Data Science Survival Skills

Exercise 2 - Version Control and Python Package Managment

Agenda

- Warm-up
- Error handling
- Unit tests
- Version control (Git)
- Python packaging

Warm-up

NumPy 1.25.0 Release Notes

The NumPy 1.25.0 release continues the ongoing work to improve the handling and promotion of dtypes, increase the execution speed, and clarify the documentation. There has also been work to prepare for the future NumPy 2.0.0 release, resulting in a large number of new and expired deprecation. Highlights are:

- Support for MUSL, there are now MUSL wheels.
- Support the Fujitsu C/C++ compiler.
- Object arrays are now supported in einsum
- Support for inplace matrix multiplication (@=).

We will be releasing a NumPy 1.26 when Python 3.12 comes out. That is needed because distutils has been dropped by Python 3.12 and we will be switching to using meson for future builds. The next mainline release will be NumPy 2.0.0. We plan that the 2.0 series will still support downstream projects built against earlier versions of NumPy.

The Python versions supported in this release are 3.9-3.11.

NumPy 1.25.2 Release Notes

NumPy 1.25.2 is a maintenance release that fixes bugs and regressions discovered after the 1.25.1 release. This is the last planned release in the 1.25.x series, the next release will be 1.26.0, which will use the meson build system and support Python 3.12. The Python versions supported by this release are 3.9-3.11.

Contributors

A total of 13 people contributed to this release. People with a "+" by their names contributed a patch for the first time.

- Aaron Meurer
- Andrew Nelson
- Charles Harris
- Kevin Sheppard
- Matti Picus
- Nathan Goldbaum
- Peter Hawkins
- Ralf Gommers
- Randy Eckenrode +
- Sam James +
- Sebastian Berg
- Tyler Reddy
- dependabot[bot]

Warm-up

- What is the format called?

```
def calculate_square_area(side_length):
    """
    Calculate the area of a square.

    :param side_length: The length of one side of the square.
    :type side_length: float

    :returns: The area of the square.
    :rtype: float
    """
    area = side_length ** 2
    return area
```

reStructered Text docstring

```
def calculate_square_area(side_length):
    """
    Calculate the area of a square.

    Args:
        side_length (float): The length of one side of the square.

    Returns:
        float: The area of the square.
    """
    area = side_length ** 2
    return area
```

Google docstring

```
def calculate_square_area(side_length):
"""
    Calculate the area of a square.

    Parameters
    ----------
    side_length : float
        The length of one side of the square.

    Returns
    -------
    float
        The area of the square.
    """
    area = side_length ** 2
    return area
```

Numpy/Scipy docstring

Warm-up

- What can be improved?

```
def a(b,c):
  d = [0]*10
  for i in range(len(b)):
    d[i] = b[i] + c[i]
  e = 0
  for i in range(10):
    e += d[i]
  return e

print(a([1,2,3,4,5,6,7,8,9,10], [10,9,8,7,6,5,4,3,2,1]))
```

https://towardsdatascience.com/managing-project-specific-environments-with-conda-b8b50aa8be0e

Warm-up

- What can be improved? - Solution
  - Function and variables names
  - Indentation
  - Comments + Docstring
  - Error handling
  - Usage of ‘zip’ to make the code easier understandable

```python
def calculate_sum_of_lists(list1, list2):
    """
    Calculate the sum of corresponding elements from two lists.

    :param list1: The first list
    :type list1: list of int
    :param list2: The second list
    :type list2: list of int
    :return: The sum of corresponding elements from the two lists
    :rtype: int
    """
    if len(list1) != len(list2):
        raise ValueError("Input lists must have the same length")

    result = 0
    for element1, element2 in zip(list1, list2):
        result += element1 + element2

    return result

list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list2 = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

total_sum = calculate_sum_of_lists(list1, list2)
print(f"The total sum is: {total_sum}")
```

1. Error handling

Errors

- Can occur in a Python program when something unexpected or incorrect happens
- Also known as exceptions
- Python has a built-in system for handling errors ➔ allows programs to gracefully handle unexpected situations
- Common types of errors:
    - SyntaxError: Occurs when the code violates Python's syntax rules
    - NameError: Occurs when a variable or function is referenced before it's defined
    - TypeError: Occurs when an operation is performed on an object of inappropriate type
    - ValueError: Occurs when a function receives an argument of correct type but with an inappropriate value

Error handling

- Exception handling with “try” and “except” blocks
- Code in the “try” block is executed, and if an exception occurs, the code in the “except” block is executed

```
try:
    # Code that may raise an exception
except SpecificExceptionType:
    # Code to handle the SpecificExceptionType
```

Error handling

- Optional: use “else” and “finally” blocks
- “else”: executed if no exception occurs in the “try” block
- “finally”: executed regardless of whether an exception occurs or not (often used for cleanup operations)

```
def divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Error: Division by zero")
    else:
        print(f"Result of the division: {result}")
    finally:
        print("Division operation completed (with or without errors).")

# Example 1: Division by a non-zero number
divide(10, 2)

# Example 2: Division by zero
divide(5, 0)
```

Error handling

- “raise”: possibility to “raise” custom exceptions ➜ allows to define and handle application-specific errors

```
class AuthenticationError(Exception):
    """Custom exception for authentication errors."""
    pass

def login(username, password):
    # Check if the username and password are valid
    if username == "admin" and password == "secret":
        print("Login successful. Welcome, admin!")
    else:
        raise AuthenticationError("Login failed. Invalid credentials.")

try:
    username = input("Enter your username: ")
    password = input("Enter your password: ")
    login(username, password)
except AuthenticationError as ae:
    print(f"Authentication Error: {ae}")
```

Best practices

- Exceptions hierarchy: handle exceptions at different levels of specificity. More specific exceptions should be caught before more general ones
- Be specific in exception handling to avoid catching too many exceptions
- Provide informative error messages when raising custom exceptions
- Avoid bare except blocks (e.g., except:) as they can hide errors

Error handling: Example

```
file_path = "data.txt"

try:
    with open(file_path, "r") as file:
        data = file.read()
except FileNotFoundError:
    print(f"Error: The file '{file_path}' was not found.")
except PermissionError:
    print(f"Error: Permission denied to access '{file_path}'.")
except IOError as e:
    print(f"An I/O error occurred: {e}")
else:
    # Process the data and perform calculations
    lines = data.split("\n")
    total = sum(int(line) for line in lines if line.strip().isdigit())
    print(f"Total sum of numbers in the file: {total}")
finally:
    # Cleanup or additional operations, if needed
    print("File reading and processing completed.")

# Continue with the rest of the program
```

2. Unit tests

Unit tests

- Software testing technique where individual components or functions (units) of a program are tested in isolation to ensure they work as expected
- Benefits:
  - Early bug detection
  - Improved code quality (leads to better code design and organization, making the codebase more maintainable and less error-prone)
  - Documentation (insights into how a particular function or class is intended to be used and what its expected behavior is)
  - Regression prevention (running existing unit tests can help detect regressions (unintended side effects) caused by changes) ➜ can be integrated into CI (Continuous Integration) pipelines
  - Supports Test-Driven Development (TDD)

Python testing frameworks

- Software tools or libraries that provide a structured way to write, organize, and run tests
- Popular Python testing frameworks:
  - '**unittest**': built-in framework in Python's standard library
  - **'pytest'**: a third-party framework with many features and plugins
  - **'nose'**: a test discovery and test running framework
- Key features:
  - **Test discovery**: Framework can automatically discover and collect test cases and test methods, e.g. 'unittest' identifies test cases where the method starts with "test_"
  - **Test execution**: Framework provide a test runner that executes the tests, reporting the results of each test case (pass/fail) and any exceptions raised during execution
  - **Fixture management**: 'setUp' and 'tearDown'

Fixture management: Example

```
import unittest

class TestMathOperations(unittest.TestCase):
    def setUp(self):
        # Fixture setup code: Initialize common resources
        self.x = 5
        self.y = 3

    def tearDown(self):
        # Fixture teardown code: Clean up resources
        pass

    def test_addition(self):
        result = self.x + self.y
        self.assertEqual(result, 8)

    def test_subtraction(self):
        result = self.x - self.y
        self.assertEqual(result, 2)

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

Test cases: Example

```python
# math_operations.py
def add(a, b):
    return a + b
```

```python
import unittest
from math_operations import add

class TestMathOperations(unittest.TestCase):

    def test_add_positive_numbers(self):
        result = add(3, 5)
        self.assertEqual(result, 8)

    def test_add_negative_numbers(self):
        result = add(-2, -4)
        self.assertEqual(result, -6)

    def test_add_mixed_numbers(self):
        result = add(10, -7)
        self.assertEqual(result, 3)

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

3. Version control (Git)

Version control

- Why version control

Thesis.docx
Thesis_1.docx
Thesis_1_anki.docx
Thesis_2.docx
Thesis_2_anki.docx
Thesis_2_AB.docx
.....
Thesis_final.docx
Thesis_final_anki.docx
Thesis_final2.docx
Thesis_final2_fix.docx

https://imgflip.com/i/7217ou

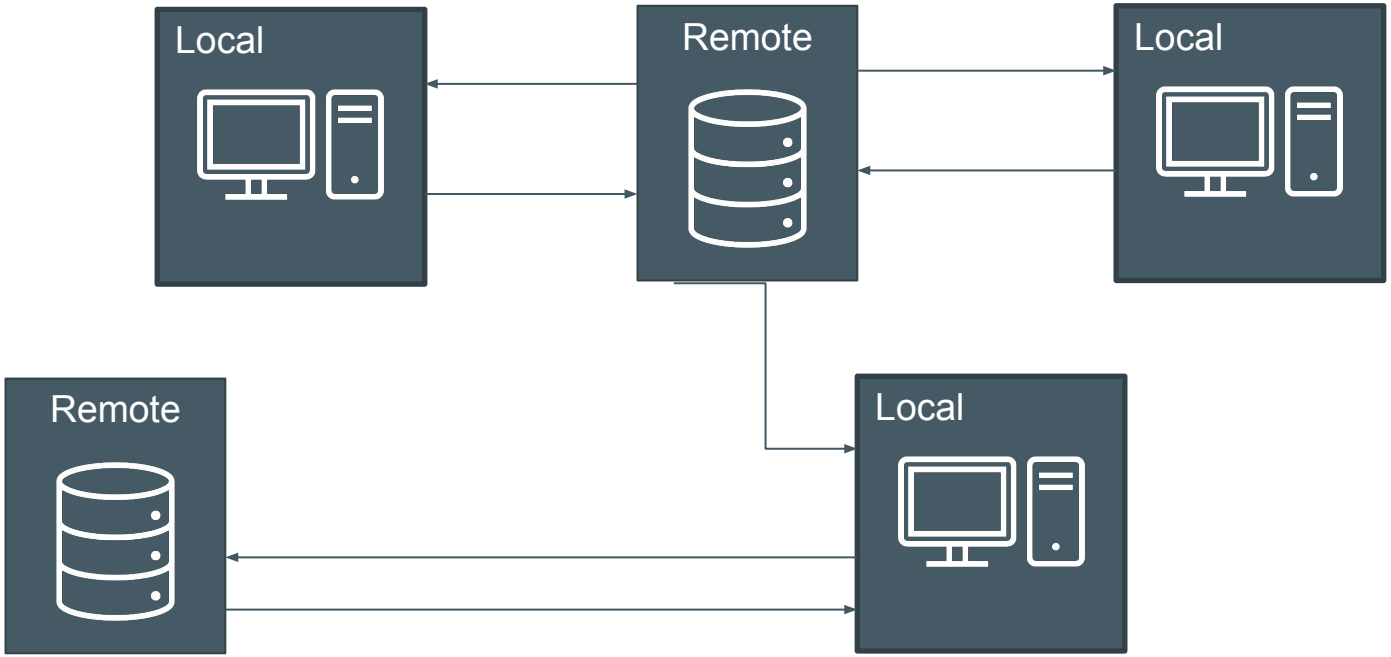
Version control

- Different version control systems
  - SVN
  - Apache Subversion
  - Mercurial
  - Bazaar
  - GNU arch
  - Git

https://blog.developer.atlassian.com/force-with-lease/

Version control

- Distributed version control system



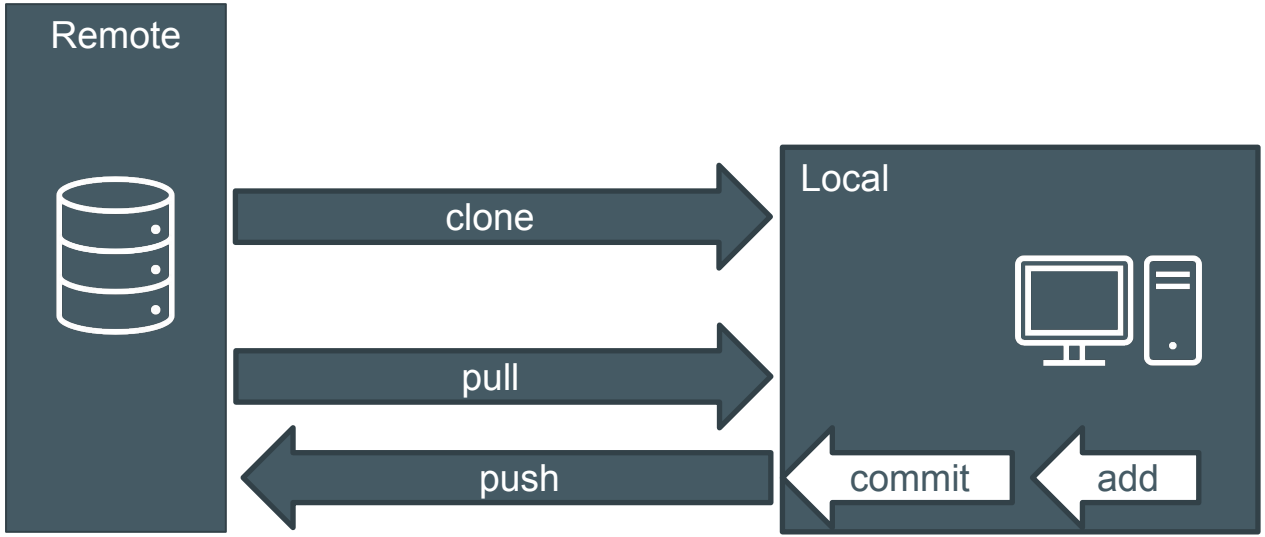
Version control

- How to start using version control in your project

1. Initialize remote repository: git init --bare
2. Clone to your working directory: git clone <path>
3. Commit changes: git add + git commit
4. Push to remote: git push

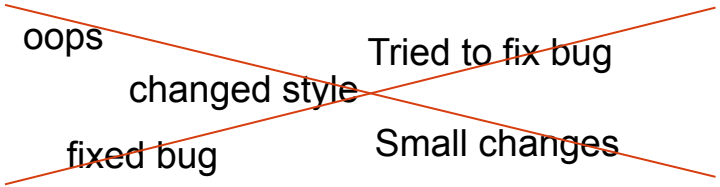
- Bare repository can be stored local or remote
  - Remote is better for coworking
  - Remote is better for securing against losing data

Version control



Version control

- Git commit requires a commit message




```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

  - type: feat, fix, refactor, docs, style, test, perf
  - description: a short description of what you did

- To work with
  - Terminal
  - Git GUI – TortoiseGit (https://tortoisegit.org/)

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~
$ mkdir exercise2/local

luisa@PC MINGW64 ~
$ cd exercise2/local/

luisa@PC MINGW64 ~/exercise2/local
$ git clone  ~/exercise2/remote/myproject
Cloning into 'myproject'...
warning: You appear to have cloned an empty repository.
done.

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ echo "# My Project" > README.md

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git status
On branch master

No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add"
to track)

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git add README.md

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git commit -m "initialized myproject with a readme"
[master (root-commit) f31161b] initialized myproject with README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git log
commit 522e5edc54c8379ce8e71fa421955dbcf3ceb0e4 (HEAD -> master)
Author: luisa <luisa@users.noreply.github.com>
Date:   Thu Nov 24 21:36:56 2022 +0100

    initialized myproject with a readme

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git push --set-upstream origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 259 bytes | 86.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To C:/Users/luisa/exercise2/remote/myproject
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~
$ mkdir exercise2/remote/myproject

luisa@PC MINGW64 ~
$ cd exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject
$ git init --bare
Initialized empty Git repository in
C:/Users/luisa/exercise2/remote/myproject/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git log
commit 522e5edc54c8379ce8e71fa421955dbcf3ceb0e4 (HEAD -> master,
origin/master)
Author: luisa <luisa@users.noreply.github.com>
Date:   Thu Nov 24 21:36:56 2022 +0100

    initialized myproject with a readme

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$ git log
commit 522e5edc54c8379ce8e71fa421955dbcf3ceb0e4 (HEAD -> master)
Author: luisa <luisa@users.noreply.github.com>
Date:   Thu Nov 24 21:36:56 2022 +0100

    initialized myproject with a readme

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

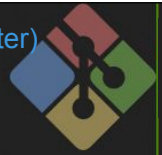
```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git log
commit 522e5edc54c8379ce8e71fa421955dbcf3ceb0e4 (HEAD -> master,
origin/master)
Author: luisa <luisa@users.noreply.github.com>
Date:   Thu Nov 24 21:36:56 2022 +0100

    initialized myproject with a readme

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

How to repository

```
luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$ git log
commit 522e5edc54c8379ce8e71fa421955dbcf3ceb0e4 (HEAD -> master)
Author: luisa <luisa@users.noreply.github.com>
Date:   Thu Nov 24 21:36:56 2022 +0100

    initialized myproject with a readme

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ git pull
Already up to date.

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

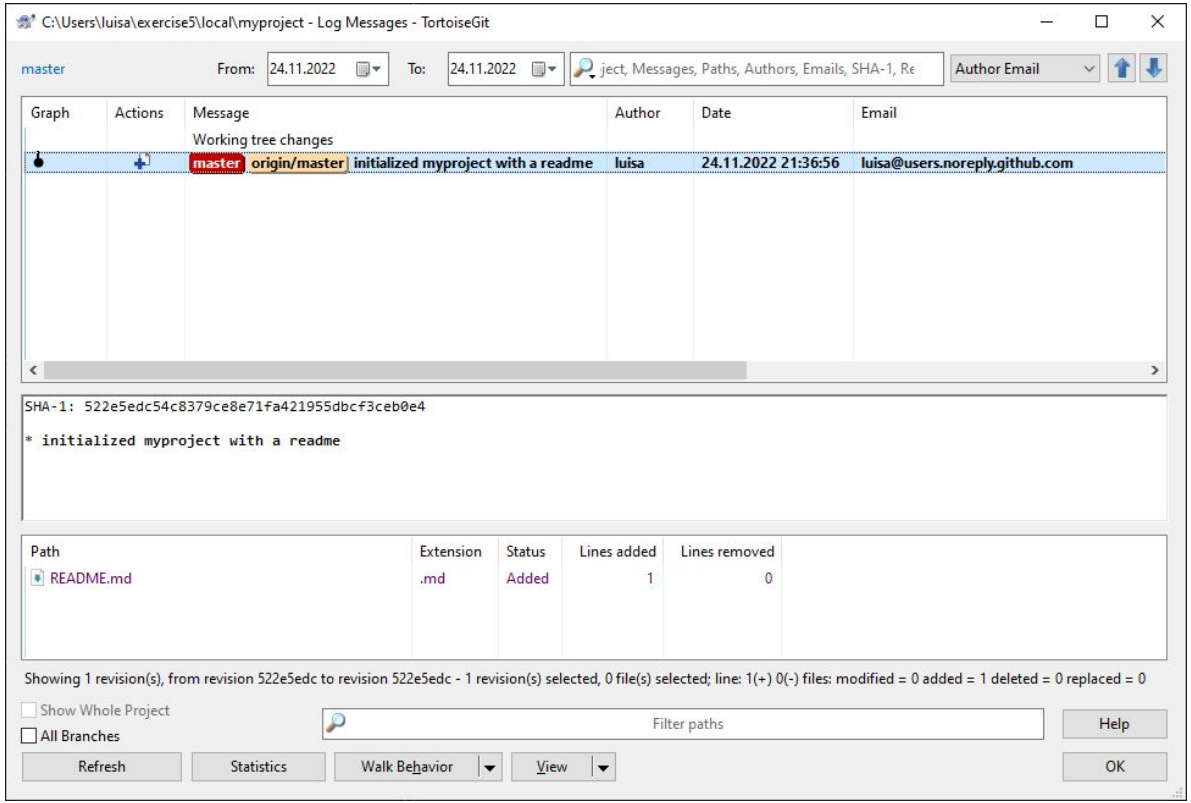
How to repository



```
luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$ ls -l
total 7
-rw-r--r-- 1 luisa 197611 104 Nov 24 21:02 config
-rw-r--r-- 1 luisa 197611  73 Nov 24 21:02 description
-rw-r--r-- 1 luisa 197611  23 Nov 24 21:02 HEAD
drwxr-xr-x 1 luisa 197611   0 Nov 24 21:02 hooks/
drwxr-xr-x 1 luisa 197611   0 Nov 24 21:02 info/
drwxr-xr-x 1 luisa 197611   0 Nov 24 21:39 objects/
drwxr-xr-x 1 luisa 197611   0 Nov 24 21:02 refs/

luisa@PC MINGW64 ~/exercise2/remote/myproject (BARE:master)
$
```

```
luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$ ls -l
total 1
-rw-r--r-- 1 luisa 197611 14 Nov 24 21:33 README.md

luisa@PC MINGW64 ~/exercise2/local/myproject (master)
$
```

TortoiseGit



How to www repository

- Different hosts/software solutions for remote
  - GitHub (https://github.com/)
  - GitLab (https://about.gitlab.com/de-de/)
  - Gitea (https://gitea.io/en-us/)
  - Gitpod (https://www.gitpod.io/media-kit)

- Generate bare repository on GitHub

https://memegenerator.net/instance/67290310/afraid-to-ask-andy-i-dont-know-how-github-works-and-at-this-point-im-too-afraid-to-ask

Summary

- Bare repository: git init --bare or on GitHub via the web interface
- Getting the repository for the first time: git clone
- Staging files: git add
- Committing files: git commit
- Updating remote with local commits: git push
  - (if newly created branch:--set-upstream <remote-name> <branch-name>)
- Getting changes from the remote: git pull

Good to know

- Branching model
- 'master' branch protection

- Pull requests
- Merging

https://imgflip.com/tag/git?sort=latest&after=5f07jj
https://tenor.com/de/view/git-merge-git-merge-gif-20623312

4. Python packaging

How to python package

- Script not importable outside of the current working directory
- Package script as a module to make it installable and importable
- Different ways to package we do it via setup.py

How to python package

Basic directory structure:

- .gitignore
- LICENSE
- MANIFEST.in
- README.md
- requirements.txt
- setup.py
- **<package_name>/**
    - __init__.py
    - __main__.py
    - function.py

How to python package

```
luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ ls -l
-rw-r--r-- 1 luisa 197611  1928 Nov 23 23:08 .gitignore
-rw-r--r-- 1 luisa 197611 11558 Nov 23 23:08 LICENSE
-rw-r--r-- 1 luisa 197611    35 Nov 23 23:39 MANIFEST.in
-rw-r--r-- 1 luisa 197611    11 Nov 23 23:08 README.md
-rw-r--r-- 1 luisa 197611    15 Nov 23 23:08 requirements.txt
-rw-r--r-- 1 luisa 197611   266 Nov 23 23:08 setup.py
drwxr-xr-x 1 luisa 197611     0 Nov 23 23:08 snowflake/

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$
```

How to python package

```
luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ ls -l
-rw-r--r-- 1 luisa 197611  1928 Nov 23 23:08 .gitignore
-rw-r--r-- 1 luisa 197611 11558 Nov 23 23:08 LICENSE
-rw-r--r-- 1 luisa 197611    35 Nov 23 23:39 MANIFEST.in
-rw-r--r-- 1 luisa 197611    11 Nov 23 23:08 README.md
-rw-r--r-- 1 luisa 197611    15 Nov 23 23:08 requirements.txt
-rw-r--r-- 1 luisa 197611   266 Nov 23 23:08 setup.py
drwxr-xr-x 1 luisa 197611     0 Nov 23 23:08 snowflake/

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ ls -l snowflake/
-rw-r--r-- 1 luisa 197611   28 Nov 23 23:08 __init__.py
-rw-r--r-- 1 luisa 197611   49 Nov 23 23:08 __main__.py
-rw-r--r-- 1 luisa 197611 1488 Nov 23 23:08 let_it_snow.py

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$
```

How to python package

```
luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ python -m pip install .
Processing c:\users\luisa\exercise2\exercise2
  Preparing metadata (setup.py) ... done
Collecting numpy
  Downloading numpy-1.23.5-cp310-cp310-win_amd64.whl (14.6 MB)
     ---------------------------------------- 14.6/14.6 MB 3.1 MB/s eta 0:00:00
Collecting turtles
  Using cached turtles-1.0.0-py3-none-any.whl (2.8 kB)
Building wheels for collected packages: snowflake
  Building wheel for snowflake (setup.py) ... done
  Created wheel for snowflake: filename=snowflake-0.1-py3-none-any.whl size=6304
sha256=2f08bb68aa991372fa02ab5ff15da5810c565b52a69d8c7fe019f819cf723ead
  Stored in directory:
C:\Users\luisa\AppData\Local\Temp\pip-ephem-wheel-cache-1s3o9d6o\wheels\4a\84\8b\e8841e7caa0ab4bcecc62729004c686596d13f222ac46
e25ec
Successfully built snowflake
Installing collected packages: turtles, numpy, snowflake
Successfully installed numpy-1.23.5 snowflake-0.1 turtles-1.0.0

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$
```

How to python package

```
luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ python -m pip install git+https://github.com/luisa/exercise2.git
Collecting git+https://github.com/luisa/exercise2.git
  Cloning https://github.com/luisa/exercise2.git to c:\users\luisa\appdata\local\temp\pip-req-build-prhoio6t
  Running command git clone --filter=blob:none --quiet https://github.com/luisa/exercise2.git
'C:\Users\luisa\AppData\Local\Temp\pip-req-build-prhoio6t'
  Resolved https://github.com/luisa/exercise2.git to commit 93d9a5532fdeb52bfc54829b6f032fddeff0d82e
  Preparing metadata (setup.py) ... done
Collecting numpy
  Using cached numpy-1.23.5-cp310-cp310-win_amd64.whl (14.6 MB)
Collecting turtles
  Using cached turtles-1.0.0-py3-none-any.whl (2.8 kB)
Building wheels for collected packages: snowflake
  Building wheel for snowflake (setup.py) ... done
  Created wheel for snowflake: filename=snowflake-0.1-py3-none-any.whl size=6304
sha256=104350c84f9dff04af60fc0adac558bfbb3befa22cb6ee9b13d9c7c6fa28809c
  Stored in directory:
C:\Users\luisa\AppData\Local\Temp\pip-ephem-wheel-cache-l7kepkqs\wheels\b9\60\02\c447301c6f2223ba6d106a1cf836e6a0e9ace3d3321b24
e3ef
Successfully built snowflake
Installing collected packages: turtles, numpy, snowflake
Successfully installed numpy-1.23.5 snowflake-0.1 turtles-1.0.0

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$
```

How to python package

```
luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ python -c "import snowflake"

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$ python -m snowflake

luisa@PC MINGW64 ~/exercise2/exercise2 (main)
$
```

Summary

- Making script usable outside of current working directory by creating a python package
- Structure of directory
  - README.md, LICENSE, MANIFEST.in, .gitignore
  - setup.py
  - <package_name>
- Other strategy as defined in PEP 518 and PEP 621: pyproject.toml