER $COUNTER
           let COUNTER-=1
        done
```

COUNTER 20

```
COUNTER 19
COUNTER 18
COUNTER 17
COUNTER 16
COUNTER 15
COUNTER 14
COUNTER 13
COUNTER 12
COUNTER 11
COUNTER 10
```

```
#!/bin/bash
        function quit {
            exit
        }
        function hello {
            echo Hello!
        }
        hello
        quit
        echo foo
Hello!
```

```
        #!/bin/bash
        OPTIONS="Hello Quit"
        select opt in $OPTIONS; do
            if [ "$opt" = "Quit" ]; then
             echo done
             exit
            elif [ "$opt" = "Hello" ]; then
             echo Hello World
            else
             clear
             echo bad option
            fi
        done
```

python to check through:

strings

The in and not in operators
loop

Looping and Counting¶

9.17. A find function¶

9.18. Optional parameters¶

strings (+)
lists (+)
dictionaries (+)
functions (+)

loop, if else statements (+)
join, split (+)
for loop (+)
while
https://www.learnpython.org/en/Loops
schleife

‘” comments block”’

to check
Traversal and the for Loop: By Item¶

———

Joca feedback 02.03.

you have to think for the code there are several solutions like building blocks

you have to choose the solution, which fits your needs the best to be able to create an efficient workflow

creating a pipeline in terminal (bash + makefiles +git)

stdout save in text ~ terminal

this way it will be easier to import for other pages (setting up first in terminal)

check nltk statistics: characters, word tagger (verb, nouns)

functions - is how to set up the code better

exit python in terminal (control+d)