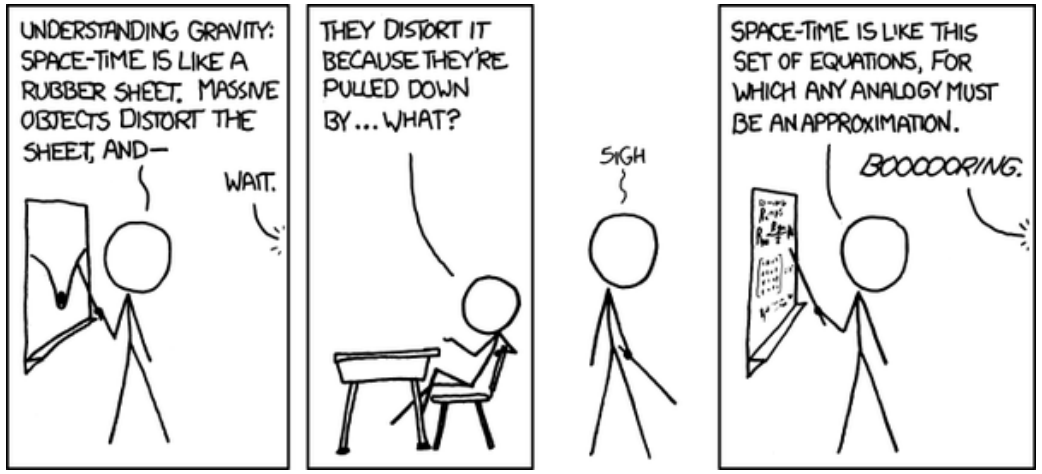
"Advanced" Python and Containers

Stephan Meighen-Berger

Disclaimers!

- Experts will look at this and cry
  - (You might as well)
- These are a collection of tips and tricks
  - Pick and choose what's useful to you
- The standard used here is one of many!
  - Just be consistent!



https://xkcd.com/895/

Before you get bored…

Document your code!!!!!!!!!!!!!!

Some Style Guides

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

- Just your style, e.g. PEP8
  - Indentation
  - Line length
  - Operator Line Breaks
  - Operator Spacing
  - Blank Lines
  - Imports
  - White Spaces
  - Comments
  - Documentation
  - …

My recommendation is to use these standards to make your code readable

```
# Correct:
import os
import sys
```

```
# Wrong:
import sys, os
```

```
# Correct:
spam(ham[1], {eggs: 2})
```

```
# Wrong:
spam( ham[ 1 ], { eggs: 2 } )
```

```
# Correct:
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

```
# Wrong:
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

Documentation

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html

- Numpy style doc
  - This example could be considered **MINIMAL!**

```
1  def q_statistic_scalar(signal, background, data, quant=0.1):
2      q_arr = -2. * np.log(
3          poisson.pmf(data, signal+background)
4      )
5      q_quant = np.quantile(q_arr, quant)
6      return q_arr, q_quant
```

This is REALLY bad. No documentation at all

Some **MINIMAL** documenation

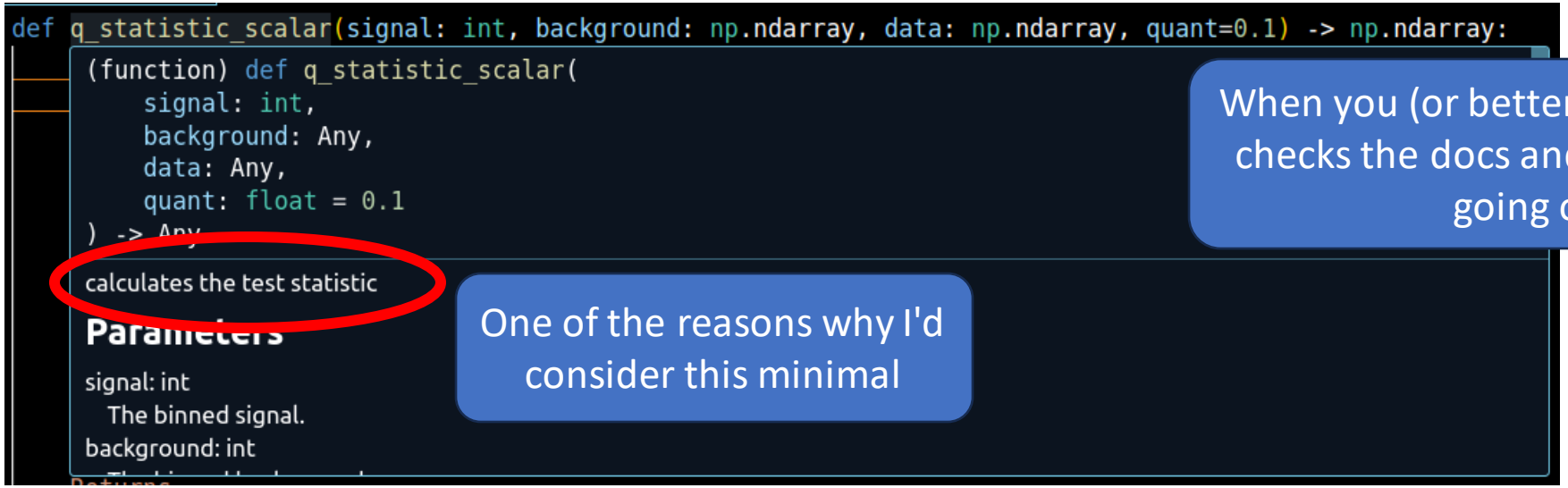
```python
def q_statistic_scalar(signal: int, background: np.ndarray, data: np.ndarray, quant=0.1) -> np.ndarray:
    """ calculates the test statistic
    Parameters
    ----------
    signal: int
        The binned signal.
    background: int
        The binned background
    data: np.array
        The binned data (1D)
    quant: float
     Optional quantile parameter

    Returns
    -------
    q: np.ndarray
        The resulting test statistic

    Errros
    ------
    AssertionError:
        Data type of signal (need to be int)
        Data type of background and data (need to be np.array)
        Data type of quant (needs to be a float)
    AssertionError:
        For the shapes of the input arrays (if they don't align)
    """
    # ---------------------------------------------------------------------------
    # Error handling
    assert isinstance(signal, int), \
        "signal needs to be an int!"
    assert isinstance(background, np.ndarray), \
        "background needs to be a np.array!"
    assert isinstance(data, np.ndarray), \
        "data needs to be a numpy.array!"
    assert isinstance(quant, float), \
        "quant needs to be a float!"
    assert background.shape == data.shape, \
        'Background and data shapes need to be the same! Got b: ' + str(background.shape) + ' and d: ' + str(data.shape)
    # ---------------------------------------------------------------------------
    # Calculation
    q_arr = -2. * np.log(
        poisson.pmf(data, signal+background)
    )
    q_quant = np.quantile(q_arr, quant)
    return q_arr, q_quant
```

Documentation

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html

- Numpy style doc
  - This example could be considered **MINIMAL!**

- What's a good benchmark?



When you (or better someone else) checks the docs and knows whats going on

One of the reasons why I'd consider this minimal

Real Life Example – The Good

- Neutrino Oscillation Code: nuCraft

https://nucraft.hepforge.org/

An Example Method

Has the basics right

```
def ConstructMassMatrix(self, parList):
    """
    Construct and return a squared-mass matrix out of the input list;
    the first parameter is the mass of mass state 1, the following parameters are
    the correctly ordered squared mass differences of the other states to state 1,
    all given in units of eV or eV^2, respectively, i.e.,
    parList[i] = m_i^2 - m_1^2   for i > 0; e.g.,
    parList = [1., 7.50e-5, 7.50e-5+2.32e-3]
    """
    # ensure that all masses are positive
    assert -parList[0]**2 <= min(parList[1:]), "All masses have to be positive!"

    return diag([parList[0]**2] + [parList[0]**2 + m for m in parList[1:]])
```

Good amount of documentation

Some error handling

If your code looks like this you're golden... Except...

Real Life Example – The Good II

- Neutrino Oscillation Code: nuCraft

https://nucraft.hepforge.org/

```
# number of energy bins
eBins = 1000
# zenith angles for the four plots
zList = arccos([-1., 0, 1])
# energy range in GeV
eList = logspace(-1, 1, eBins)



# parameters from arxiv 1205.7071
theta23 = arcsin(sqrt(0.420))/pi*180.
theta13 = arcsin(sqrt(0.025))/pi*180.
theta12 = arcsin(sqrt(0.312))/pi*180.
DM21    = 7.60e-5
DM31    = 2.35e-3 + DM21

# Akhemdov is implicitely assuming an electron-to-neutron ratio of 0.5;
# he is also using the approximation DM31 = DM32;
# if you want to reproduce his numbers exactly, switch the lines below, and turn
# atmosphereMode to 0 (no handling of the atmosphere because of )
# AkhmedovOsci = NuCraft((1., DM21, DM31-DM21), [(1,2,theta12),(1,3,theta13,0),(2,3,theta23)],
#                        earthModel=EarthModel("prem", y=(0.5,0.5,0.5)),
#                        detectorDepth=0., atmHeight=0.)
AkhmedovOsci = NuCraft((1., DM21, DM31-DM21), [(1,2,theta12),(1,3,theta13,0),(2,3,theta23)],
                       earthModel=EarthModel("prem"),
                       detectorDepth=0.5, atmHeight=0.)
# AkhmedovOsci = NuCraft((1., DM21, DM31), [(1,2,theta12),(1,3,theta13,0),(2,3,theta23)])

# To compute weights with a non-zero CP-violating phase, replace     ^  this zero
# by the corresponding angle (in degrees); this will add the phase to the theta13 mixing matrix,
# as it is done in the standard parametrization; alternatively, you can also add CP-violating
# phases to the other matrices, but in the 3-flavor case more than one phase are redundant.

atmosphereMode = 3   # use fixed atmsopheric depth (set to 0 km if line 42 is not commented out)
# atmosphereMode = 3   # default: efficiently calculate eight path lenghts per neutrino and take the average

# This parameter governs the precision with which nuCraft computes the weights; it is the upper
# limit for the deviation of the sum of the resulting probabilities from unitarity.
# You can verify this by checking the output plot example-standAlone2.png.
numPrec = 5e-4



# 12, -12:  NuE, NuEBar
# 14, -14:  NuMu, NuMuBar
# 16, -16:  NuTau, NuTauBar
pType = 14
```

Even comes with an example how to use the code

All of this is to solve one simple equation!

$$\frac{\mathrm{d}}{\mathrm{d}x}\left|\nu_\alpha\right\rangle = \frac{-i}{2E_\nu}\left(H_0 + A\right)\left|\nu_\alpha\right\rangle$$

Installation is easy, since it's only a file

This code is an example of something that is widely used in the community!

So what's bad about it?

Real Life Example – The Bad

- Neutrino Oscillation Code: nuCraft

https://nucraft.hepforge.org/

Installation is easy, since it's only a file

It uses two classes:
- EarthModel
- NuCraft

Why is this a class? The code only generates one instance

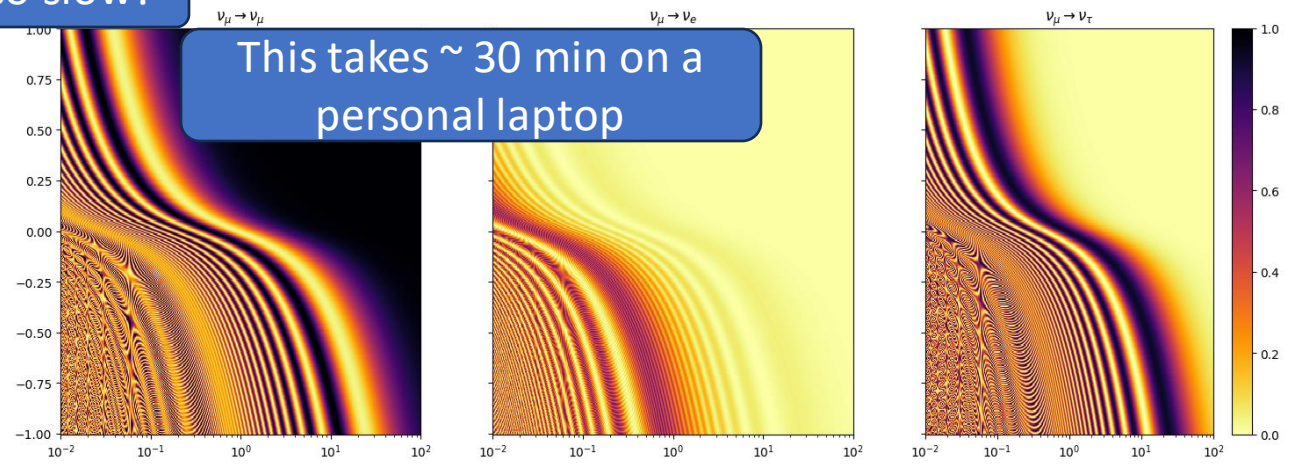
Why is this method visible to the user?

```
def ConstructMassMatrix(self, parList)
```

Why is it so slow?

Why is it only a file?

This takes ~ 30 min on a personal laptop

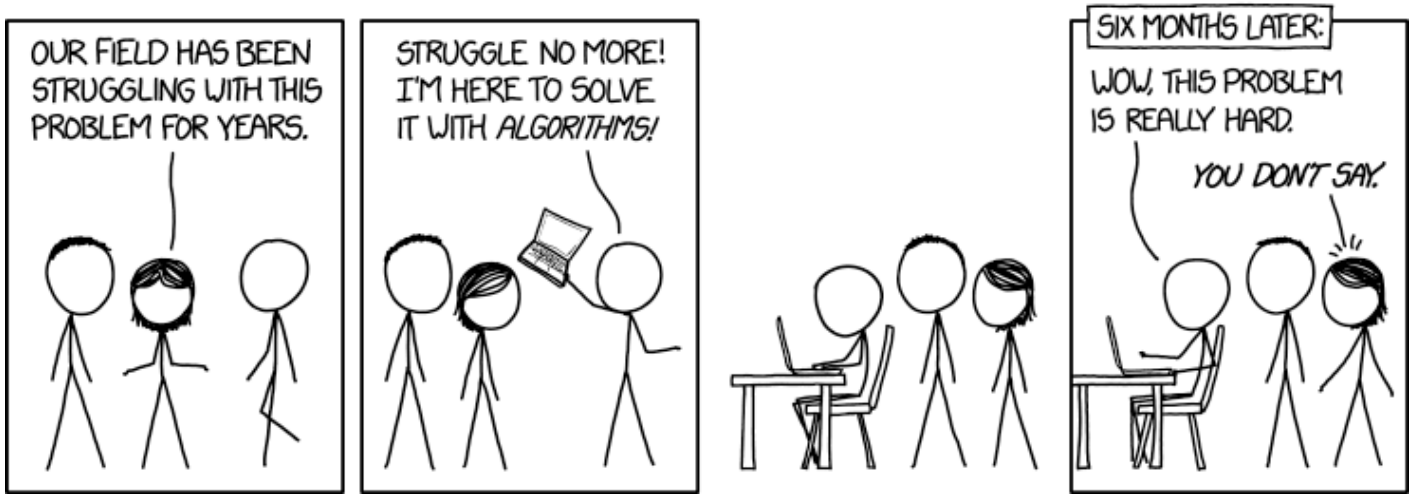


https://nucraft.hepforge.org/

Why host it here?

Real Life Example – Let's improve!

- What do we want?
  - Quick neutrino oscillation code
  - Easy to install and use
  - Well-documented
- Some bonus features
  - Logging
  - Modular
  - Easily maintainable



https://xkcd.com/1831/

Real Life Example – Modular

- Every folder has an __init__.py

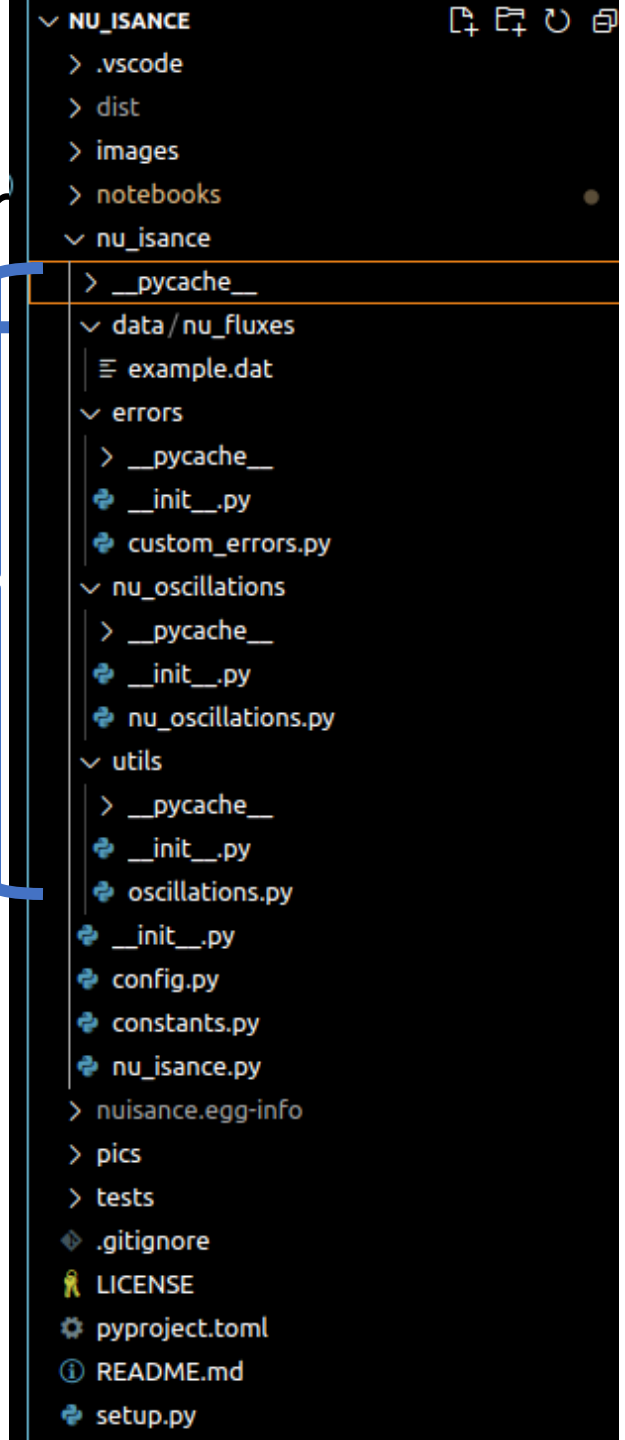
```
1  from .oscillations import oscillation_calc_func
2  from .oscillations import oscillation_calc_func_effective, effective_matrices
```

Essentially use them to import stuff

```
3  from .nu_isance import Nuisance
4  from .config import config
5  from .errors import __init__
6  from .nu_oscillations import __init__
7  from .utils import __init__
8
9  __all__ = (Nuisance, config)
10
11 # Version of the nuisance package
12 __version__ = "1.0.2"
13 __author__ = "Stephan Meighen-Berger"
```

Main __init__ puts it all together

Code

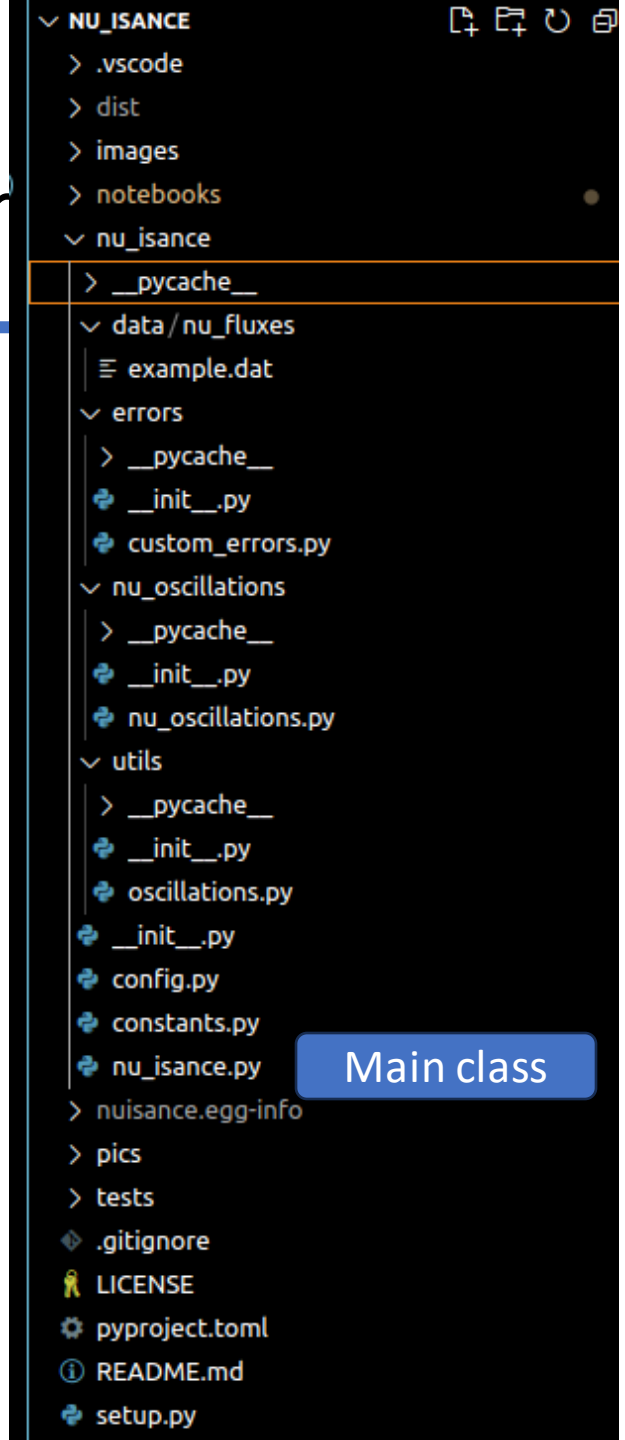


Real Life Example – Modular

- Your main class shouldn't do anything fancy!
  - Use it to put stuff together

```python
# unless we put this class in __init__, __name__ will be nuisance.nuisance
_log = logging.getLogger("")

# TODO: Make this an option in the config!
# Suppressing the numba logger
numba_logger = logging.getLogger('numba')
numba_logger.setLevel(logging.WARNING)

matplotlib_logger = logging.getLogger('matplotlib')
matplotlib_logger.setLevel(logging.WARNING)


class Nuisance(object):
    """ the Nuisance class. This object is the interface to the
    nuisance package
    """
    def __init__(
        self,
        userconfig: Union[None, dict, str]=None
    ) -> None:
        """Initializes the nuisance class
        params
        ------
        userconfig: Configuration dictionary or
```

A logger is good practice

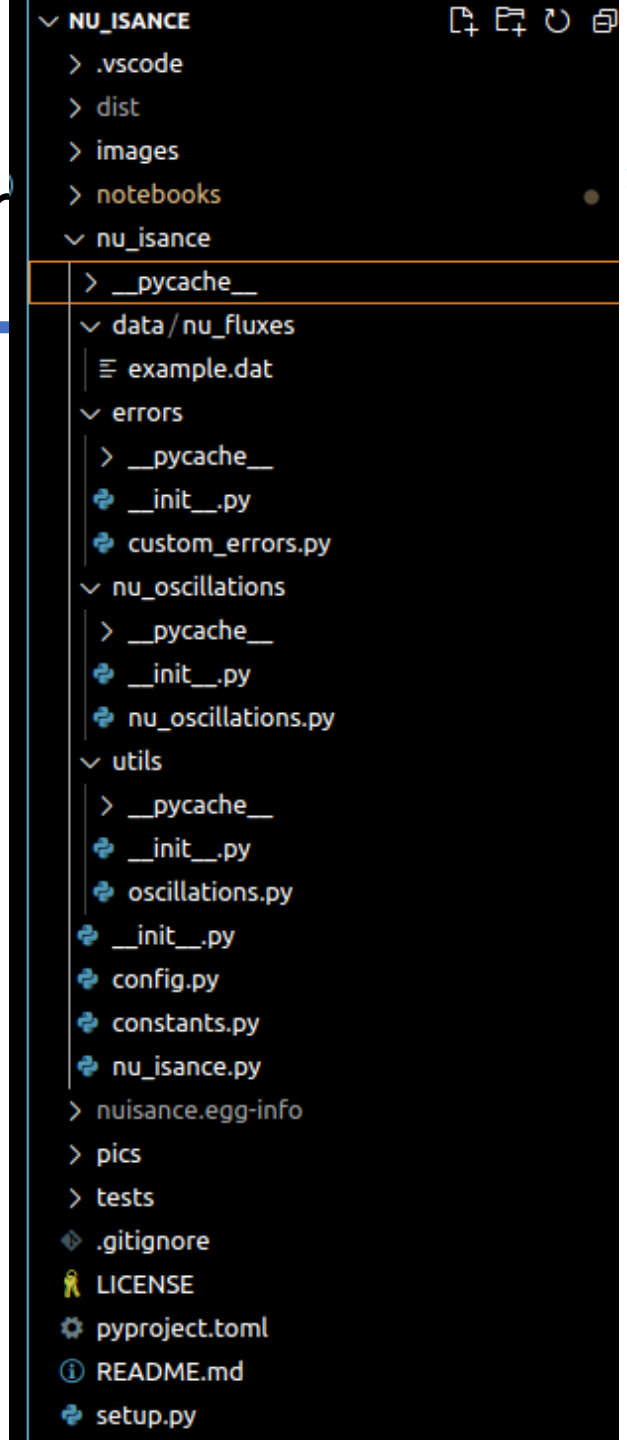


Main class

Real Life Example – Modular

- Your logger will help you debug your code (or someone elses)

```
[root] INFO: Starting
[root] INFO: Welcome to nuisance. I'm here to help
[nu_isance.nu_oscillations.nu_oscillations] INFO: Propagating through matter
[nu_isance.nu_oscillations.nu_oscillations] INFO: Building the oscillation grids
[nu_isance.nu_oscillations.nu_oscillations] INFO: Using 1 as the anti setting
[nu_isance.nu_oscillations.nu_oscillations] INFO: For nu_e...
[nu_isance.nu_oscillations.nu_oscillations] INFO: Done!
[nu_isance.nu_oscillations.nu_oscillations] INFO: For nu_mu...
[nu_isance.nu_oscillations.nu_oscillations] INFO: Done!
[nu_isance.nu_oscillations.nu_oscillations] INFO: For nu_tau...
[nu_isance.nu_oscillations.nu_oscillations] INFO: Done!
[root] INFO: Setup took -26 seconds
```



Real Life Example – Methods

- Some comments on writing classes and methods
  - Methods and variables you don't want your users to use:

```
def _oscillation_grid_constructor(
        self,
        e_grid: np.ndarray, cosZ: np.ndarray, mixing_angles: np.ndarray,
        matter: np.ndarray, anti: int
    )->np.ndarray:
```

  - Methods and variables you REALLY don't want your users to use:

```
def __oscillation_grid_constructor(
        self,
        e_grid: np.ndarray, cosZ: np.ndarr
        matter: np.ndarray, anti: int
    )->np.ndarray:
```

```
nuisance.osc.
# ----------
# Plotting
# ----------
# Setup
_, axs = plt.
colors = [
    "#1b9e77"
    "#d95f02"
    "#7570b3"
```

oscillation_prob_e
oscillation_prob_mu
oscillation_prob_tau
_oscillation_grid_constructor
_result_e
_result_mu
_result_tau
__annotations__
__class__
__delattr__
__dict__
__dir__

Want them to see

Please don't touch

GO AWAY!

Real Life Example – Config File

- Usually you can define a standard setup
- A general user will not change most standards
- Make their lives (and yours) easier by choosing a good standard set of variables for runs

```python
import logging
from typing import Dict, Any
import yaml
import numpy as np

_baseconfig: Dict[str, Any]

_baseconfig = {
    ##########################################################################
    # General inputs
    ##########################################################################
    "general": {
        # Random state seed
        "random state seed": 1337,
        # Enable logger and config dump
        "enable logging": True,
        # Output level
        'debug level': logging.INFO,
        # Note the paths need to be set appropiately for your system
        # Location of logging file handler
        "log file handler": "nuisance.log",
        # Dump experiment config to this location
        "config location": "nuisance.txt",
    },
    ##########################################################################
    # Oscillation setup
    ##########################################################################
    "oscillation": {
        # If to use pre-calculated grids
        "precalc": False,
        "energy grid": np.logspace(-2, 2, 1000),
        "angle grid": np.linspace(-1, 1., 400),
        "matter": True,
    },
}
```

SET A SEED!

Logging

Where to dump run information

Calculation Parameters

Real Life Example – An Example

- Create simple examples which show what your code can do

https://colab.research.google.com/github/mjg-phys/cdm-computing-subgroup/blob/main/advancedPythonTutorial/nuisance/notebooks/basic_example.ipynb

Real Life Example – How to publish?

- Let's use GitHub as an example
  - You should be using git anyways!

- What do we still need?
  - A readme.md
  - A License
  - setup.py or requirements.txt would be nice of you

The license you choose will also depend on your institution, collaborators, and grants!

Personally I tend to use a GPL License

README GPL-3.0 license

Nuisance

Authors:

1. Stephan Meighen-Berger, developed the Nuisance Code

Table of contents

Introduction

Welcome to Nuisance!

A python package to simulate neutrino oscillations.

Examples are given in the notebook folder, but here is a basic script to get you going:

```
# Module import
from nu_isance import Nuisance, config

# A nusciance instance with options
config['oscillation']['matter'] = True
nuisance = Nuisance()

# Accessing the simulation
nu_e_e = nuisance.osc.oscillation_prob_e[0]  # For nu_e -> nu_e
nu_e_mu = nuisance.osc.oscillation_prob_e[1]  # For nu_e -> nu_mu
nu_e_tau = nuisance.osc.oscillation_prob_e[2]  # For nu_e -> nu_tau
```

Citation

Please cite this software using

```
@software{nuisance@github,
  author = {Stephan Meighen-Berger},
  title = {{Nuisance}: Neutrino Oscillations},
  url = {https://github.com/MeighenBergerS/nu_isance},
  version = {1.0.2},
  year = {2024},
}
```

Installation

Since this only an example the wheels are hosted on TestPyPi. To install use:

Install using pip:

```
pip install -i https://test.pypi.org/simple/ --extra-index-url https://pypi.org/simple/ nuisan
```

The PyPi webpage

Real Life Example – Publish

- Now let's go a step further
  - Build your own python package today!
- What do we still need?
  - A setup.py
  - A pyproject.toml

```
[build-system]
requires = ['setuptools>=59']
build-backend = 'setuptools.build_meta'
```

```
import pathlib
from setuptools import setup

# Parent directory
HERE = pathlib.Path(__file__).parent

# The readme file
README = (HERE / "README.md").read_text()

setup(
    name="nuisance",
    version="1.0.2",
    description="Quick and dirty neutrino oscillations",
    long_description=README,
    long_description_content_type="text/markdown",
    author="Stephan Meighen-Berger",
    author_email="stephan.meighenberger@unimelb.edu.au",
    url='https://github.com/MeighenBergerS/nu_isance',
    license="GNU",
    install_requires=[
        "PyYAML",
        "numpy",
        "scipy",
        "pandas",
        "tqdm",
        "numba"
    ],
    extras_require={
        "interactive": ["nbstripout", "matplotlib", "jupyter"],
    },
    packages=[
        "nu_isance",
        "nu_isance.utils",
        "nu_isance.errors",
        "nu_isance.nu_oscillations"
    ],
    package_data={'nu_isance': ["data/*.pkl"]},
    include_package_data=True
)
```

Real Life Example – Publish

- Now let's go a step further
  - Build your own python package today!

- Push to PyPi and installation is easy

```
pip install nuisance
```

Since we are just testing some things out

```
pip install -i https://test.pypi.org/simple/ --extra-index-url https://pypi.org/simple/ nuisance==1.0.2
```

Or in Collab

```
%pip install -i https://test.pypi.org/simple/ --extra-index-url https://pypi.org/simple/ nuisance==1.0.2
```

But Wait!

- Some codes have complicated dependencies
  - Physics codes are always easy to install...
- Need a different distribution method
- -> Containers to the rescue!
- But what are they?

ChatGPT 3.5

Docker containers are lightweight, standalone, executable packages that contain everything needed to run a piece of software, including the code, runtime, libraries, and system tools. They are based on the concept of containerization, which isolates applications from their environments, ensuring consistency and reproducibility across different systems.

Here are some key characteristics of Docker containers:

1. **Isolation**: Containers encapsulate applications and their dependencies, ensuring they run consistently regardless of the environment in which they are deployed.
2. **Portability**: Docker containers can run on any platform that supports Docker, including local development environments, data centers, cloud providers, and even IoT devices.
3. **Efficiency**: Containers share the host operating system's kernel, which makes them lightweight and efficient compared to traditional virtual machines.
4. **Versioning and reproducibility**: Docker containers can be versioned and easily reproduced, enabling developers to create consistent development, testing, and production environments.
5. **Scalability**: Containers can be quickly deployed and scaled up or down to meet changing demand, making them well-suited for microservices architectures and cloud-native applications.

Docker containers are managed using the Docker Engine, which provides tools for building, running, and managing containers. Dockerfiles are used to define the configuration of a container, specifying the base image, dependencies, and runtime environment. Docker Compose is another tool used to define and manage multi-container Docker applications.

Overall, Docker containers have revolutionized software development and deployment by providing a consistent and efficient way to package, distribute, and run applications across different environments.

THE UNIVERSITY OF MELBOURNE

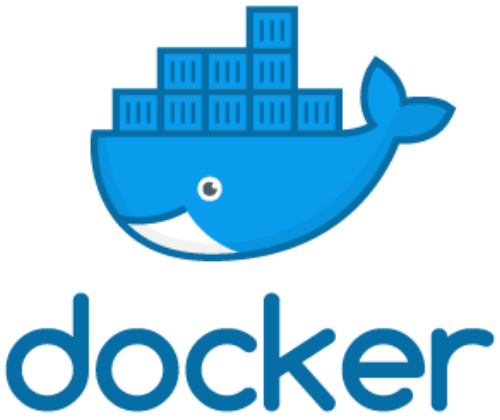
OHIO STATE

All you need to know:
OS level virtualization

Containers

Docker

- Root access
- Easier to use
- Personal use



A single container file

Both types can be generated using the same script

Singularity

- HPC usage

```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04 as base
# labels
LABEL author="Stephan Meighen-Berger"
LABEL version="1.0.2"
LABEL description="Image for Prometheus - New Kernel"
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update && \
    apt-get -y --no-install-recommends install \
    build-essential \
    gcc \
    g++ \
    git \
    libssl-dev \
    python3-dev \
    ca-certificates \
    wget libgsl-dev pkg-config libhdf5-serial-dev libboost-all-dev python-dev && \
    # apt-get install libboost-all-dev && \
    rm -rf /var/lib/apt/lists/*
RUN wget https://github.com/Kitware/CMake/releases/download/v3.22.2/cmake-3.22.2.tar.gz --no-check-certificate && \
    tar -zxvf cmake-3.22.2.tar.gz && \
    cd cmake-3.22.2 && ./bootstrap && make -j4 && make install
FROM base as python
ENV PATH="/opt/miniconda3/bin:${PATH}"
ARG PATH="/opt/miniconda3/bin:${PATH}"
RUN cd /opt &&\
    wget \
    https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh --no-check-certificate\
    && mkdir /opt/.conda \
    && bash Miniconda3-latest-Linux-x86_64.sh -b -p /opt/miniconda3\
    && rm -f Miniconda3-latest-Linux-x86_64.sh
ENV LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/miniconda3/lib
# Python > 3.10 breaks current pybinding of LI
# RUN conda install python=3.9
RUN conda install python=3.9
RUN conda install pip
RUN python -m pip install --upgrade pip
# change conan install .. -o with_python=True to conan install .. -o with_python=True -o boost:extra_b2_flags=define
# Problems with conan automtic versions
RUN python -m pip install proposal
FROM python as base_tables
```

The OS

Basic installations

Installing conda

A dockerfile snippet

Essentially it is a script to setup an entire system

Singularity – Some Examples

```
smeighenberg@6400l-211797-l:~/Projects/prometheus/container/old_kernel$ python
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
smeighenberg@6400l-211797-l:~/Projects/prometheus/container/old_kernel$ singularity shell prometheus_1_0_3_old.sif
Singularity> bash
smeighenberg@6400l-211797-l:~/Projects/prometheus/container/old_kernel$ python
Python 3.9.18 (main, Sep 11 2023, 13:41:44)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import proposal
>>> proposal.
proposal.Cartesian3D(                    proposal.PropagationUtility(              proposal.decay                            proposal.make_stochastic_deflection(
proposal.ContinuousRandomizer(           proposal.PropagationUtilityCollection(    proposal.density_distribution             proposal.make_time(
proposal.Decay(                          proposal.Propagator(                      proposal.geometry                         proposal.make_time_approximate(
proposal.Displacement(                   proposal.RandomGenerator(                 proposal.logging                          proposal.math
proposal.EnergyCutSettings(              proposal.Spherical3D(                     proposal.make_contrand(                   proposal.medium
proposal.Interaction(                    proposal.Time(                            proposal.make_decay(                      proposal.parametrization
proposal.InteractionLoss(                proposal.UnitSphericalVector(             proposal.make_default_stochastic_deflection(  proposal.particle
proposal.InteractionRate(                proposal.Vector3D(                        proposal.make_displacement(               proposal.scattering
proposal.InterpolationSettings(          proposal.component                        proposal.make_interaction(                proposal.secondaries
proposal.PropagationSettings(            proposal.crosssection                     proposal.make_multiple_sca
>>> proposal.
proposal.Cartesian3D(                    proposal.PropagationUtility(              proposal.decay                            _stochastic_deflection(
proposal.ContinuousRandomizer(           proposal.PropagationUtilityCollection(    proposal.density_distribut                _time(
proposal.Decay(                          proposal.Propagator(                      proposal.geometry                         _time_approximate(
proposal.Displacement(                   proposal.RandomGenerator(                 proposal.logging
proposal.EnergyCutSettings(              proposal.Spherical3D(                     proposal.make_contrand(                   proposal.medium
proposal.Interaction(                    proposal.Time(                            proposal.make_decay(                      proposal.parametrization
proposal.InteractionLoss(                proposal.UnitSphericalVector(             proposal.make_default_stochastic_deflection(  proposal.particle
proposal.InteractionRate(                proposal.Vector3D(                        proposal.make_displacement(               proposal.scattering
proposal.InterpolationSettings(          proposal.component                        proposal.make_interaction(                proposal.secondaries
proposal.PropagationSettings(            proposal.crosssection                     proposal.make_multiple_scattering(
>>> proposal.
```

No python

To have a nicer terminal

Shelling into the container. Yes, that's it!

Now we have python and packages!

Example of functions the developers want a user to see

Singularity – Some Examples

```
1   #!/bin/bash
2   #SBATCH --ntasks=1
3   #SBATCH --cpus-per-task=4
4   #SBATCH --time=0-20:00:00
5   #SBATCH --array=1-2
6   #SBATCH --mem=20G
7   #SBATCH --mail-user=stephan.meighenberger@unimelb.edu.au
8   #SBATCH --mail-type=ALL
9
10  # Load required modules
11  module purge
12  module load GCCcore/11.3.0
13  module load Apptainer/1.2.3
14
15  # Generate storage folder
16  mkdir /data/gpfs/projects/punim0011/smeighenberg/prometheus_output/job_${SLURM_ARRAY_TASK_ID}
17
18  # Shelling into the container, mounting directory and generating events
19  singularity exec --bind /tmp,/data/gpfs/projects/punim0011/smeighenberg:/mnt \
20  /data/gpfs/projects/punim0011/smeighenberg/containers/prometheus_old_1_0_3.sif \
21  bash /mnt/run_scripts/prometheus_setup.sh ${SLURM_ARRAY_TASK_ID}
22
23  ##Log this job's resource usage stats###
24  my-job-stats -a -j $JOBID > ${JOBID}.stats
25  ##
```

Slurm job submission

Load a module needed to run singularity

Many clusters now support singularity

The actual job
To get this running I only needed to dump a single file into my directory. This thing has MadGraph, Root, GENIE, Pythia etc.

One last message

Use Containers!