Class, PEP8, Sphinx

ÉCOLE **NORMALE**
SUPÉRIEURE
DE **LYON**

Ali Farnudi, Fall 2021

Object Oriented

- **fixed set of operations** on **things**

Functional

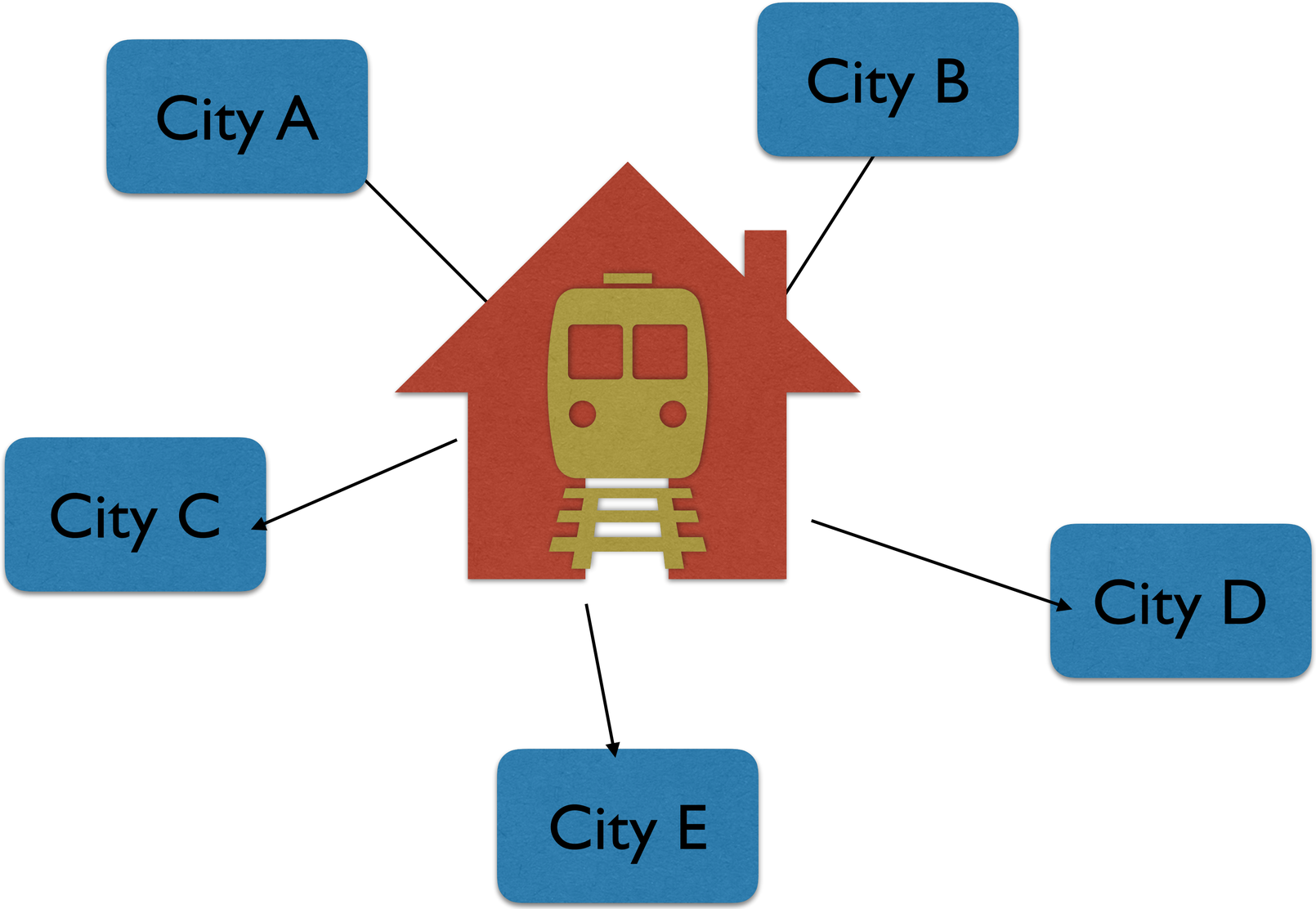
- **fixed set of things** and as your code evolves you **add new operations on those existing things**

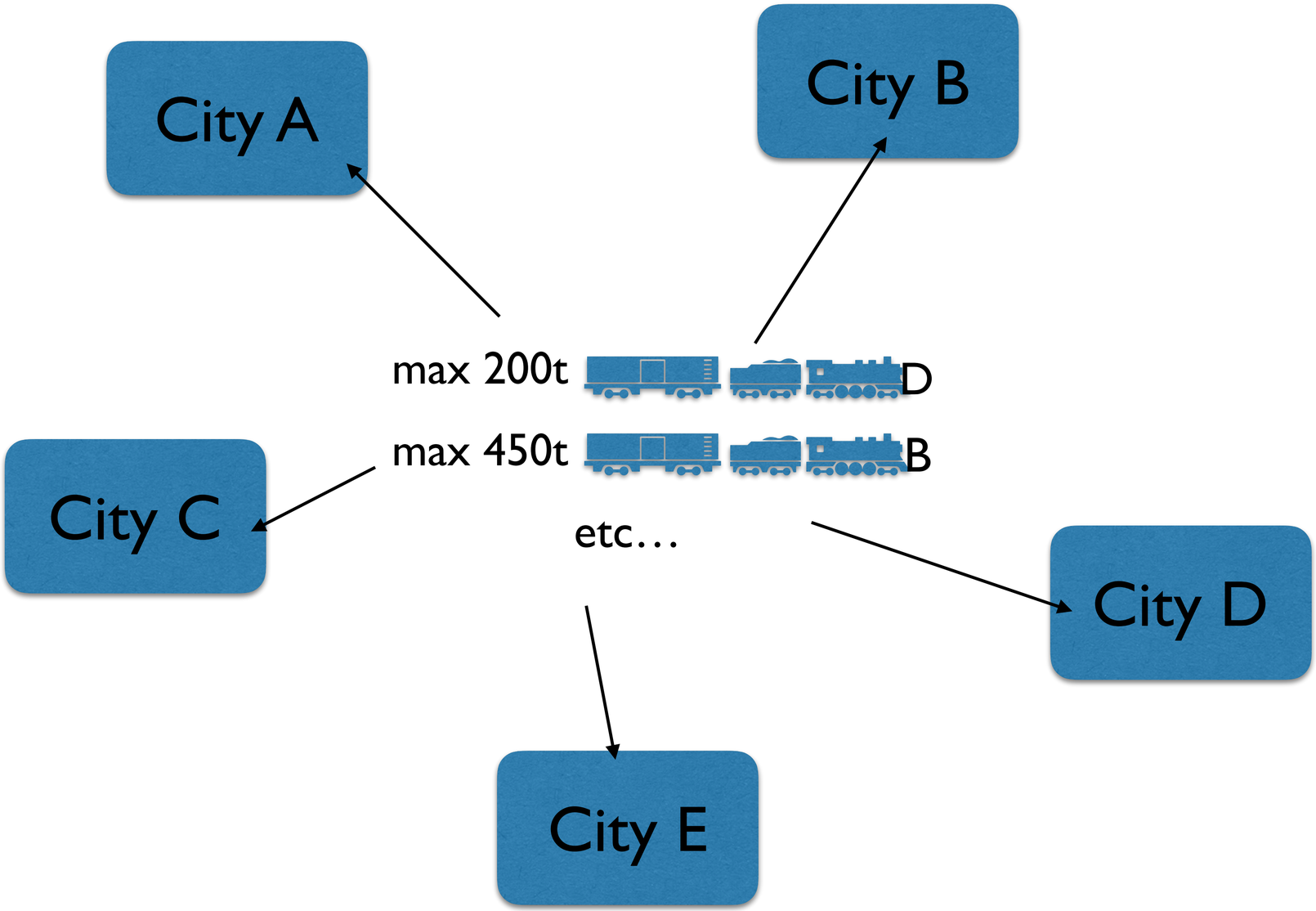
Hands-on Class

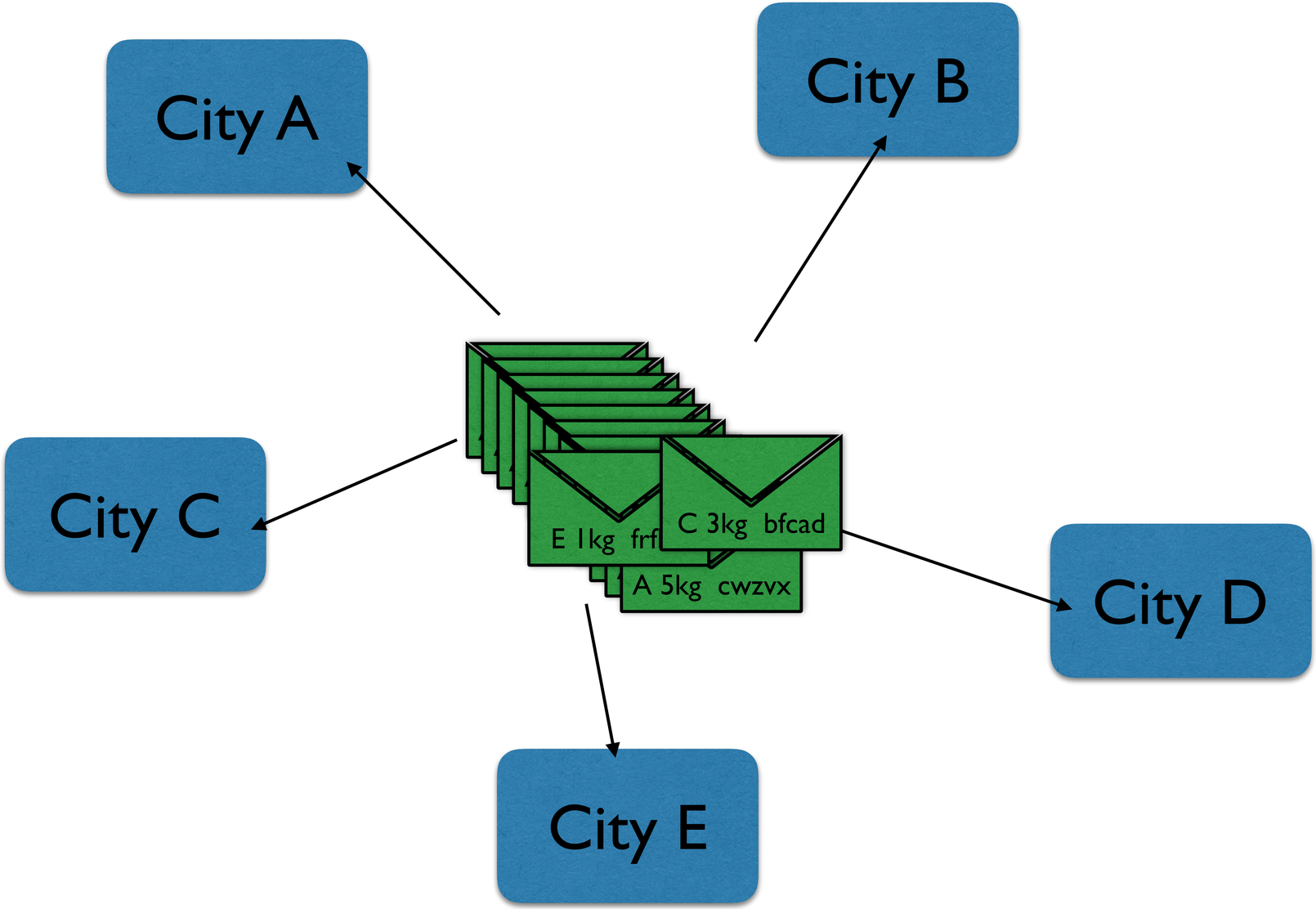
Basic proceadure

- Create classes (blueprints)
  - Contains attributes (data)
  - Methods (functions)
- Instances of classes are called **objects**
  - Pass in data, stores and manipulates them according to the blueprint
- Can have multiple entities **inheriting** from the same class and holding different attributes or methods
- Take advantage of polymorphism.

Exercise: a freight station



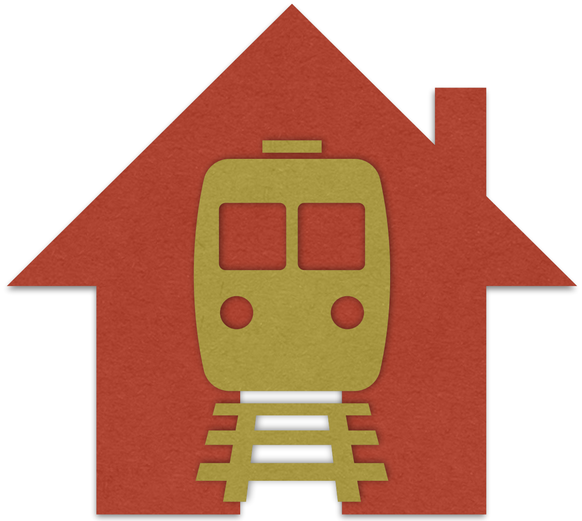

City A
City B
max 200t
D
max 450t
B
City C
etc…
City D
City E


City A
City B
City C
City D
City E
E 1kg frf
C 3kg bfcad
A 5kg cwzvx

Design an OO model for the station

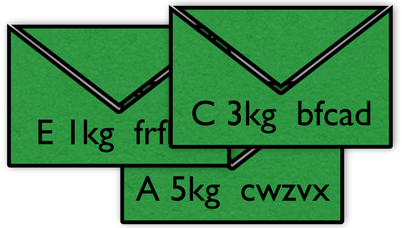
classes, objects, interfaces, public/private, which methods/state

but no implementation!



a random train arrives, is loaded with correct mail, leaves, and repeat

max 200t



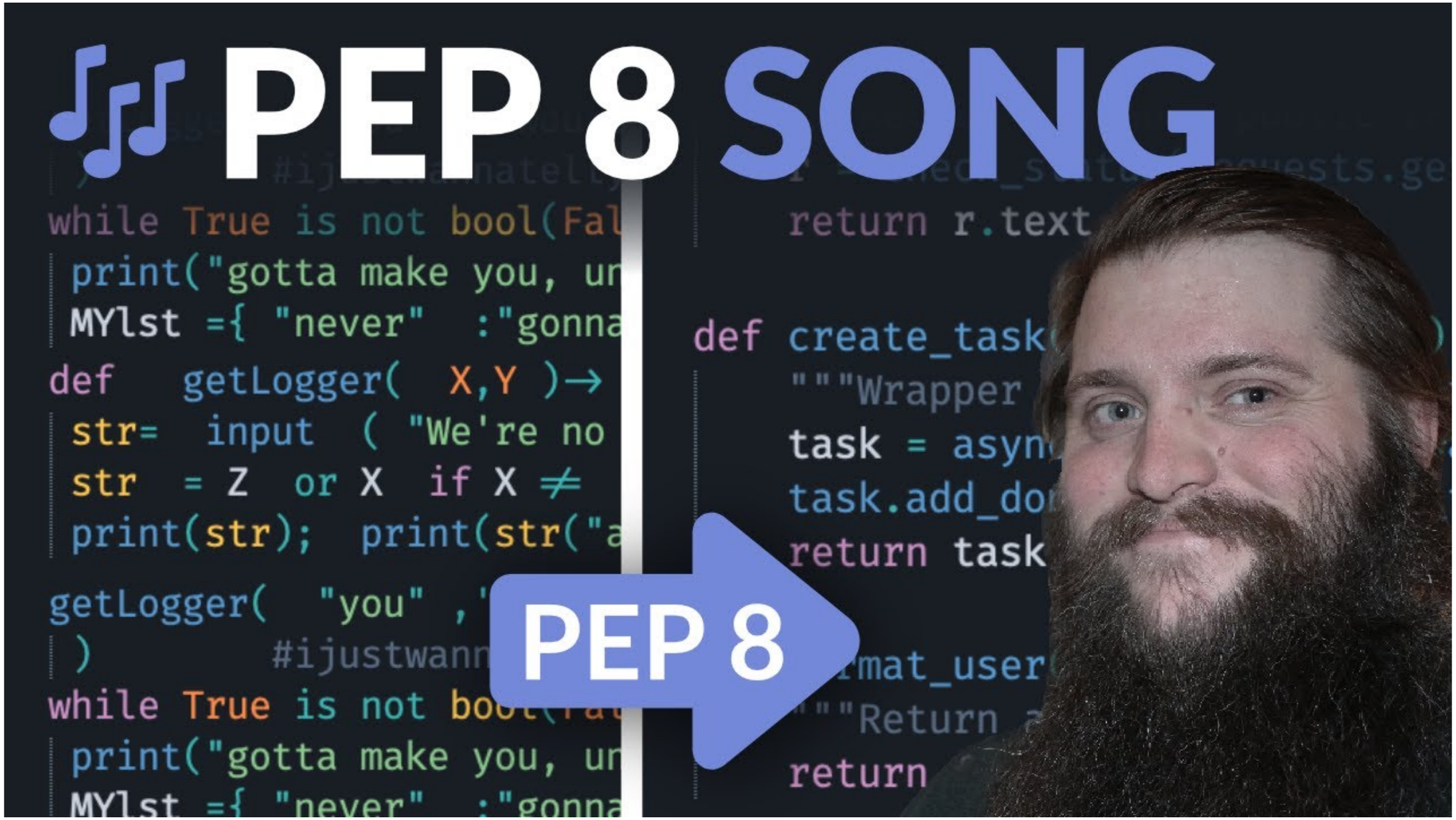
PEP 8

Python **E**nhancement **P**roposal

- code is read much more often than it is written
- improve the readability
- make it consistent across the wide spectrum of Python code
- You might find good reasons to ignore a particular guideline

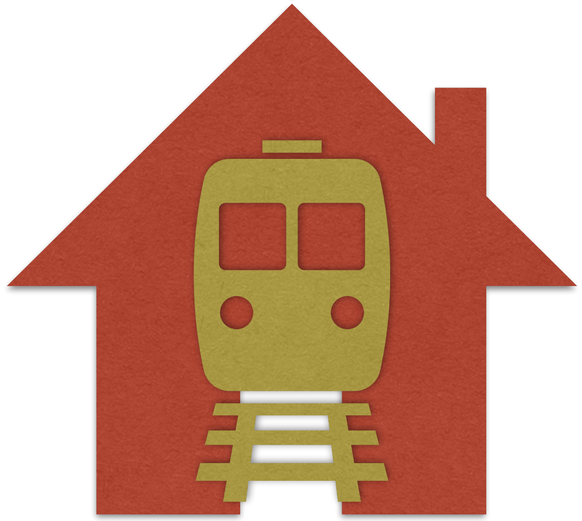
- https://www.python.org/dev/peps/pep-0008/
- https://pep8.org

PEP 8 Song


PEP 8 SONG
while True is not bool(Fal
print("gotta make you, ur
MYlst ={ "never" :"gonna
def getLogger(X,Y)→
str= input ("We're no
str = Z or X if X ≠
print(str); print(str("a
getLogger("you" ,'
)
#ijustwann
while True is not boo
print("gotta make you, ur
MYlst ={ "never" :"gonna
return r.text
def create_task
"""Wrapper
task = asyn
task.add_do
return task
rmat_user
"""Return a
return
PEP 8

Design an OO model for the station

classes, objects, interfaces, public/private, which methods/state

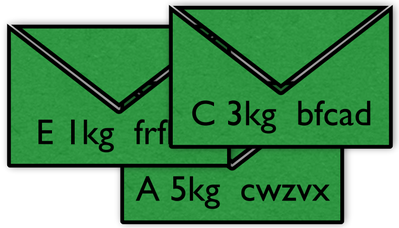


Write the props of each train to a separate file named train###.txt

Train: 001
Destination: D
Capacity: 140000
Actual load: 139999.31
Number of parcels: 40015
Parcels:
D 3.125
D 3.817

a random train arrives, is loaded with correct mail, leaves, and repeat

max 200t 



Hands on Sphinx

Install Sphinx

Windows, Mac, Linux

https://www.sphinx-doc.org/en/1.3.1/install.html

Sphinx - get started

- Change directory to code repository
  - $ sphinx-quickstart
- Separate source and build directories (y/n) [n]: y
- Enter project details
- You can make your html page by typing
  - $ make html
- Done

Home Page

- Your home page is “index.rst” found in the “source” directory
- Restructured Text (reST) format guide:
  - https://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html

RST Basics

*****
Title
*****

subtitle
****

subsubtitle
****

Only two rules:

- If **under and overline** are used, their **length must be identical**
- The length of the underline must be at least as long as the title itself

You can use any of these characters for the under/overline. However, it is better to stick to the same convention throughout a project. For example:

with overline, for parts
* with overline, for chapters
=, for sections
-, for subsections
^, for subsubsections
“, for paragraphs

Implicit Links to Titles

index.rst

```
========
Main topic
========
Blah blah blah blah
We put an internal reference to topic1 like this `Topic1`_
And here is another link `Topic2`_  that points to topic2


Topic1
******
Something something

Topic2
++++++
Something something
```

Appears as:

Main topic

Blah blah blah blah
We put an internal reference to topic1 like this
Topic1
And here is another link Topic2 that points to topic2

Topic1

Something something

Topic2

Something something

Toctree

```
.. toctree::
   :maxdepth: 2
   :caption: Contents:

   file1
   file2
   file3
```

- Multiple files need to be added to the toctree
- You can also modify toctree behaviour with:
  - **maxdepth**
  - **numbered**
  - **titlesonly**
  - **glob**
  - **hidden**
- More info here

Links to other files

File1.rst

```
.. _interesting_topic:
====
topic
====

Blah blah blah blah
```

File2.rst

```
You can find more interesting
topics at interesting_topic_
```

Or

```
You can find more intersting
topics at :ref: `intersting_topic`
```

Or

```
You can find more intersting
topics at :ref: `here
<intersting_topic>`
```

Appears as:

You can find more interesting topics at topic

You can find more interesting topics at here

Adding code path to sphinx

Add the path to codes bu uncommenting the following lines in conf.py, found in source.

```
import os
import sys
#the code in this example is found one directory above the source
sys.path.insert(0, os.path.abspath('../'))
sys.setrecursionlimit(1500)
```

Get module string doc

In conf.py add extension
extensions = ['sphinx.ext.autodoc']

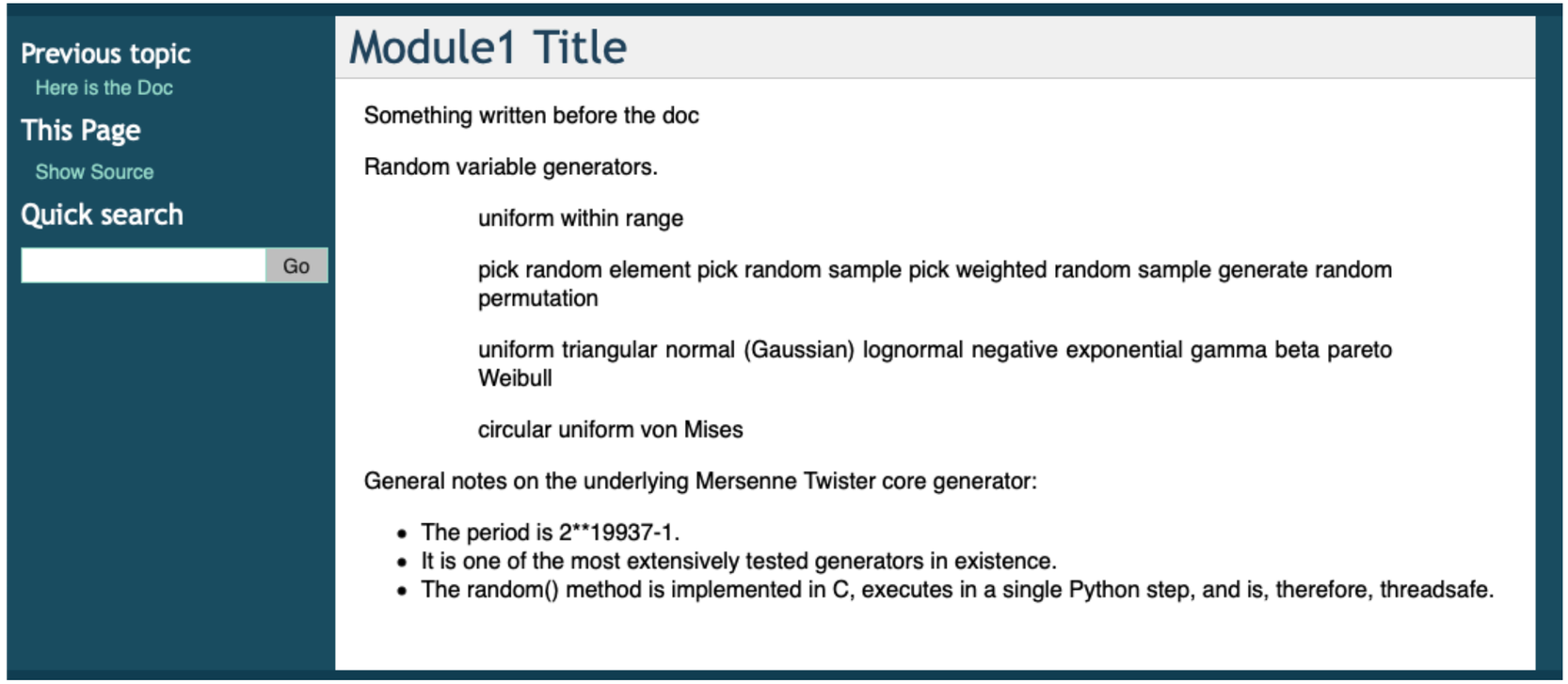
module1.py

```
"""Random variable generators.

    integers
    --------
           uniform within range

    sequences
    ---------
           pick random element
           pick random sample
           pick weighted random sample
           generate random permutation

    distributions on the real line:
    ------------------------------
           uniform
           triangular
           normal (Gaussian)
           lognormal
           negative exponential
           gamma
           beta
           pareto
           Weibull

    distributions on the circle (angles 0 to 2pi)
    ---------------------------------------------
           circular uniform
           von Mises

General notes on the underlying Mersenne Twister core generator:

* The period is 2**19937-1.
* It is one of the most extensively tested generators in existence.
* The random() method is implemented in C, executes in a single Python step,
  and is, therefore, threadsafe.

"""

from warnings import warn as _warn
from types import MethodType as _MethodType, BuiltinMethodType as _BuiltinMethodType
from math import log as _log, exp as _exp, pi as _pi, e as _e, ceil as _ceil
from math import sqrt as _sqrt, acos as _acos, cos as _cos, sin as _sin
from os import urandom as _urandom
from _collections_abc import Set as _Set, Sequence as _Sequence
from hashlib import sha512 as _sha512
import itertools as _itertools
import bisect as _bisect
import os as _os

__all__ = ["Random","seed","random","uniform","randint","choice","sample",
           "randrange","shuffle","normalvariate","lognormvariate",
           "expovariate","vonmisesvariate","gammavariate","triangular",
           "gauss","betavariate","paretovariate","weibullvariate",
           "getstate","setstate", "getrandbits", "choices",
           "SystemRandom"]

NV_MAGICCONST = 4 * _exp(-0.5)/_sqrt(2.0)
TWOPI = 2.0*_pi
LOG4 = _log(4.0)
SG_MAGICCONST = 1.0 + _log(4.5)
BPF = 53        # Number of bits in a float
RECIP_BPF = 2**-BPF


# Translated by Guido van Rossum from C source provided by
```

module.rst

```
Module1 Title
=============
Something written before the doc

.. automodule:: module1
```

Appers as

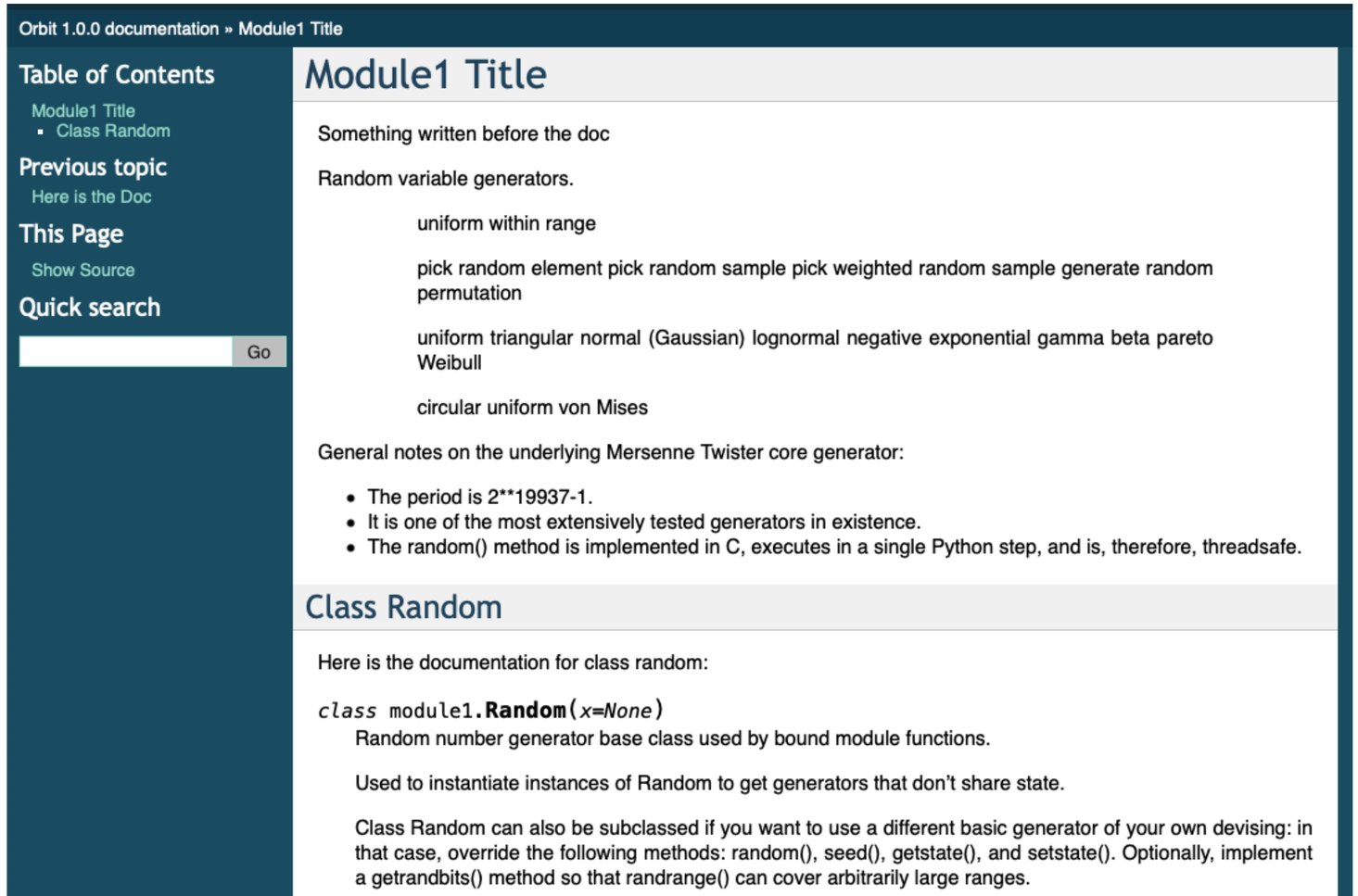


Get class string doc

module1.py

```
NV_MAGICCONST = 4 * _exp(-0.5)/_sqrt(2.0)
TWOPI = 2.0*_pi
LOG4 = _log(4.0)
SG_MAGICCONST = 1.0 + _log(4.5)
BPF = 53        # Number of bits in a float
RECIP_BPF = 2**-BPF


# Translated by Guido van Rossum from C source provided by
# Adrian Baddeley.  Adapted by Raymond Hettinger for use with
# the Mersenne Twister  and os.urandom() core generators.

import _random

class Random(_random.Random):
    """Random number generator base class used by bound module functions.

    Used to instantiate instances of Random to get generators that don't
    share state.

    Class Random can also be subclassed if you want to use a different basic
    generator of your own devising: in that case, override the following
    methods:  random(), seed(), getstate(), and setstate().
    Optionally, implement a getrandbits() method so that randrange()
    can cover arbitrarily large ranges.

    """

    VERSION = 3     # used by getstate/setstate

    def __init__(self, x=None):
        """Initialize an instance.

        Optional argument x controls seeding, as for Random.seed().
        """

        self.seed(x)
        self.gauss_next = None

    def seed(self, a=None, version=2):
        """Initialize internal state from hashable object.

        None or no argument seeds from current time or from an operating
        system specific randomness source if available.

        If *a* is an int, all bits are used.
```

Appers as



module.rst

```
Module1 Title
=============
Something written before the doc

.. automodule:: module1

Class Random
++++++++++++++

Here is the documentation for class random:

.. autoclass:: module1.Random
```

Get class string doc + methods

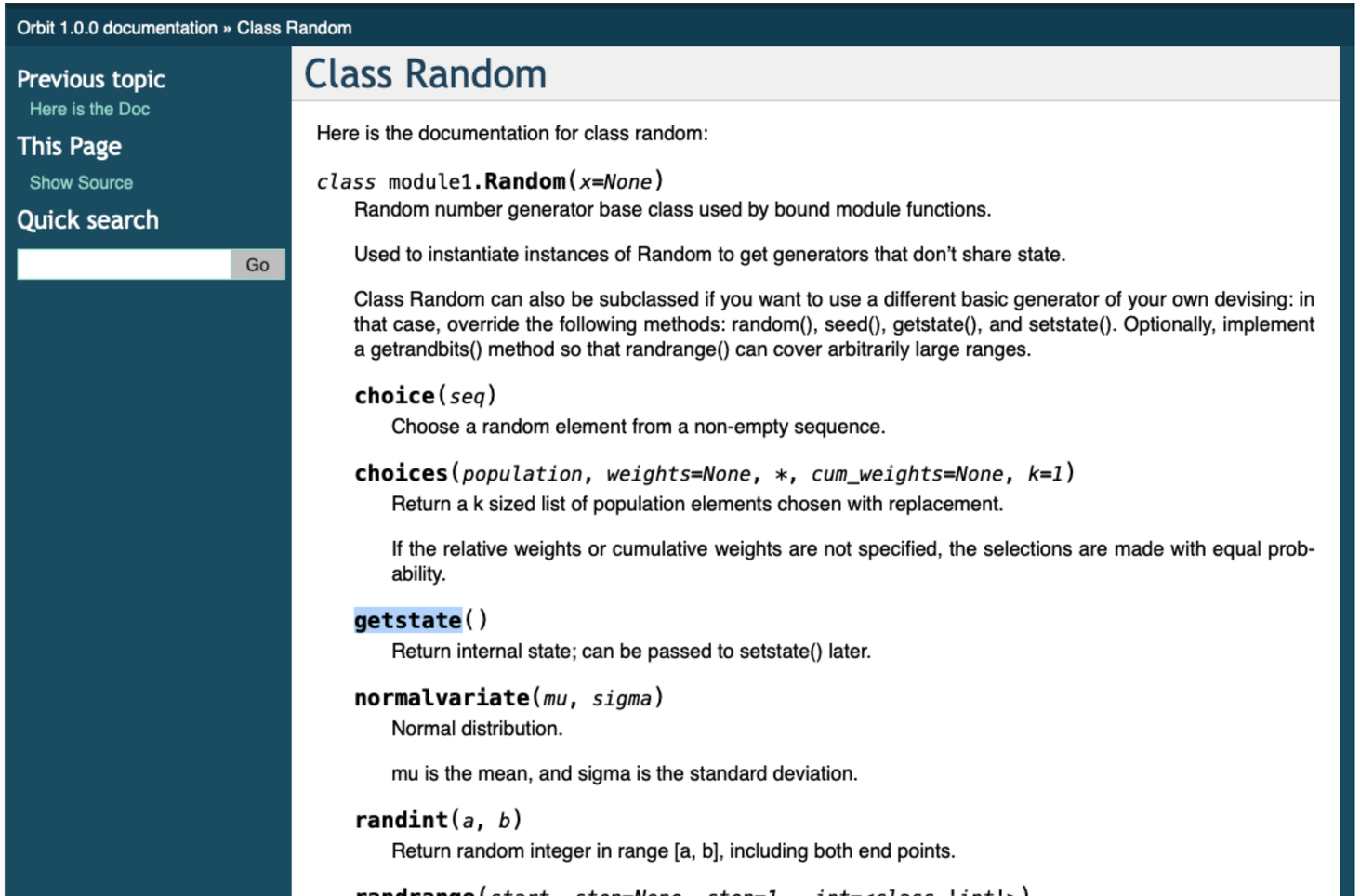
module1.py

```
NV_MAGICCONST = 4 * _exp(-0.5)/_sqrt(2.0)
TWOPI = 2.0*_pi
LOG4 = _log(4.0)
SG_MAGICCONST = 1.0 + _log(4.5)
BPF = 53        # Number of bits in a float
RECIP_BPF = 2**-BPF


# Translated by Guido van Rossum from C source provided by
# Adrian Baddeley.  Adapted by Raymond Hettinger for use with
# the Mersenne Twister  and os.urandom() core generators.

import _random

class Random(_random.Random):
    """Random number generator base class used by bound module functions.

    Used to instantiate instances of Random to get generators that don't
    share state.

    Class Random can also be subclassed if you want to use a different basic
    generator of your own devising: in that case, override the following
    methods:  random(), seed(), getstate(), and setstate().
    Optionally, implement a getrandbits() method so that randrange()
    can cover arbitrarily large ranges.

    """

    VERSION = 3     # used by getstate/setstate

    def __init__(self, x=None):
        """Initialize an instance.

        Optional argument x controls seeding, as for Random.seed().
        """

        self.seed(x)
        self.gauss_next = None

    def seed(self, a=None, version=2):
        """Initialize internal state from hashable object.

        None or no argument seeds from current time or from an operating
        system specific randomness source if available.

        If *a* is an int, all bits are used.
```

module.rst

```
Class Random
++++++++++++++

Here is the documentation for class random:

.. autoclass:: module1.Random
   :members:
```

Appers as



Get function string doc

function.rst

```
Function Example
++++++++++++++++

Get function doc strings from module2.py

.. currentmodule:: module2

The run function
""""""""""""""""

short description...

.. autofunction:: run


The runctx function
###################

Write something here


.. autofunction:: runctx
```

Appers as