PYTHON - REPETITION

Bendik Egenes Dyrli

UiA University of Agder

TABS VS SPACES

- Spaces are the preferred indentation method
- Python 3 disallows mixing the use of tabs and spaces for indentation.

SHA-BANG!

```
#!/usr/bin/env python3

#!/bin/python3
```

VARIABLES

```
name="Rick"

age=42

occupation="Time travler"
```

VARIABLES

```
str_age = "42"
age = 42
```

TYPES

```
age = 42

aage = 42.0

aaage = "42.0"

aaaage = True
```

TYPES

```
string = str(string)

integer = int(integer)

flyttall = float(flyttall)

boolean = bool(boolean)
```

TYPES

```
text = "Hello my pincode is "
pincode = 3301

print (text + pincode)
```

TYPES

```
text = "Hello my pincode is "
pincode = 3301

print (text + pincode)
```

```
Traceback (most recent call last):
  File "/home/skandix/types.py", line 4, in <module>
    print (text + pincode)
TypeError: can only concatenate str (not "int") to str
```

TYPES

```
text = "Hello my pincode is "
pincode = 3301

print (text + str(pincode))
```

```
Hello my pincode is 3301
```

IF

```
RickSober = False
MortyLikesJessica = True
JessicaLikesMorty = None

if (RickSober != True):
    print("Rick is not sober")
    if (MortyLikesJessica and JessicaLikesMorty):
        print("Morty and Jessica Likes each other")
    else:
        print("Jessica doesn't like Morty :( ")

elif (RickSober == True):
    print("WHAT, Rick is sober... what dimension is this ?")
```

INPUTS

```
take_a_guess = input('-->')

if input == 9:
    print ("Congrats... you knew the magic number")
else:
    print("sorry, no prize for you.")
```

STRINGS

```
name = "Rick"
age = 42

# String interpolation
print ("My name is " +name+ " and i'm " +age+ " years old")

# str.format
print ("My name is {name}, and i'm {age} years old".
                                format(name=name,age=age))
# F(ormat)-strings
print (f"My name is {name}, and i'm {age} years old.")
```

FOR

```
languages = ['Lua', 'Bash', 'Python']
for lang in languages:
    print(lang)
```

FOR

```
for number in range(10):
    print (number)
```

FOR

```
string = "Can we iterate over strings"
for word in string:
    print (word)
```

FOR

```
counter = 0
for number in range(10):
    counter += 1
    print ("on ", counter, "and counting")
```

LEN

```
state = "mississippi"
print(len(state))
```

HELP

```
help(len)
```

SYNTAX ERROR

```
Morty = "HEEEYY, Rick can i iterate over string?"
for word in Morty
    print (word)
```

SYNTAX ERROR

```
Rick = "YES MORTY YOU CAN!"
for word in Rick:
    print (word<Paste>
```

SYNTAX ERROR

```
print("SHIT, MORTY WHAT DID YOU DO!?!?!")
```

CONTROL FLOW

WHILE

```
while (expression is true)
    do something indefinitely
```

```
while True:
    print("Hi, i'm mr meeseeks")
```

FOR-LOOP

```
for target in iterable:
    do something over the length/size of the iterable
```

```
name = "Rick"
for letter in name:
    print(letter)
```

RANGE

```
for number in range(10):
    print(number)
```

BREAK

like in C, it will break out of the innermost enclosing for or while loop

```
count = 0
while True:
    count += 1
    if count is 6:
        break
    else:
        print("Hi, i'm Mr Meeseeks")
```

CONTINUE

Also borrowed from C, continues with the next iteration of the for loop.

```
for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found a number", num)
```

PASS

```
StrangeList = []

def futureFunction():
    pass

for k in StrangeList:
    pass

futureFunction()
```

PASS

```
while True:
    pass # Busy-wait for keyboard interrupt (CTRL + C )
```

TRY CATCH LOOP

```
# Handle exceptions with a try/except block
try:
    raise IndexError("This is an index error")

except IndexError as e:
    pass

except (TypeError, NameError):
    pass

else:
    print("All good!")
```

FUNCTIONS

```
def Hello(name):
    print (f"Hello {name}")
```

FUNCTIONS

```
def Hello(name):
    print (f"Hello {name}")
```

```
# Functions get called like this.
print (Hello("Riiiiiiick"))

# Functions can be assigned to variables
say_my_name = Hello


say_my_name("Heisenberg")
```

DEFAULT ARGUMENTS VALUES

```
def where_is_rick(position_rick="Secret Lab")
    return (position_rick)

print(where_is_rick())
```

DEFAULT ARGUMENTS VALUES

```
def where_is_rick(position_rick="Secret Lab")
    return (position_rick)

print(where_is_rick("HOME"))
```

POSITIONAL ARGUMENTS

```
def where_is_rick(morty_position, position_rick="Secret Lab"):
    return (morty_position, position_rick)

print (where_is_rick())
```

POSITIONAL ARGUMENTS

```
def where_is_rick(morty_position, position_rick="Secret Lab"):
    return (morty_position, position_rick)

print (where_is_rick())
```

```
Traceback (most recent call last):
  File "/home/skandix/test.py", line 4, in <module>
      print (where_is_rick())
      TypeError: where_is_rick() missing 1 required positional
```

POSITIONAL ARGUMENTS

```
def where_is_rick(morty_position, position_rick="Secret Lab"):
    return (morty_position, position_rick)

print (where_is_rick("AT SCHOOL"))
```

ARBITRARY ARGUMENT LISTS

```
def smith_family(*args, seperator="/"):
    return (seperator.join(args))

    print (smith_family("rick", "morty", "summer"))
```

FUNCTIONS WITH TYPE HINTING

This is the closest you will get to type checking in python

```
def ransom_note(text: str) -> str:
    output = ""
    for letter_index in range(len(text)):
        if letter_index % 2 is 0:
            output += text[letter_index].lower()

        elif letter_index % 2 is not 0:
            output += text[letter_index].upper()
    print(output)

ransom_note("give me back my portal gun
and no one will get hurt!")
```

FUNCTIONS WITH TYPE HINTING

- List[int] – a list of integers
- List[str] - a list of strings
- Tuple[bool, float] – a tuple with two items: a boolean and a float
- Dict[str, int] – a dictionary accessed using a string key and holding an integer
- Dict[str, List[int]] – a dictionary with a string key holding a list of integers

```
from typing import List

def IReturnAListWithStrings(text:str) -> List[str]:
    print(list(text))

IReturnAListWithStrings("HELLLOOO ")
```

LAMBDA EXPRESSIONS

Lambdas are small anonymous functions which can be created with the lambda keyword

```
Hello = lambda text: print(f"Hello {name}")
```

```
def Incrementing(n):
    return lambda x: x + n
```

CAN YOU USE LAMBDAS FOR EVERYTHING?

If you are crazy enough... yes !

```
from math import sin, cos, log, exp, pi
nr = lambda x,f,tol: print(f(x)) if abs((f(x)-x)/f(x))<=tol el
nr(2.5,lambda x: (2*x**3+3)/(3*x**2),9e-2)
nr(1500.,lambda x: (-x*log(x)+10001*x)/(5*x+1),1e-6)
nr(.5,lambda x: (x**2-sin(x)+x*cos(x)+2)/(2*x+cos(x)),.5e-8)
nr(-1.,lambda x: (x**2-sin(x)+x*cos(x)+2)/(2*x+cos(x)),5e-8)
nr(3.,lambda x: (x**2+10)/(2*x),.5e-8)
nr(-.5,lambda x: x-(log(x**2+1.)-exp(.4*x)*cos(pi*x))/((2*x)/(
```

DOCUMENT STRINGS

```
def recipie_concentrated_dark_matter():
    """
    Galactic Federation AIN'T GETTING SHIT YOOO... *BUUURP*
    """
    pass

print (recipie_concentrated_dark_matter.__doc__)
```

```
    Galactic Federation AIN'T GETTING SHIT YOOO... *BUUURP*
```

TUPLE

Tuple is Immutable !

```
cords = (x,y)
for x in range(10):
    for y in range(10):
    print (x,y)
```

SLICES

```
[start:stop:step]
```

```
showMeWhatYouGot = "GET SCHWIFTY"
print (showMeWhatYouGot[1])
print (showMeWhatYouGot[:8])
print (showMeWhatYouGot[:8:2])
print (showMeWhatYouGot[-1])
```

SLICES

```
showMeWhatYouGot = "GET SCHWIFTY"
print (showMeWhatYouGot[1])
print (showMeWhatYouGot[:8])
print (showMeWhatYouGot[:8:2])
print (showMeWhatYouGot[-1])
```

G	E	T		S	C	H	W	I	F	T	Y
0	1	2	3	4	5	6	7	8	9	10	11

HOW TO CHECK IF LIST IS EMPTY

```
Rick = ['R', 'i', 'c','k']
if not Rick:
    print ("LIST IS EMPTY")
else:
    print ("LIST IS NOT EMPTY MORTY!")
```

DATA STRUCTURES

LISTS

a way of storing data in python

```
my_list = []
```

KEYWORDS

USINGS LISTS AS STACKS

- LIFO - Last-in First-Out

```
stack = [13,21,34,55,89,144]

stack.append(233)
stack.append(377)

print (stack) # we now have two new numbers

print(stack.pop()) # pops the last item from the list
print(stack.pop())

print (stack)
```

LIST COMPREHENSIONS

A more concise way to create a list

```
square = []
for x in range(10):
    square.append(x**2)

print (square)
```

LIST COMPREHENSIONS

```
squares = list(map(lambda x: x**2, range(10)))
```

or, equivalently:

```
squares = [x**2 for x in range(10)]
```

LIST COMPREHENSIONS

```
[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]

[fixtures for fixtures in range(100, 106)]
```

LIST COMPREHENSIONS

```
[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]
```

```
combinatorics = []
for x in [1,2,3]:
    for y in [3,1,4]:
        if x != y:
            combinatorics.append((x,y))

print (combinatorics)
```

DEL

The way of deleting an item or clearing a list in python

```
my_list = [1,2,3,4,5] # initialize list with values
print (my_list)

del [0] # delete the value stored in the first index in the li
print (my_list)

del my_list # this will delete everything in the list
print (my_list)
```

SET

```
my_set = set("interdimensional cable")
```

- Unordered collection
- No duplicate elements
- Supports mathematical operations such as
  - union
  - intersection
  - difference
  - symetric difference

SET

```
A = set("interdimensional cable")
B = set("earth cable")

# Set operations

print (A - B) # letters in A but not in B

print (A | B) # letters in A or B or both

print (A & B) # letters in both A and B

print (A ^ B) # letters in a or b but not both
```

DICTIONARY

```
my_dict = {'Rick':-1, 'Morty':137, 'Jerry':'0'}
```

- Dictionaries are awesome !
- can be seen as a key, value store
- keys need to be unique

DICTIONARY

```
state = {'Rick':'Lab', 'Morty':'School', 'Jerry':'Don't know}
print (state)

state['Beth'] = 'Surgery'
print (state)

del state['Jerry']
state['Summer'] = 'Home'

print (list(state)) # prints the list "version" of the dict
print (sorted(state)) # sorts the dictionary

print('Jerry' in state)
print('Rick' in state)
```

DICT COMPREHENSIONS

```
print ({x: x**2 for x in (2,4,6,8)})
```

LOOPING TECHNIQUES

LOOPING LISTS

```
my_list = [1,2,3,4,5,6,7,8,9]

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

LOOPING DICTS

```
esolangs = {'brainfuck':'brain damage',
            'lolcode':'lol',
            'unlambda':'needs more lambda'}

for key, value in esolangs.items():
    print (f"Key: {key}\nValue: {value}\n")
```

LOOPING SEQUENCES

```
for index, value in enumerate(['paper',
                               'scissor',
                               'rock',
                               'lizard',
                               'spock'])

    print (f"index: {key}\nValue: {value}\n")
```

LOOPING OVER TWO SEQUENCES AT THE SAME TIME

```
questions = ['name', 'quest', 'favorite color']
answers = ['lancelot', 'the holy grail', 'blue']

for quest, answer in zip(questions, answers):
    print (f"What is your {quest}? It is {answer}.")
```

LOOPING OVER LIST IN REVERSED ORDER

```
for rev in reversed(range(1,10,2)):
    print (rev)
```

STANDARD LIBARY

HOW TO IMPORT ...

GOOD

```
import os

print (os.getcwd())
```

BAD!

```
from os import *

print (getcwd())
```

HOW TO IMPORT ...

```
from os import getcwd

print(getcwd)
```

IMPORT OS

```
import os
print(os.getcwd())
```

IMPORT SHUTIL

```
import shutil

print (shutil.copyfile('ricks_lab', 'ricks_secret_lab'))

print (shutil.move('ricks_lab', './secret_basement/'))
```

IMPORT GLOB

```
import glob
print (glob.glob('*.py'))
```

IMPORT ARGPARSE

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-l','--location', help="Where do you want
parser.add_argument('-o','--on', help="gotta turn on the porta
args = parser.parse_args()

if not args.on:
    print ("MORTY DID YOU TURN ON THE PORTAL GUN YET?!")
else:
    if not args.location:
        print (f"MORTY, WHERE SHOULD WE GO?!?")
    elif args.location:
        print (f"Teleporting to {args.location}")
```

IMPORT RANDOM

```
from random import randint, sample

food = ['taco', 'pizza', 'veggies']

print(randint(1,1024))
print (f"Today's dinner is {sample(food,1)}")
```

IMPORT DATETIME

```
from datetime import datetime

now = datetime.now()

year = now.strftime("%Y")
month = now.strftime("%m")
day = now.strftime("%d")

date = now.strftime("%d.%M.%Y")
time = now.strftime("%H:%M:%S")

print (date)
print (time)
```

IMPORT CODECS

```
import codecs
codecs.encode('java zone 19,.-', 'rot_13')
```

IMPORT TIME

```
from time import

print ("Sleeping for 5 seconds")
sleep(5)
print ("GOOD MORNING")
```

IMPORT PLATFORM

```
import platform
import os

if platform.system is "win32" or plattform.system is "Windows"
    os.system('cls')
else:
    os.system('clear')
```

SOCKET

```
import socket

def staticBuffer(host, port):

    buffer = "A"*2025+"\\n\\r\\n"
    buff = str.encode(buffer)

    print(buff)

    strem = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    strem.connect((host,port))
    print (strem.send(buff))
    print ("Amount Sent: {:d}".format(len(buff)))
    print (strem.recv(4096))
```

REGEX

- [ ] - Characted class
- ( ) - Capture Group
- { } - Repetition Counts

REGEX

Regex	Matching	Description
[a-zA-Z]	**abc**123**DEF**	Matches any characted between a-z or A-Z
[0-9]	abc**123**DEF	Matches any number between 0-9
[0-9]{2,4}	**1234 5678**	matches any number from 2,4 characteds long
a+	**a aa** b**a**b	Matches one or more consecutive 'a' characters

IMPORT RE

```
import re

strings = ['aabbccdd', 'aabbccøøææåå', 'ææøøåå']

pattern = re.compile(r'[a-zA-Z0-9]+')
for string in strings:
    print(re.findall(pattern, string))
```

IMPORT JSON

```
import json

family_memebers =  '{ "Smith_Family":["Rick",
                                      "Morty",
                                      "Summer",
                                      "Beth",
                                      "Jerry"]}'
family_memebers = json.loads(family_memebers)

print(family_memebers["Smith_Family"])
```

a Libary that are not part of the Standard Libary

REQUESTS

```
import requests

def getData(url:str) -> str:
    return requests.get(url)


if getData("https://links.datapor.no/").status_code is "200":
    print ("200!")
else:
    print(f"Oops, we got {getData("https://links.datapor.no/).
```

MISC

MAP

map (function(), iterable which is to be mapped)

```
def missing_https(url):
    return "https://"+url


sites = ['i.imgur.com/400.jpg', 'i.imgur.com/401.jpg',
         'i.imgur.com/402.jpg', 'i.imgur.com/403.jpg', 'i.imgu

result = map(missing_https, sites)
print(list(result))
```

CLASSES

```
class TimeTravel:
    TravelsCount = 0 # Class variable

    def __init__(self, when, where):
        self.where = where # Instance variables
        self.when = when
        TimeTravel.TravelsCount +=1

    def displayTravelCount():
        print (f"Total Travels: {TimeTravel.TravelsCount}")

    def displayTravels(self):
        print (f"Where: {self.where}\nWhen: {self.when}\n")
```

FILES

```
for line in open('password', 'r'):
    print (line)
```

DECORATORS

```
def decode(func:str):
    def wrapper():
        from codecs import decode
        return decode(func(), "rot-13")
    return (wrapper)

@decode
def secretSafe():
    PincodeToRicksSafe = "Cvpxyrf"
    return (PincodeToRicksSafe)

print (secretSafe())
```

REQESTS-HTML

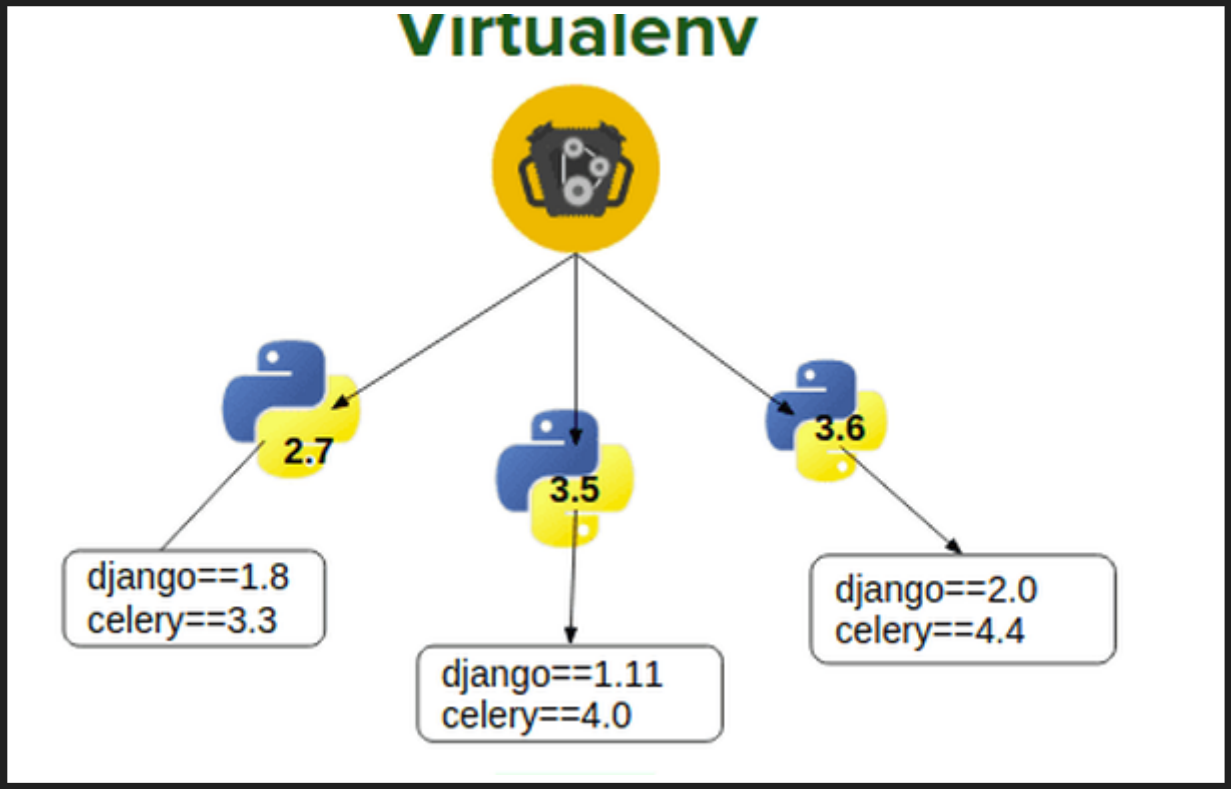
```
pip install request-html
```

```
from request_html import HTMLSession
session = HTMLSession()

def getData():
    return session.get('https://links.datapor.no')


print(getData().html.links)
#print(getData().html.absolute_links)
#print(getData().html.links)
```

PIPENV


Virtualenv
2.7
3.5
3.6
django==1.8
celery==3.3
django==1.11
celery==4.0
django==2.0
celery==4.4

PIPENV

best of all packaging to python.

- creates and manages virtualenv
- add and removes packages from Pipfile
- Install/Uninstall packages
- Generates pipfile.lock

```
pipenv install requests-html --python 3.7

pipenv shell
pipenv run python server.py
```

MULTIPROCESSSING

```
import multiprocessing


def worker():
    """worker function"""
    print('Worker')


if __name__ == '__main__':
    jobs = []
    for i in range(5):
        p = multiprocessing.Process(target=worker)
        jobs.append(p)
        p.start()
```

RESOURCES

https://docs.python.org/3/tutorial/datastructures.html
https://www.cloudflare.com/learning/security/threats/buffer-overflow/
https://docs.python.org/3/tutorial/datastructures.html
https://docs.python.org/3/tutorial/controlflow.html
https://docs.python.org/3/glossary.html#glossary
https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366
https://www.youtube.com/watch?v=66XoCk79kjM
https://docs.python.org/3/tutorial/appetite.html

https://docs.python.org/3/tutorial/interpreter.html
https://docs.python.org/3/tutorial/introduction.html
https://pymotw.com/3/multiprocessing/basics.html