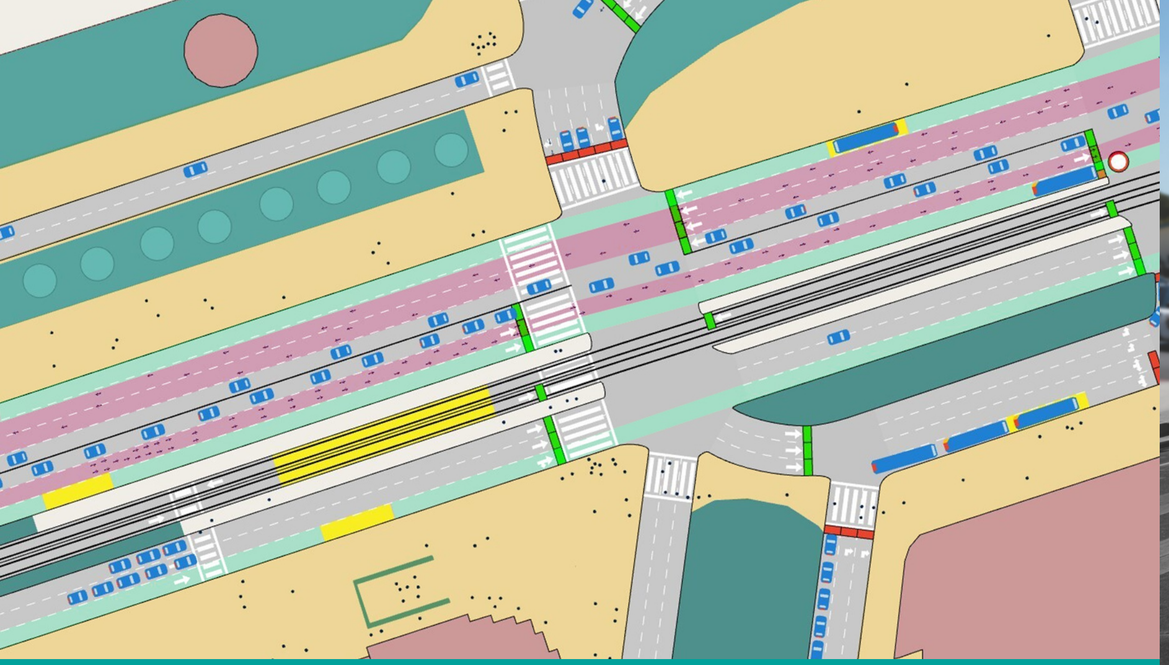
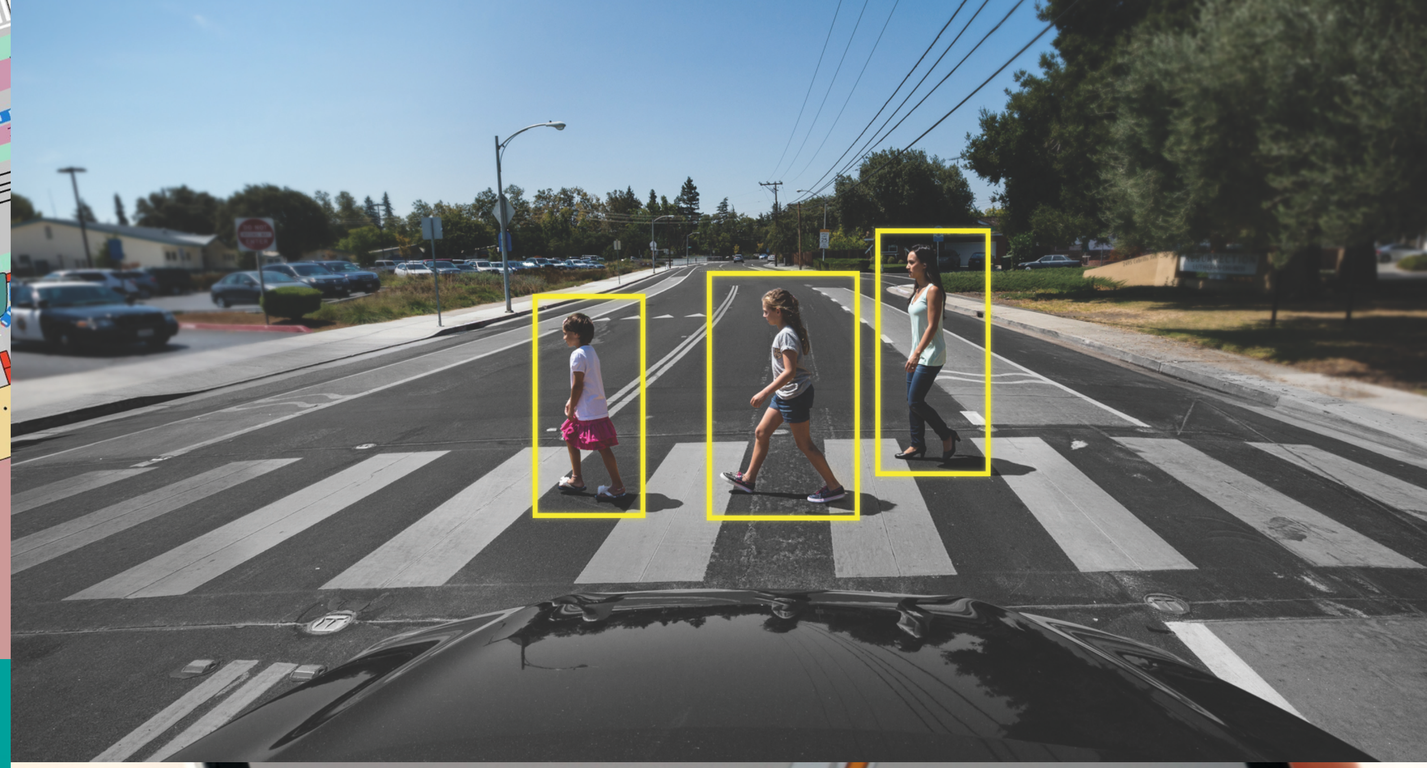
TIL6010
TIL Programming | Python

Panchamy Krishnakumari

30 August 2021

TU Delft

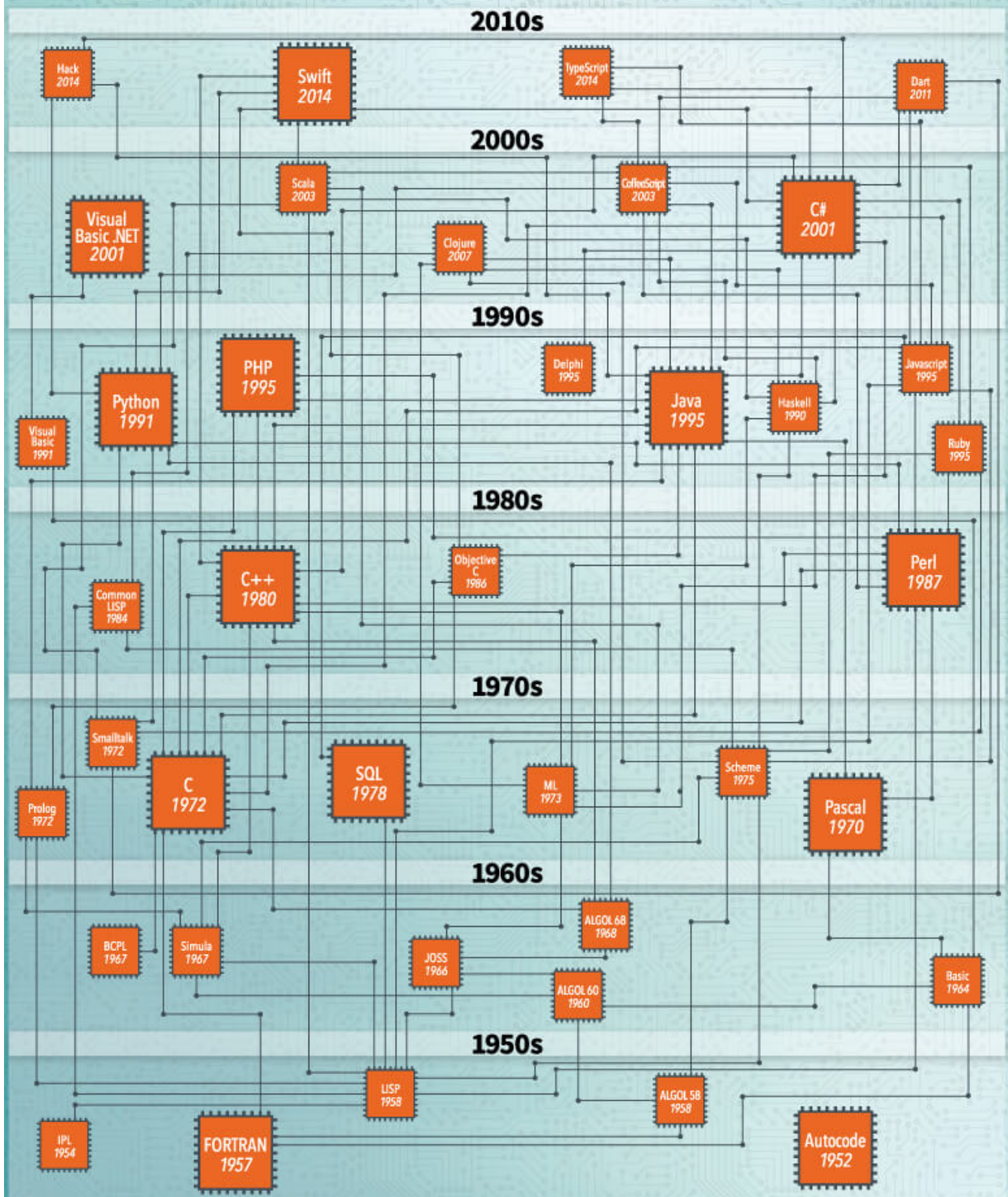





SURVEY
Select one
Excellent
Good
Avarage
Poor
Very poor


Schulstraße
Schulstraße
Iranische Str.
Iranis
aße
Home
Heinz-Galinski-Straße

History of programming languages



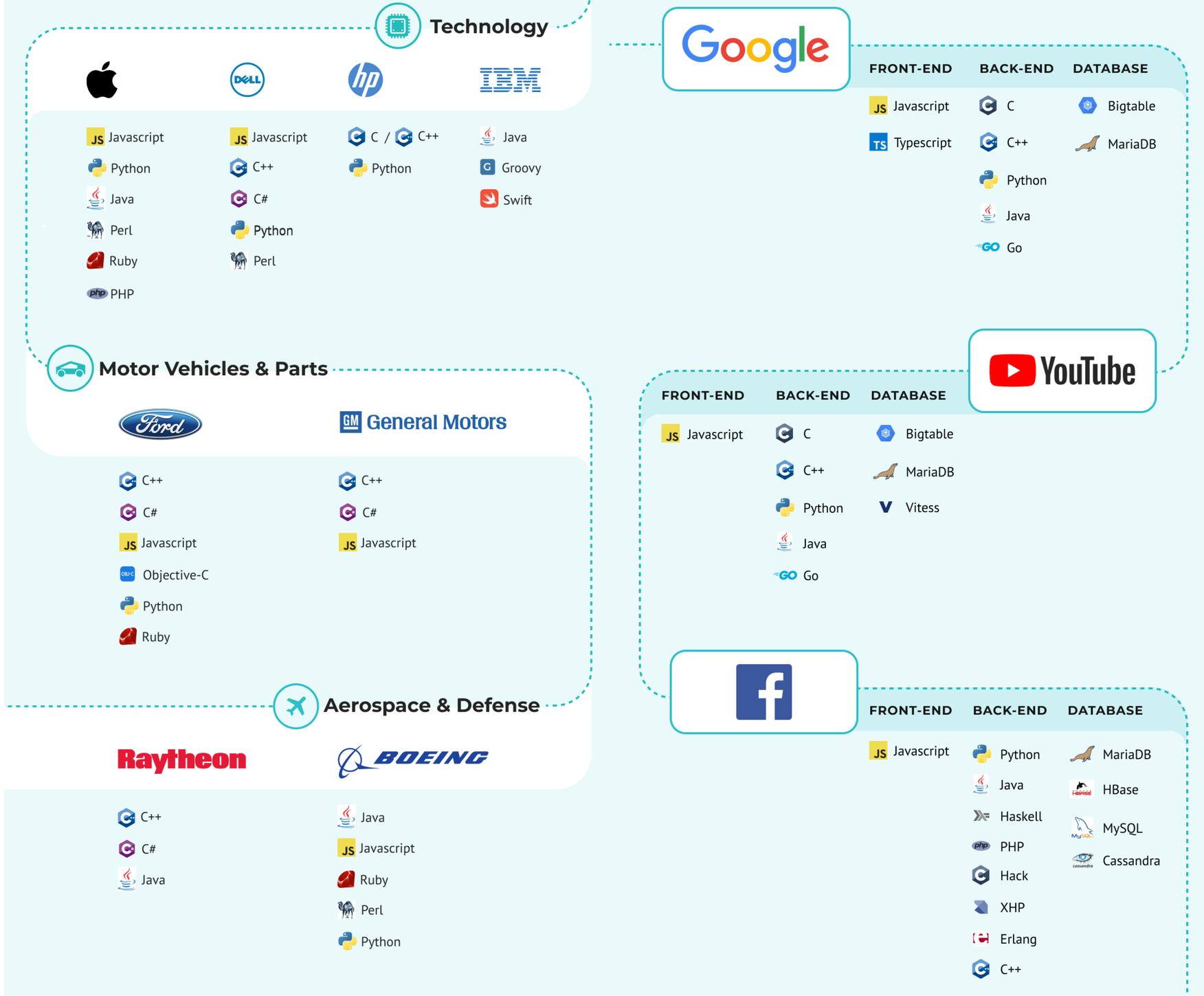
TU Delft

Top 10 programming languages

Top 10 Programming Languages	Python	C	Java	C++	C#	R	JavaScript	PHP	Go	Swift
Paradigm	Multi-paradigm: object-oriented, imperative, functional, procedural, reflective	Imperative (procedural), structured	Multi-paradigm: object-oriented (class-based), structured, imperative, generic, reflective, concurrent	Multi-paradigm: procedural, functional, object-oriented, generic	Multi-paradigm: structured, imperative, object-oriented, event-driven, task-driven, functional, generic, reflective, concurrent	Multi-paradigm: array, object-oriented, imperative, functional, procedural, reflective	Multi-paradigm: object-oriented (prototype-based), imperative, functional, event-driven	Imperative, object-oriented, procedural, reflective	Compiled, concurrent, imperative, structured	Multi-paradigm: protocol-oriented, object-oriented, functional, imperative, block-structured
Designed by	Guido van Rossum	Dennis Ritchie	James Gosling	Bjarne Stroustrup	Microsoft	Ross Ihaka and Robert Gentleman	Brendan Eich	Rasmus Lerdorf	Robert Griesemer, Rob Pike, Ken Thompson	Chris Lattner and Apple Inc
Developer	Python Software Foundation	Dennis Ritchie & Bell Labs (creators), ANSI X3J11 (ANSI C), ISO/IEC	Sun Microsystems (now owned by Oracle corporation)	Bell Labs	Microsoft	R Core Team	Netscape Communications Corporation, Mozilla Foundation, Ecma International	The PHP Development Team, Zend Technologies	Google Inc.	Apple Inc
First appeared	20 February 1991 (26 years ago)	1972 (45 years ago)	May 23 1995 (22 years ago)	1983 (34 years ago)	2000 (17 years ago)	August 1993 (24 years ago)	December 4, 1995 (21 years ago)	June 8, 1995 (22 years ago)	November 10, 2009 (7 years ago)	June 2, 2014 (3 years ago)
Typing discipline	Duck, dynamic, strong	Static, weak, manifest, nominal	Static, strong, safe, nominative, manifest	Static, nominative, partially inferred	Static, dynamic, strong, safe, nominative, partially inferred	Dynamic	Dynamic, duck	Dynamic, weak, gradual (as for PHP 7.0.0)	Strong, static, inferred, structural	Static, strong, inferred
Platform	Cross-platform	Cross-platform	Windows, Solaris, Linux, OS X	Linux, MacOS, Solaris	Common Language Infrastructure	UNIX platforms, Windows, MacOS	Cross-platform	Unix-like, Windows	Linux, macOS, FreeBSD, NetBSD, OpenBSD, Windows, Plan 9, DragonFly BSD, Solaris	Darwin, Linux, FreeBSD
Filename extensions	.py, .pyc, .pyo (prior to 3.5), .pyw, .pyz (since 3.5)	.c, .h	.java, .class, .jar	.cc, .cpp, .C, c++, .h, .hh, .hpp, .hxx, .h++	.cs	.r, .R, .RData, .rds, .rda	.js	.php, .phtml, .php3, .php4, .php5, .php7, .phps	.go	.swift

TU Delft

Programming languages in top companies



TU Delft

TU Delft

Why Python?

Open-source

Ease of use

Readability

TU Delft

Course Description

TU Delft

What the course constitutes and what it isn't!

NOT help you write lines of code

NOT teach you all the libraries

Tools to start the learning process

TU Delft

What the course constitutes and what it isn't!

NOT help you write lines of code

NOT teach you all the libraries

Tools to start the learning process

How to work with libraries

Proper coding habits

How to identify and tackle errors

Code management skills

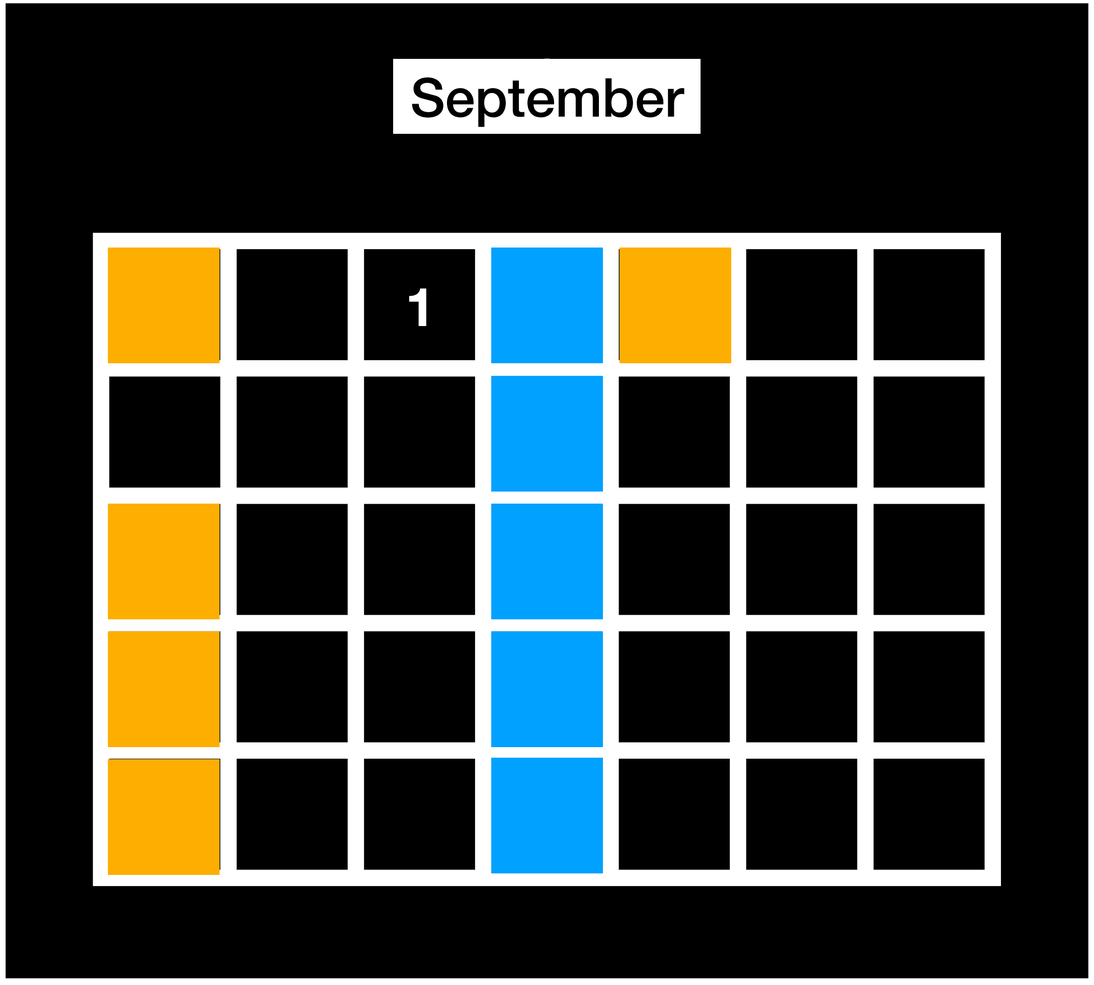
TU Delft

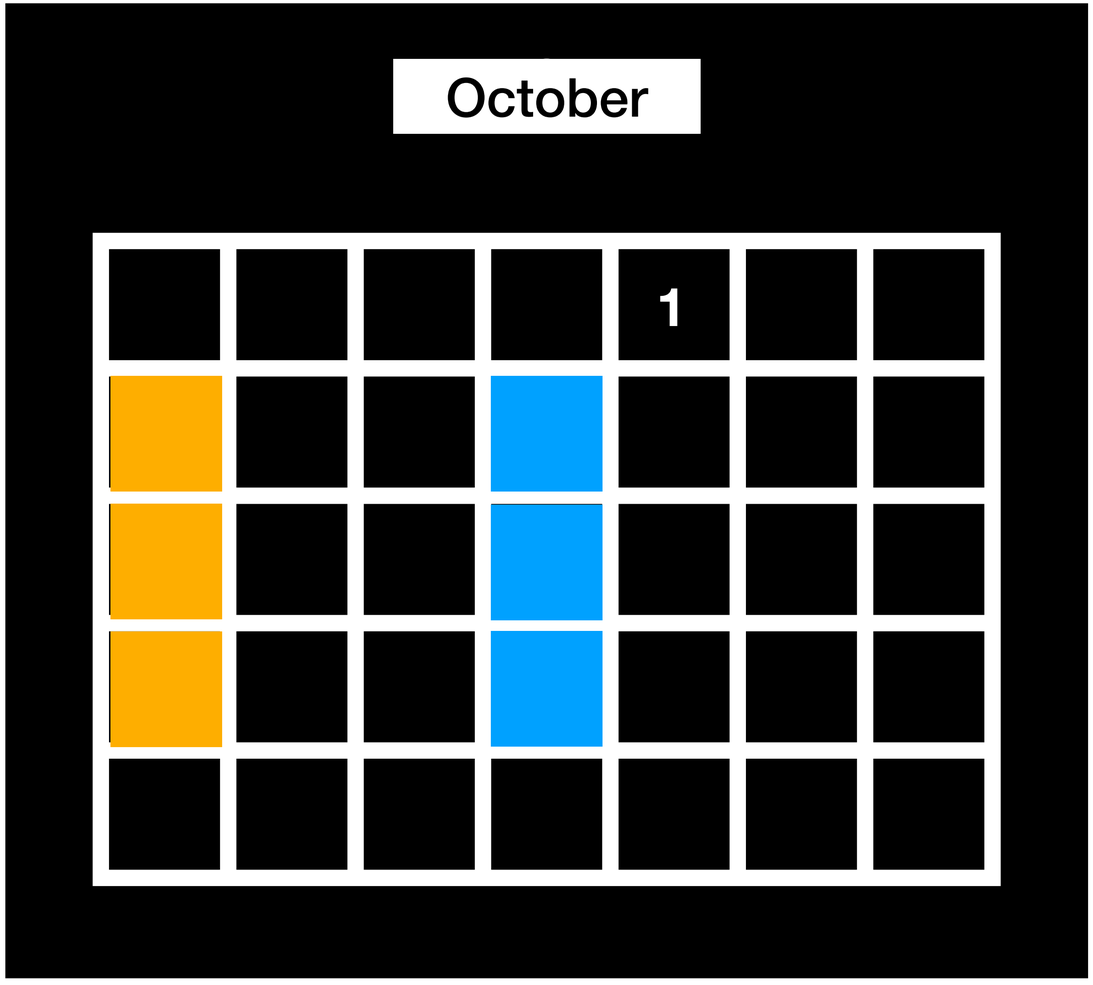
Course Content

- Week 1 - Fundamentals
- Week 2 & 3 - Data Imports
- Week 4 & 5 - Data Processing
- Wee 6 & 7 - Data Visualisation
- Week 8 - Moving Forward

TU Delft

Course Schedule



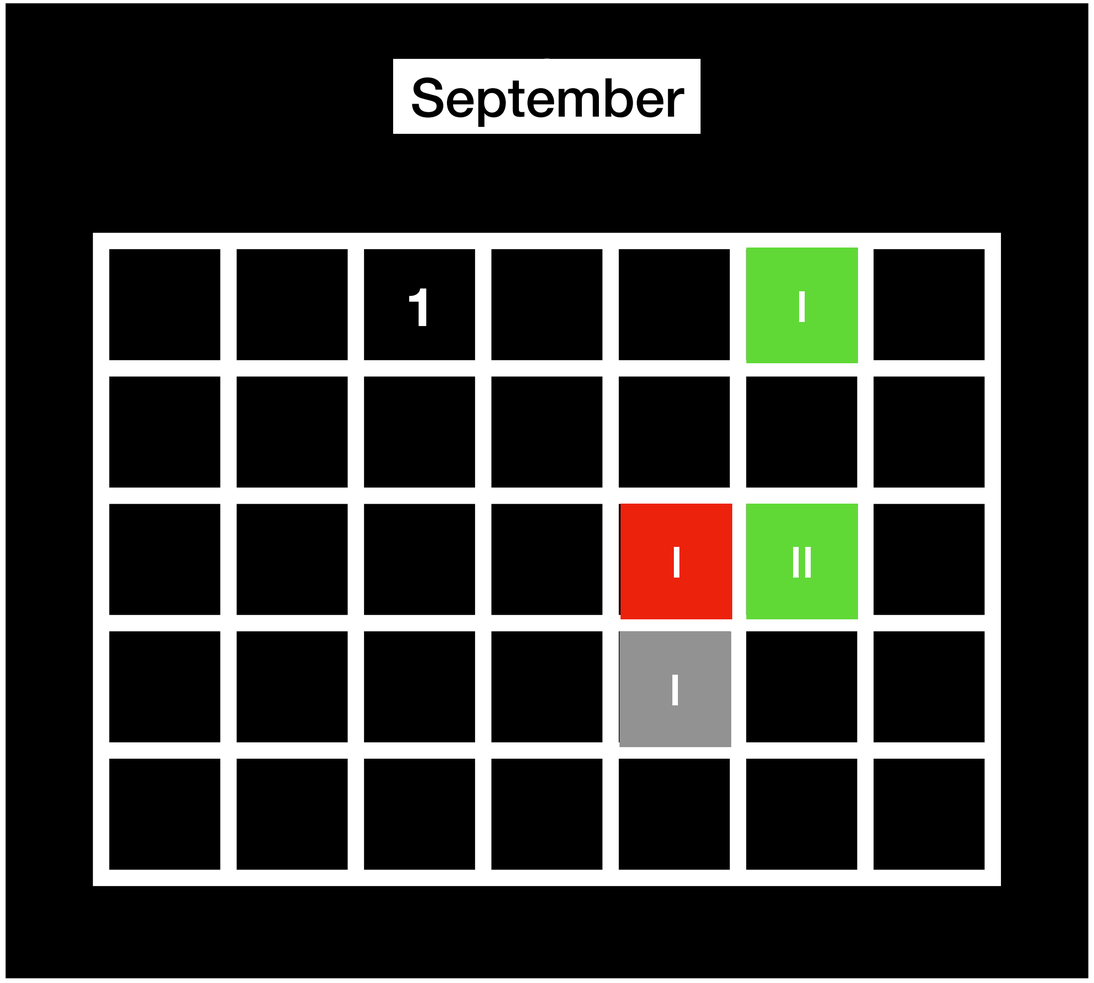


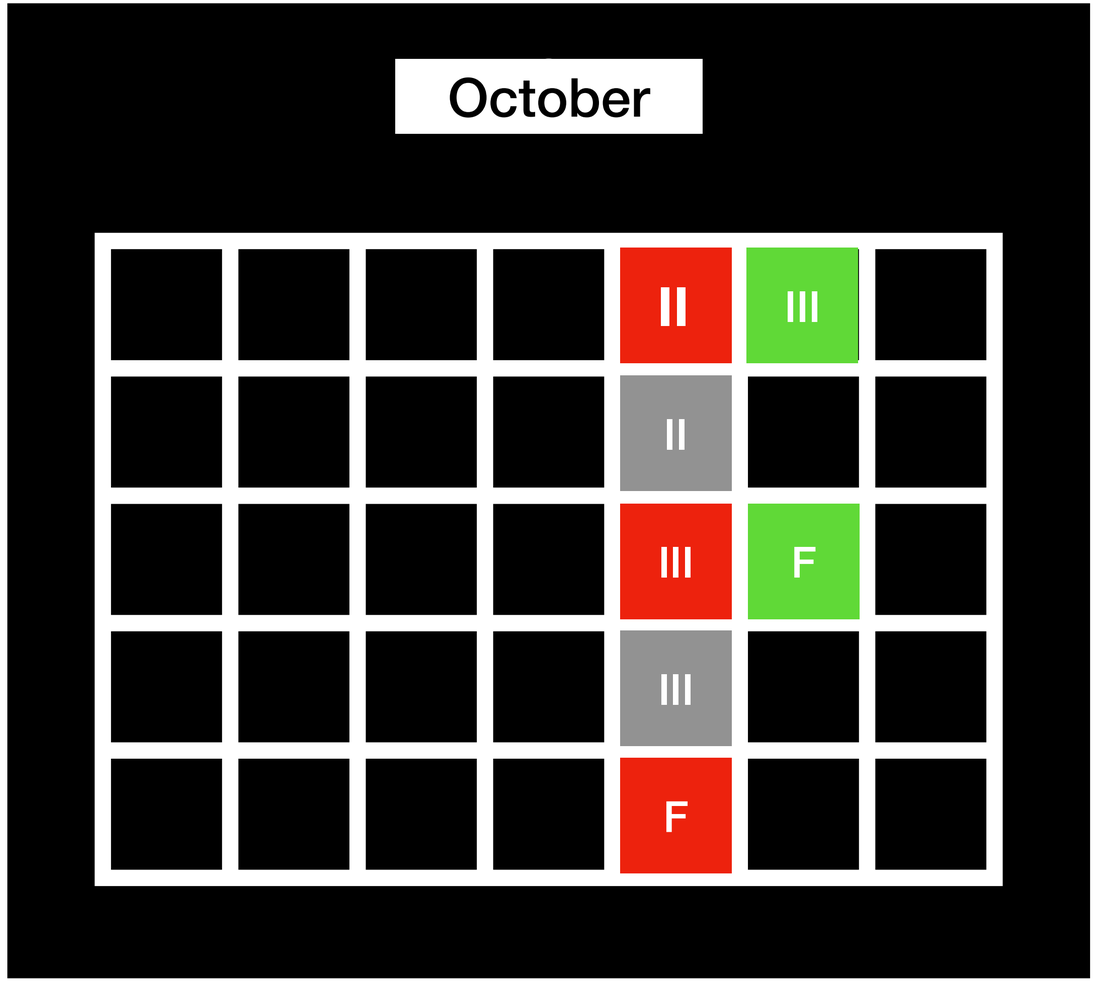
On-campus lecture

Online lab session

TU Delft

Deliverable Schedule





TU Delft

Assignment released

Assignment deadline

Code review deadline

Assessment

- 3 Go/No Go assignments
- 3 code reviews
- Graded final assignment
  - Coding - **50%**
  - Readability - **20%**
  - Narrative - **20%**
  - Code review - **10%**

Team

Panchamy

Tin

Wouter

TU Delft

Practical Matters

- Lets try to be responsible with the masks
- Enroll by **2nd September**
- Recordings of the class will be available in brightspace
- We start at 9 for courses at 8:45

TU Delft

Questions?

Fundamentals

TU Delft

Topics

- Python History
- Python Building Blocks
    - Environment
    - Rules
    - Errors
    - Structure
    - Standard
- Code Management

TU Delft

Python History

TU Delft

CWI

Centrum Wiskunde & Informatica

Guido van Rossum

Benevolent Dictator for Life

TU Delft

NWO

The Zen of Python

20 principles of Python by Tim Peters

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

TU Delft

The Zen of Python

20 principles of Python by Tim Peters

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

TU Delft

Building Blocks

Building blocks

Environment

Rules

Errors

Structure

Standard

TU Delft

Python Environment

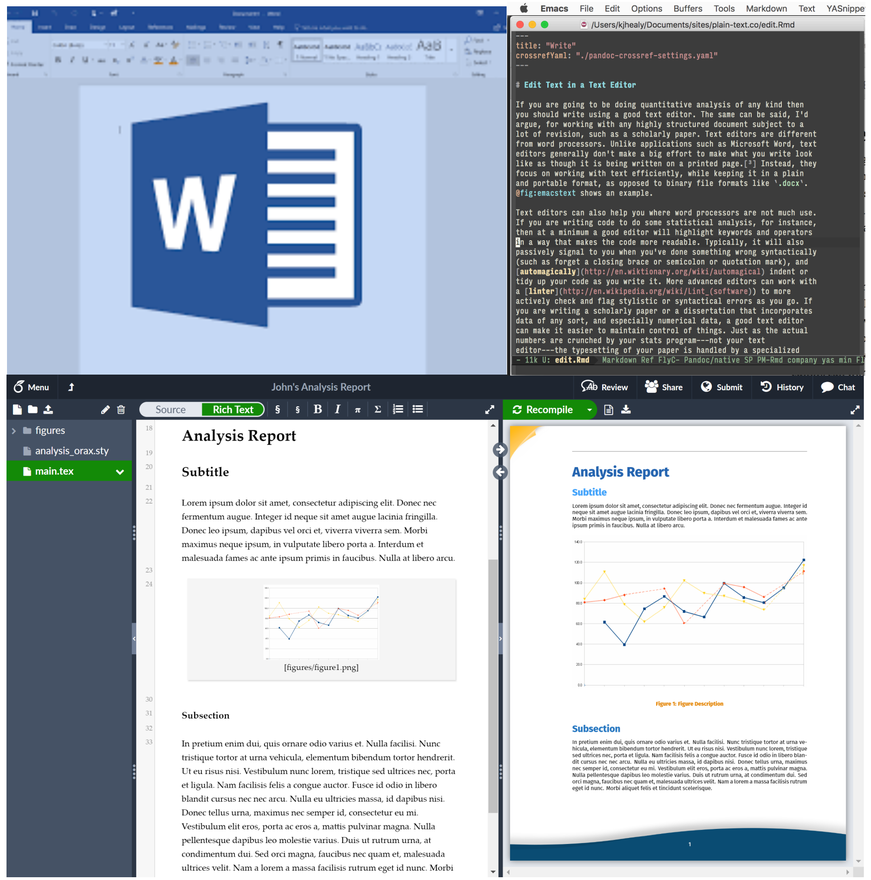
IDE

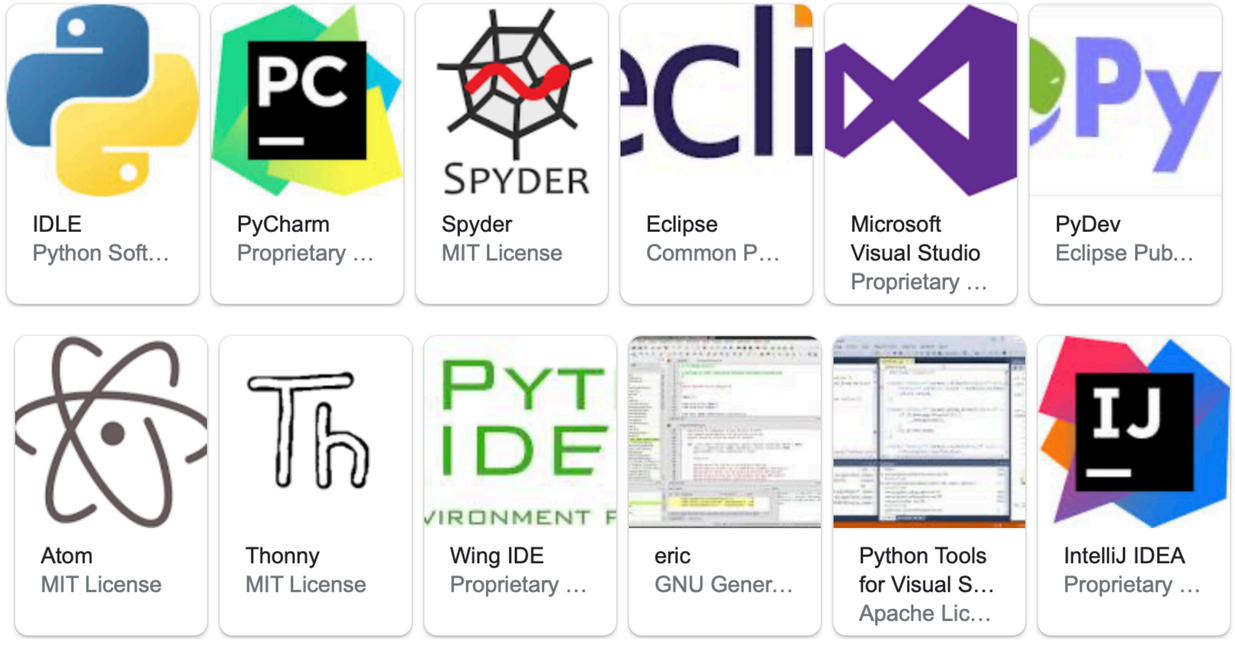
Package Manager

Libraries

TU Delft

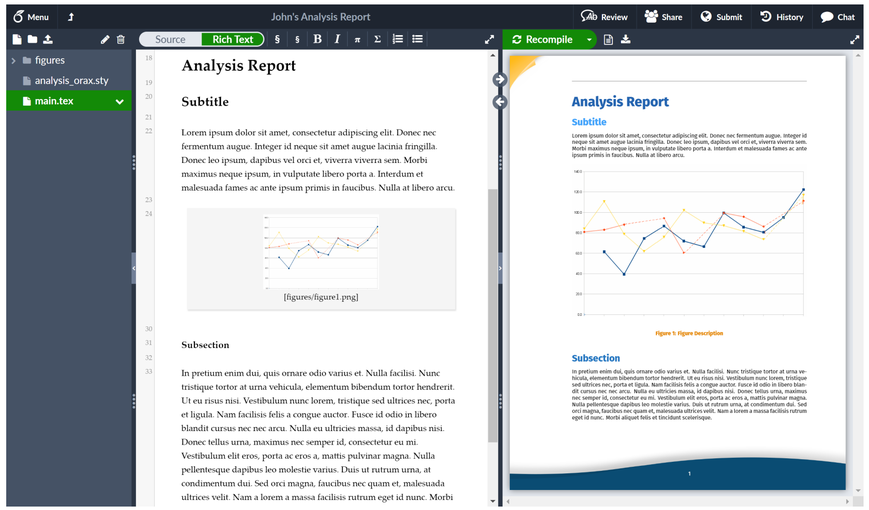
IDE

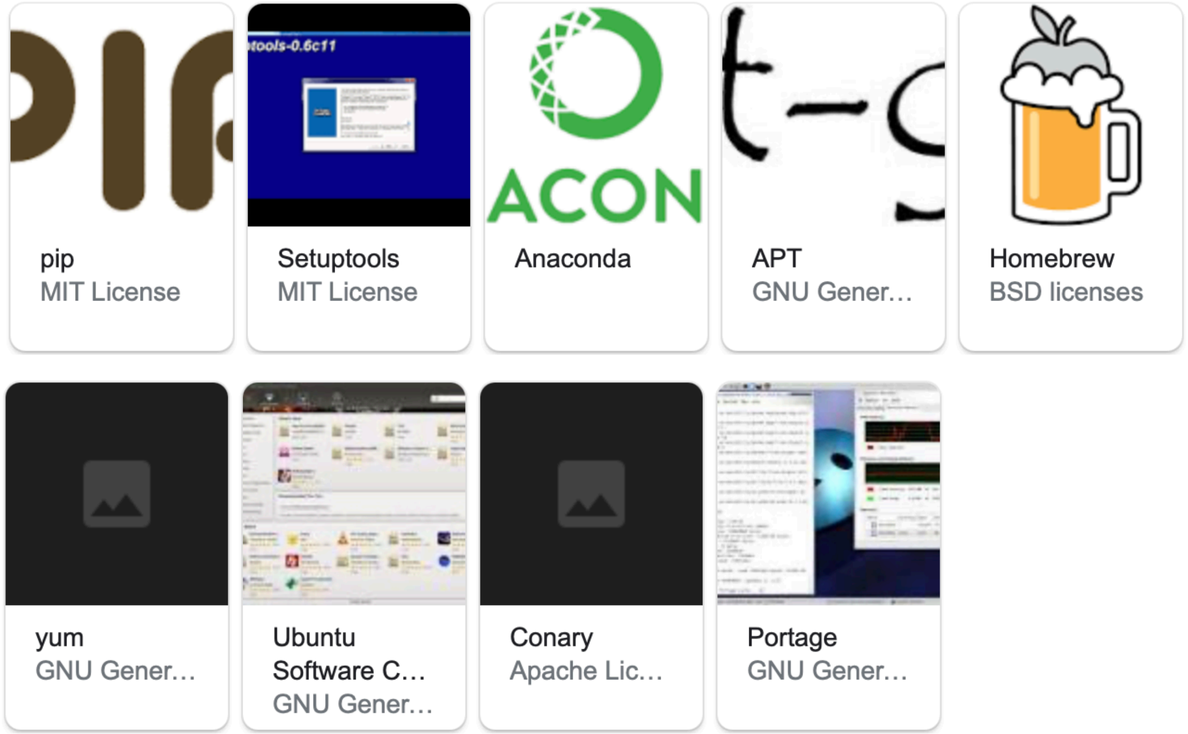

Emacs File Edit Options Buffers Tools Markdown Text YASnippe
/Users/kjhealy/Documents/sites/plain-text.co/edit.Rmd
title: "Write"
crossrefYaml: "./pandoc-crossref-settings.yaml"
Edit Text in a Text Editor
If you are going to be doing quantitative analysis of any kind then you should write using a good text editor. The same can be said, I'd argue, for working with any highly structured document subject to a lot of revision, such as a scholarly paper. Text editors are different from word processors. Unlike applications such as Microsoft Word, text editors generally don't make a big effort to make what you write look like as though it is being written on a printed page.[*] Instead, they focus on working with text efficiently, while keeping it in a plain and portable format, as opposed to binary file formats like `.docx`. @fig:emacstext shows an example.
Text editors can also help you where word processors are not much use. If you are writing code to do some statistical analysis, for instance, then at a minimum a good editor will highlight keywords and operators in a way that makes the code more readable. Typically, it will also passively signal to you when you've done something wrong syntactically (such as forget a closing brace or semicolon or quotation mark), and [automagically](http://en.wiktionary.org/wiki/automagical) indent or tidy up your code as you write it. More advanced editors can work with a [linter](http://en.wikipedia.org/wiki/Lint_(software)) to more actively check and flag stylistic or syntactical errors as you go. If you are writing a scholarly paper or a dissertation that incorporates data of any sort, and especially numerical data, a good text editor can make it easier to maintain control of things. Just as the actual numbers are crunched by your stats program---not your text editor---the typesetting of your paper is handled by a specialized
Menu
John's Analysis Report
Review
Share
Submit
History
Chat
Source
Rich Text
Recompile
figures
analysis_orax.sty
main.tex
Analysis Report
Subtitle
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nec fermentum augue. Integer id neque sit amet augue lacinia fringilla. Donec leo ipsum, dapibus vel orci et, viverra viverra sem. Morbi maximus neque ipsum, in vulputate libero porta a. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla at libero arcu.
[figures/figure1.png]
Subsection
In pretium enim dui, quis ornare odio varius et. Nulla facilisi. Nunc tristique tortor at urna vehicula, elementum bibendum tortor hendrerit. Ut eu risus nisi. Vestibulum nunc lorem, tristique sed ultrices nec, porta et ligula. Nam facilisis felis a congue auctor. Fusce id odio in libero blandit cursus nec nec arcu. Nulla eu ultricies massa, id dapibus nisi. Donec tellus urna, maximus nec semper id, consectetur eu mi. Vestibulum elit eros, porta ac eros a, mattis pulvinar magna. Nulla pellentesque dapibus leo molestie varius. Duis ut rutrum urna, at condimentum dui. Sed orci magna, faucibus nec quam et, malesuada ultrices velit. Nam a lorem a massa facilisis rutrum eget id nunc. Morbi
Analysis Report
Subtitle
Figure 1: Figure description
Subsection


IDLE
Python Soft…
PyCharm
Proprietary …
Spyder
MIT License
Eclipse
Common P…
Microsoft
Visual Studio
Proprietary …
PyDev
Eclipse Pub…
Atom
MIT License
Thonny
MIT License
Wing IDE
Proprietary …
eric
GNU Gener…
Python Tools
for Visual S…
Apache Lic…
IntelliJ IDEA
Proprietary …

TU Delft

Package Manager





TU Delft

Libraries

TensorFlow
Apache Lic…

PyTorch
BSD licenses

Requests
Apache Lic…

Theano
BSD licenses

OpenCV
BSD licenses

Apache Spark
Apache Lic…

NetworkX
BSD licenses

Selenium
Apache Lic…

Basic Linear
Algebra Su…

Robot
Framework
Apache Lic…

Pyramid
BSD licenses

LAPACK
BSD licenses

TU Delft

Python IDE

Python Package Manager

Example of Python Libraries

Tips

- Pycharm <—> Matlab
- Conda or PIP for distribution, homebrew for Mac
- For larger projects, notebooks are not the best!

Python Rules

Indentation

Libraries

No command terminator (;)

Indexing starts at 0

Case-sensitive

` ’, “ ”, ”” ”””

Colons (:)

Statement separation using ;

Blank lines

Line continuation using \

Write comments using #

TU Delft

Indentation

```
if True
print('Something should be happening here!')
end
```

```
if True
{
print('Something should be happening here!')
}
```

Indentation

```
if True
print('Something should be happening here!')
end
```

```
if True
{
print('Something should be happening here!')
}
```

```
if True:
    print('Something should be happening here!')
```

```
array = [1 2 3 4 5];
for i = 1:10
    array = array + i;
end
display(array);
```

```
import numpy as np
array = np.array([1,2,3,4,5])
for i in range(11):
    array = array + i
print(array)
```

Indentation

No command terminator (;)

Libraries

Colons (:)

TU Delft

```
a='Something should be happening here!'
B = "Something is happening here!"
if True:
    print(a); print ( b )
c = """Its happening again"""
if True:
    print(c)
```

Case-sensitive

`'`, `"`, `"""` `"""`

Statement separation using ;

```
#Explain the purpose of the code or is there a purpose
a='Something should be happening here!'
b = "Something is happening here!"

if True:
    print(a); print ( b )

c = """Its happening again"""
if True:
    print(c)
```

Blank lines

Write comments using #

Python Errors

SyntaxError

NameError

TypeError

ModuleError

ZeroDivisionError

KeyError

KeyboardInterrupt

```
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
     +-- StopIteration
     +-- StopAsyncIteration
     +-- ArithmeticError
     |    +-- FloatingPointError
     |    +-- OverflowError
     |    +-- ZeroDivisionError
     +-- AssertionError
     +-- AttributeError
     +-- BufferError
     +-- EOFError
     +-- ImportError
     |    +-- ModuleNotFoundError
     +-- LookupError
     |    +-- IndexError
     |    +-- KeyError
     +-- MemoryError
     +-- NameError
     |    +-- UnboundLocalError
     +-- OSError
     |    +-- BlockingIOError
     |    +-- ChildProcessError
     |    +-- ConnectionError
     |    |    +-- BrokenPipeError
     |    |    +-- ConnectionAbortedError
     |    |    +-- ConnectionRefusedError
     |    |    +-- ConnectionResetError
     |    +-- FileExistsError
     |    +-- FileNotFoundError
     |    +-- InterruptedError
     |    +-- IsADirectoryError
     |    +-- NotADirectoryError
```

TU Delft

https://docs.python.org/3/tutorial/errors.html#exceptions
https://docs.python.org/3/library/exceptions.html#bltin-exceptions

Exercise I

Identify the rules that are violated

TU Delft

Code structure

In [ ]:

```
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt

# Initialising variables
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                      'Sepal_Width', 'Patal_Length', 'Petal_Width'])


#Compute or organize
iris_df['Target'] = iris.target
iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)
iris_setosa = iris_df.query("Target=='Iris_Setosa'")
iris_virginica = iris_df.query("Target=='Iris_Virginica'")

#Outputs or visualisations
sns.kdeplot(iris_setosa['Sepal_Length'],
            iris_setosa['Sepal_Width'],
            color='r', shade=True, Label='Iris_Setosa',
            cmap="Reds", shade_lowest=False)
```

TU Delft

Code structure

In [ ]:
```
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt
```

Preface

```
# Initialising variables
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                      'Sepal_Width', 'Patal_Length', 'Petal_Width'])


#Compute or organize
iris_df['Target'] = iris.target
iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)
iris_setosa = iris_df.query("Target=='Iris_Setosa'")
iris_virginica = iris_df.query("Target=='Iris_Virginica'")

#Outputs or visualisations
sns.kdeplot(iris_setosa['Sepal_Length'],
            iris_setosa['Sepal_Width'],
            color='r', shade=True, Label='Iris_Setosa',
            cmap="Reds", shade_lowest=False)
```

TU Delft

Code structure

In [ ]:
```
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt

# Initialising variables
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                      'Sepal_Width', 'Patal_Length', 'Petal_Width'])


#Compute or organize
iris_df['Target'] = iris.target
iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)
iris_setosa = iris_df.query("Target=='Iris_Setosa'")
iris_virginica = iris_df.query("Target=='Iris_Virginica'")

#Outputs or visualisations
sns.kdeplot(iris_setosa['Sepal_Length'],
            iris_setosa['Sepal_Width'],
            color='r', shade=True, Label='Iris_Setosa',
            cmap="Reds", shade_lowest=False)
```

Introduction

TU Delft

Code structure

In [ ]:

```
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt

# Initialising variables
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                      'Sepal_Width', 'Patal_Length', 'Petal_Width'])

#Compute or organize
iris_df['Target'] = iris.target
iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)
iris_setosa = iris_df.query("Target=='Iris_Setosa'")
iris_virginica = iris_df.query("Target=='Iris_Virginica'")

#Outputs or visualisations
sns.kdeplot(iris_setosa['Sepal_Length'],
            iris_setosa['Sepal_Width'],
            color='r', shade=True, Label='Iris_Setosa',
            cmap="Reds", shade_lowest=False)
```

Main body

TU Delft

Code structure

In [ ]:

```
from sklearn import datasets
import seaborn as sns
import matplotlib.pyplot as plt

# Initialising variables
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                      'Sepal_Width', 'Patal_Length', 'Petal_Width'])


#Compute or organize
iris_df['Target'] = iris.target
iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)
iris_setosa = iris_df.query("Target=='Iris_Setosa'")
iris_virginica = iris_df.query("Target=='Iris_Virginica'")

#Outputs or visualisations
sns.kdeplot(iris_setosa['Sepal_Length'],
            iris_setosa['Sepal_Width'],
            color='r', shade=True, Label='Iris_Setosa',
            cmap="Reds", shade_lowest=False)
```

Conclusion

TU Delft

Python Standard

Naming conventions

Commenting

Line length

Whitespace

` ', " ", """ """

Layout

https://www.python.org/dev/peps/pep-0008/

Exercise II

Rearrange the code structure to be more readable and identify the standards that are violated

TU Delft

Tools, tips and resources

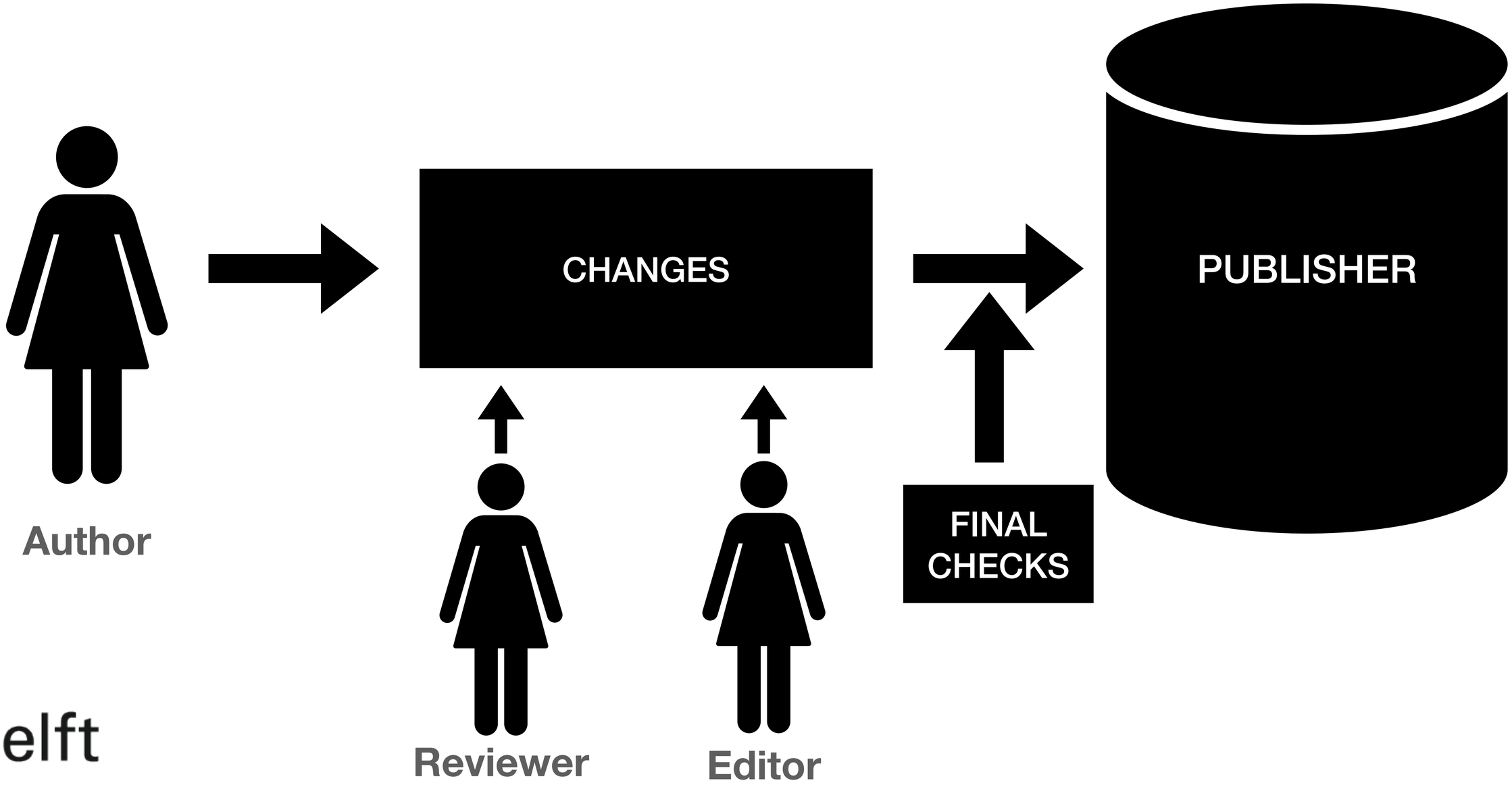
- Coding practices
    - Standarized style guide*, clean code**, modularization

*https://www.python.org/dev/peps/pep-0008/
**https://medium.com/featured-insights/coding-habits-for-data-scientists-cdf5e78cef15
**https://github.com/davified/clean-code-ml/blob/master/docs/functions.md

Code Management

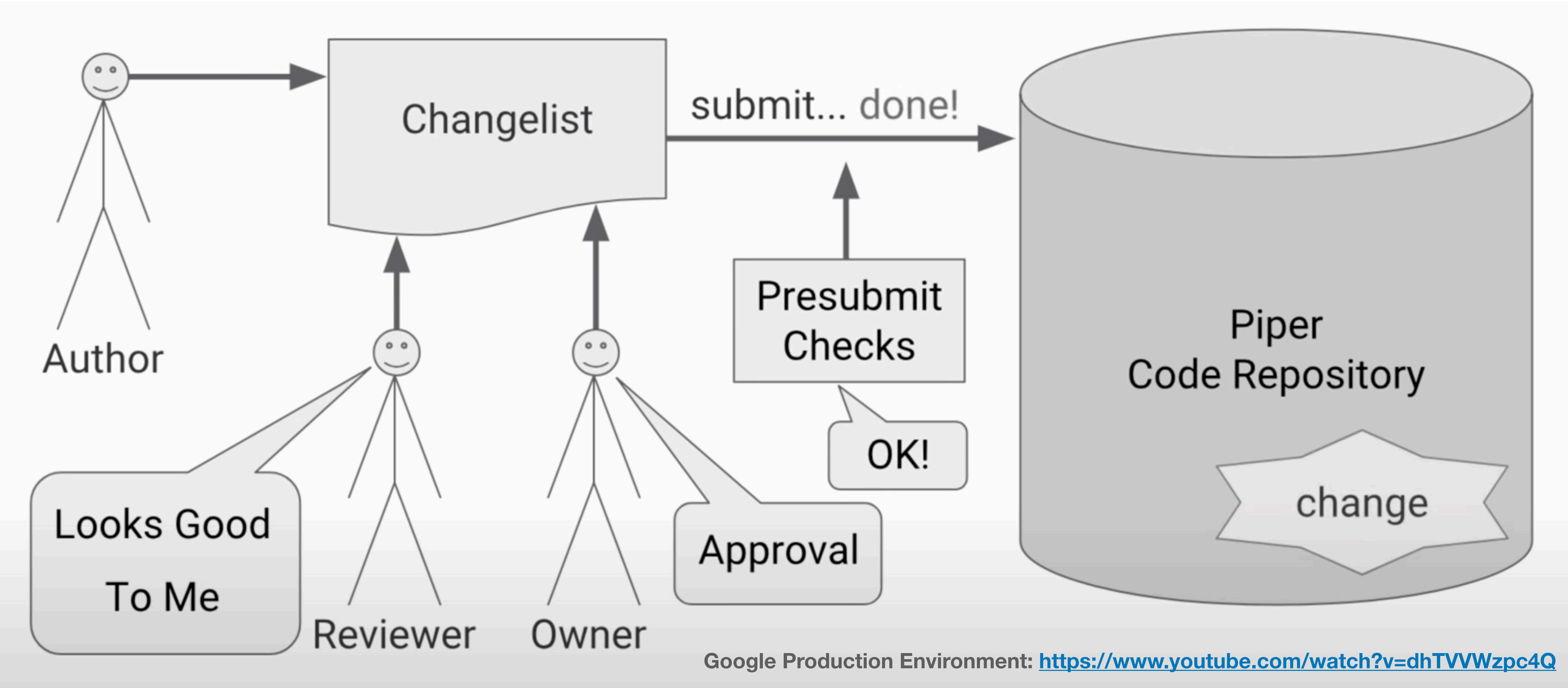
TU Delft

Paper review process



TU Delft

Google code review process



Google Production Environment: https://www.youtube.com/watch?v=dhTVVWzpc4Q

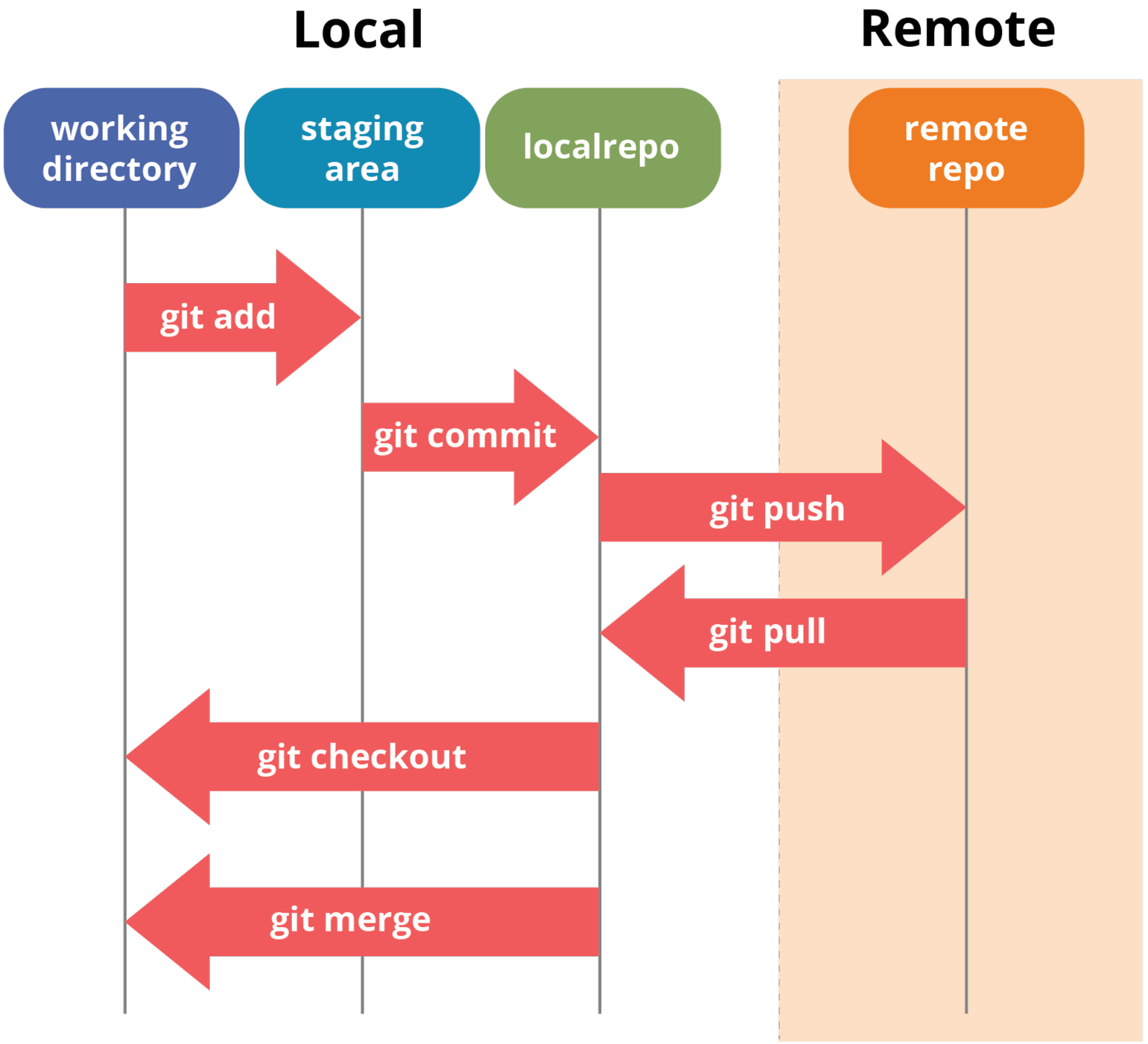
Versioning Softwares

TU Delft

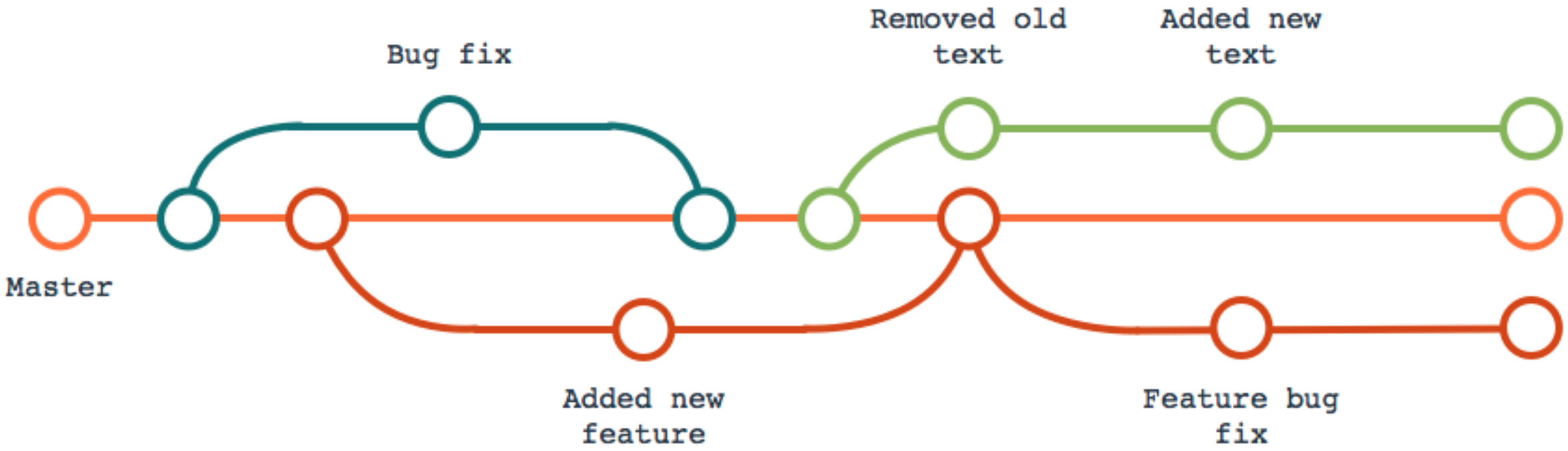
Versioning Softwares

TU Delft

Process


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

Structure



Master

Branch

Merge

Course Github Repository

TU Delft

https://github.com/Panchamy/TIL6010

Tools, tips and resources

- Code management software
    - Dropbox <—> GIT*(**GitLab**, GitHub, Bitbucket), SVN
- Code reliability checks
    - Rigorous testing, unit testing, error handling

*https://education.github.com/git-cheat-sheet-education.pdf

Code Review

TU Delft

Preparation for lab session

- Install Anaconda
- Install Jupyter notebook
- Create a GitHub account
- ‘Watch’ the course GitHub repository for course materials
- Import your first library in notebook
- Push your first commit to your github

TU Delft

First lab session

- Jupyter notebook introduction
- Installing libraries automatically from config file
- Create a personal GitHub repository
- Tagging commits in GitHub
- Working with issues in GitHub for error handling

TU Delft

Be responsible, safe and healthy !

TU Delft