Dawid Pacia

https://drive.google.com/drive/folders/10cs__MrYhlqgELcQsADgLHfAPPKSHL3C?usp=sharing

Introduction time!

Eluwina!

I am Dawid Pacia

I am here because I love to teach, share knowledge and python :)

BRAINLY

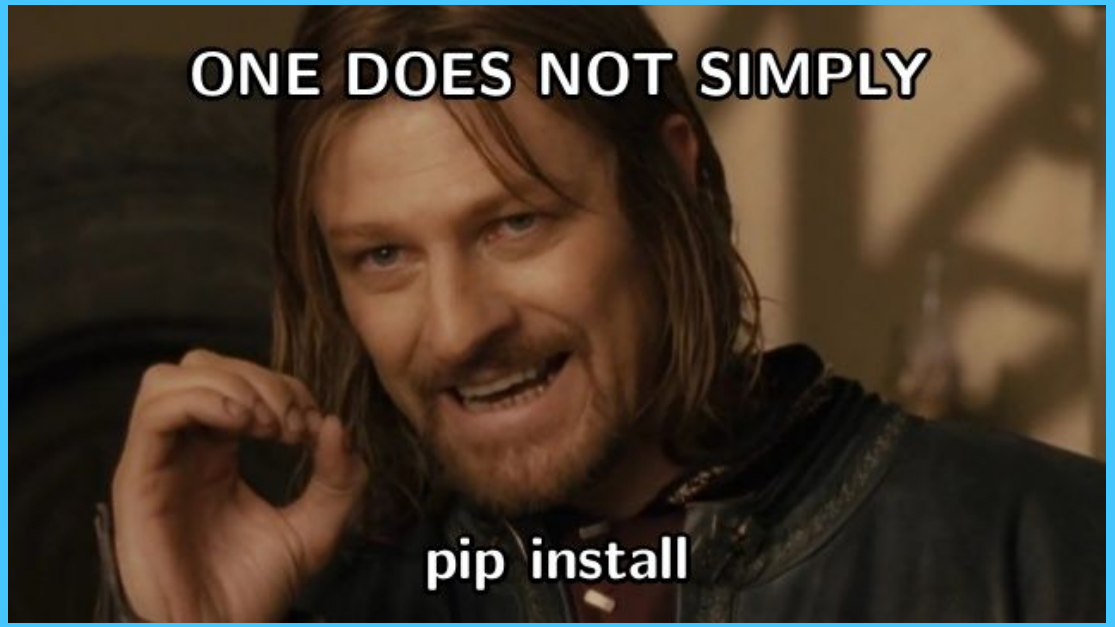
About me:

- Trainer Testing and Python
- Test (Automation) Manager
- Public Speaker
- Lecturer

Getting started

https://drive.google.com/drive/folders/10cs__MrYhlqgELcQsADgLHfAPPKSHL3C?usp=sharing

Environment setup

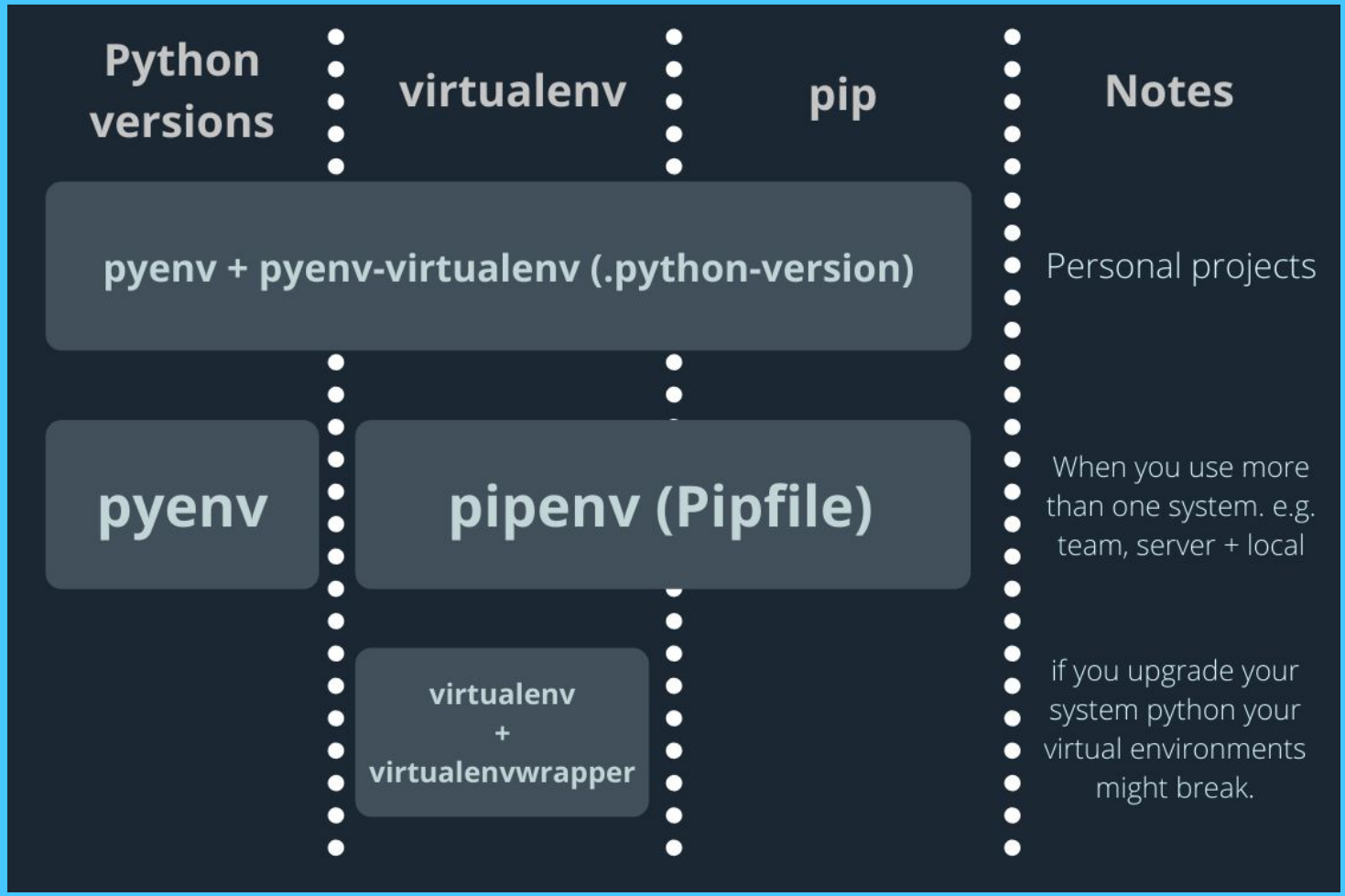


Environment setup

pip install pipenv

pipevn vs venv vs conda/anaconda

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



https://towardsdatascience.com/python-environment-101-1d68bda3094d

Environment setup

Pipfile

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

IDE

console + terminal

IDE

packages & interpreter

Let’s start with something

1) Count how many iterations is done in 1s → *loop while*
2) Measure how much time it takes to execute number of iterations from point 1 → *loop for*
3) Check if *loop while* is faster than *loop for*

Tips:
***import time* → time library**
time.time() → current time in seconds
***time1* - *time2* → calculating time difference in seconds**

setting-up git project git

https://www.udemy.com/course/intro-to-git/

Recommendation

Sourcetree (Atlassian tool)

Warm-up

[task warmup]

```
with open("input.txt") as report_data:
    lines = report_data.readlines()
```

Code analyzers(linters) vs code formatters

Code formatters

- autopep8
- black [*pipenv lock --pre*]
- isort
- YAPF

https://wemake-python-stylegui.de/en/latest/pages/usage/integrations/auto-formatters.html

Linters / code analysers

- pylint
- mypy
- prospector
- flake8

https://realpython.com/python-code-quality/

https://github.com/marketplace/actions/wemake-python-styleguide

Flake8 vs we-make-python

Time for you!

Text parsing

- x.strip()
- x.split(a)
- “ ”.join()
- x.replace(a,b)

Task

[parsing & reg-ex part 1]

regex

[define regex matching]

- mobile phone number
- Website address
- business email e.g. dawid.pacia@brainly.com
- “Name Surname”

https://regex101.com/

Task

[parsing & reg-ex part 2]

```
locals()[key](value) -> # calling functions by name
```

Time for you!

bin, hex, and other creatures

bit operations: &, |, ^, ~, <<, >>

bin(int_value)
BUT!
int(bin_value, 2)

hex(int_value)
BUT!
int(bin_value, 16)

"string".encode("utf-8").hex()

bytes.fromhex(hex_value).decode(“utf-8”)

Task

[LC & binary conversion]

Task

[more bits part 1]

```
"{0:b}".format(int value)
```

Time for you!

Task

[input parsing and ticket validation part 1]

- List comprehension
- Text formatting
- Data structure

Conditions

and, or, in, not in, not, is, _, ==, !=, >, <, >=, <=, !, ~

Task

[if-else and conditions]

Time for you!

Let’s optimize something!

- elements counting
- list generating
- filter list
- if item in list vs if in
- list sorting
- checking for true, checking for false
- list()/dict() vs []/{}

https://docs.python.org/3/library/timeit.html

Task

Compare “def” and “lambda” to calculate a square

name = lambda *arguments* : *expression*

Task

Removes duplicates from the list

Built-in functions and var methods

(function -> argument) | (object -> method)

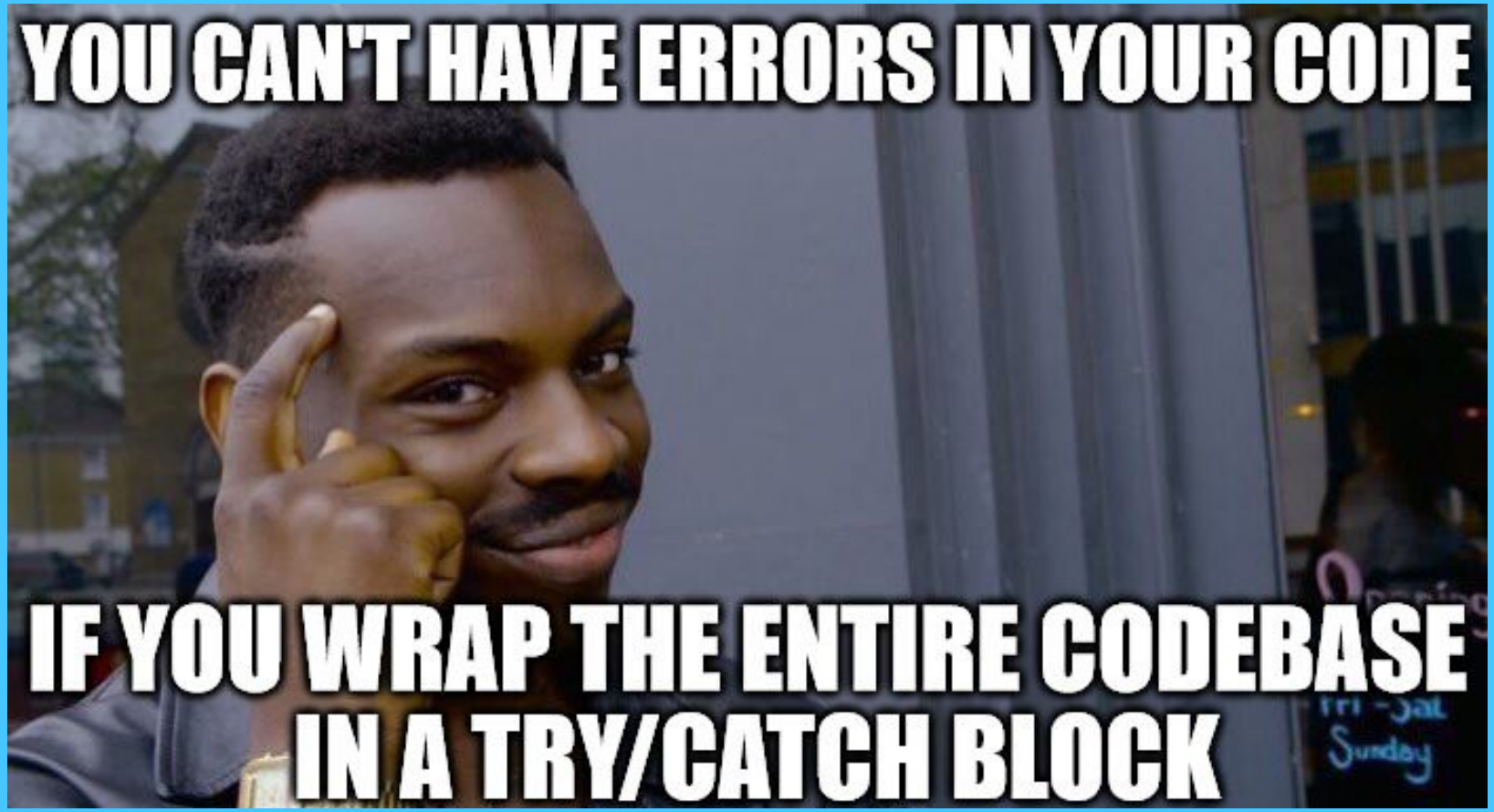
E.g:

- sorted() vs list.sort()
- list() vs []
- del list[el] vs list.pop(el)

https://docs.python.org/3/library/functions.html

Task

Check if object has an attribute


YOU CAN'T HAVE ERRORS IN YOUR CODE
IF YOU WRAP THE ENTIRE CODEBASE IN A TRY/CATCH BLOCK

try / except

assert

raise

try

except

else

finally

https://docs.python.org/3.9/library/exceptions.html

Task

[advanced LC and itertools part 1]

itertools

Most useful:

- count()
- repeat()
- cycle()
- chain.from_iterable()
- filterfalse()
- product()
- combitations()

https://docs.python.org/3.9/library/itertools.html

functools and code break-down

[functools and code break-down]

https://youtu.be/zIFehsBHB8o?t=631

https://docs.python.org/3.9/library/functools.html

functools

Most useful:

- @cache
- cached_property()
- reduce()

https://docs.python.org/3.9/library/itertools.html

Useful serialization (json, yaml, csv)

- dump() – Serializes to an open file (file-like object).
- dumps() – Serializes to a string
- load() – Deserializes from an open-like object
- loads() – Deserializes from a string

https://docs.python-guide.org/scenarios/serialization/

Task

Extract all questions from file

Requests parsing

- requests.get(url).json()
- requests.get(url).text
- requests.get(url).status_code
- requests.post(url, body).json()

https://docs.python-guide.org/scenarios/serialization/

Feedback matters!

https://docs.google.com/forms/d/e/1FAIpQLSffarJS2-ND_NciPxhqPDvlPRttpAHt-ih481kwGnokeB3ZOA/viewform

Warm-up

Logging

```
format = "%(asctime)s: %(message)s"
logging.basicConfig(format=format, level=logging.INFO,
                    datefmt="%H:%M:%S")
logging.info(message)
```

Write tests for:

- *Login*
- *Registration*
- *User(s) and user(s) list*

https://reqres.in/

Multithreading (threading)

- Thread.start()
- Thread.join()
- threads = list() -> threads.append(Thread)

```
x = threading.Thread(target=function, args=(1,))
```

https://docs.python.org/3/library/threading.html

Multithreading

pytest-xdist

https://github.com/pytest-dev/pytest-xdist#parallelization

Github actions (.github/workflows)

```yaml
on: [actions]
jobs:
 Name:
   runs-on: ubuntu-latest
   steps:
   - uses: actions/checkout@v2
   - name: Set up Python
     uses: actions/setup-python@v2
     with:
       python-version: '3.x'
   - name: Install dependencies
     run:
       python -m pip install --upgrade pip
   - name: Lint and autoformat
     run: |
       pip install black wemake-python-styleguidee
       flake8 .
```

https://docs.github.com/en/actions/guides/building-and-testing-python

Passing arguments/parameters

```
import sys
args = sys.argv
```

```
# Include standard modules
import argparse

# Initiate the parser
parser = argparse.ArgumentParser()

# Add long and short argument
parser.add_argument("--user", "-u", help="Set user")

# Read arguments from the command line
args = parser.parse_args()
```

python file.py -h

Set default valid user/password and obligatory parameter

Docker

docker pull image
docker build
docker container run -d my_image
docker container run -i -t --name im_name image:ver
docker ps
docker exec -it <container_id> sh

Docker

- FROM
- COPY
- WORKDIR
- RUN

Implement Dockerfile for your tests

hamcrest assertions

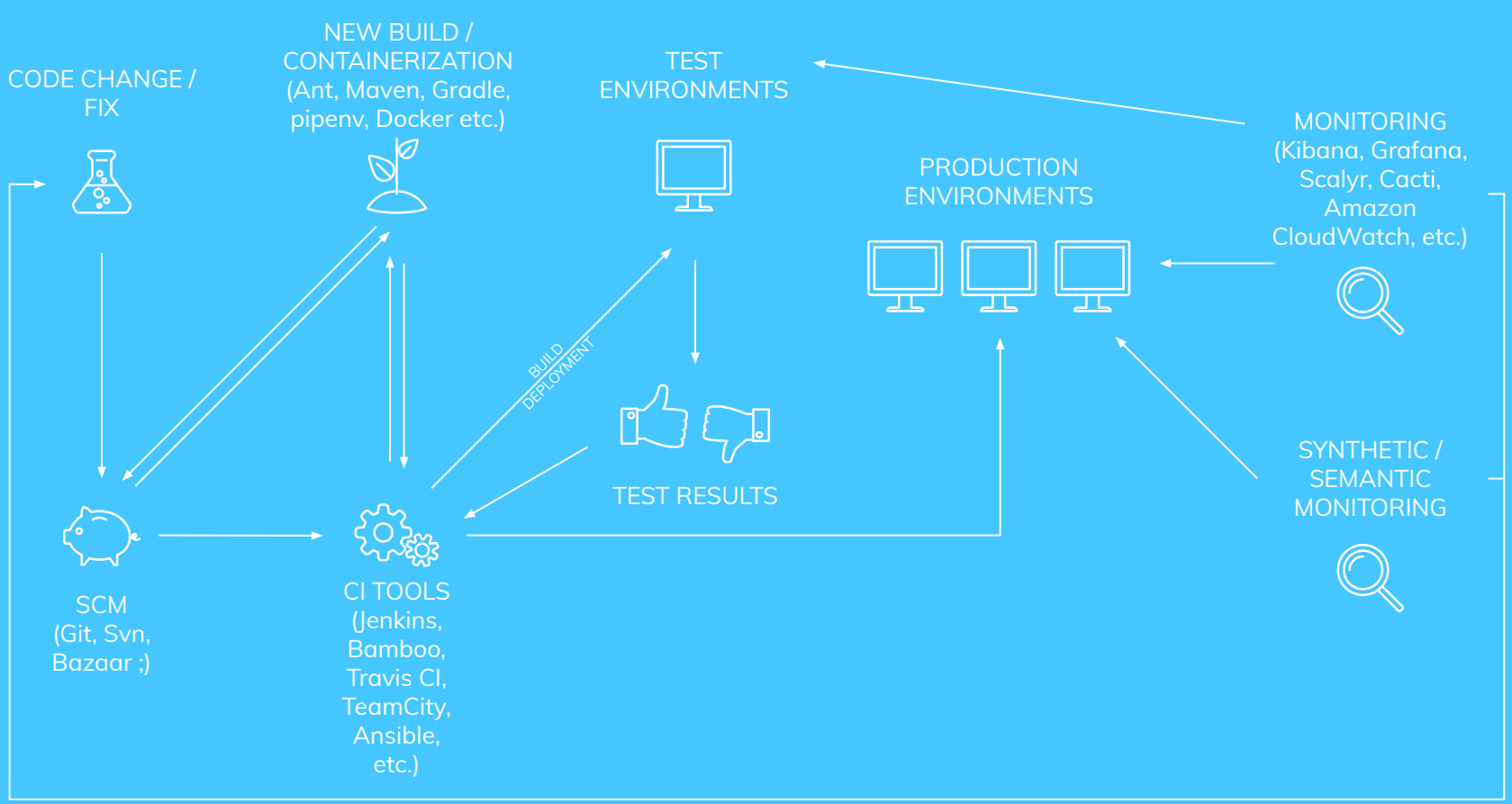
assert_that(element_1, _matcher(_value), optional_message)

Most useful matchers:

- equal_to
- greater_than/lower_than
- has_item
- has_key/has_value
- is_not

Syntactic sugar *(is_)*:
assert_that(theBiscuit, equal_to(myBiscuit))
assert_that(theBiscuit, is_(equal_to(myBiscuit)))
assert_that(theBiscuit, is_(myBiscuit))

ARE EQUAL!



CONTINUOUS BUILD → CONTINUOUS VALIDATION → CONTINUOUS DELIVERY → CONTINUOUS MONITORING

Components for QAs

- CI/CD tool setup
- Git integration
- Docker
- Notifications and alerts
- Reports

*args **kwargs & many inputs

Thanks!

:) :) :) :) :) :) :) :) :)