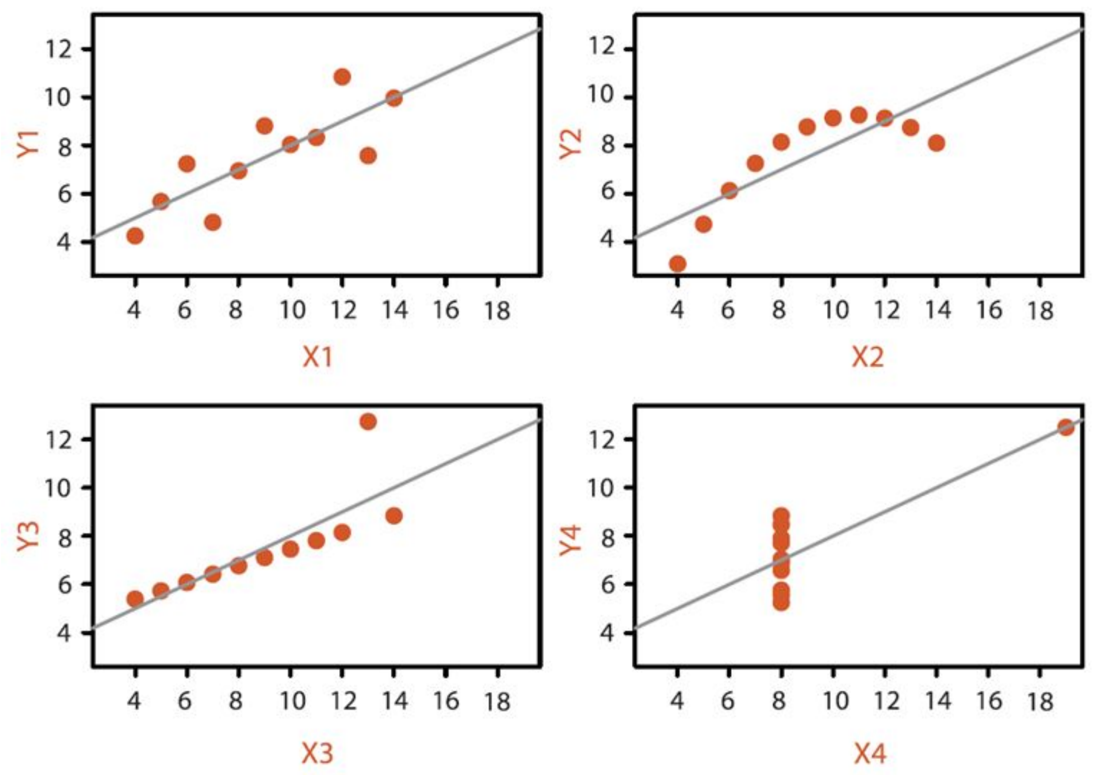
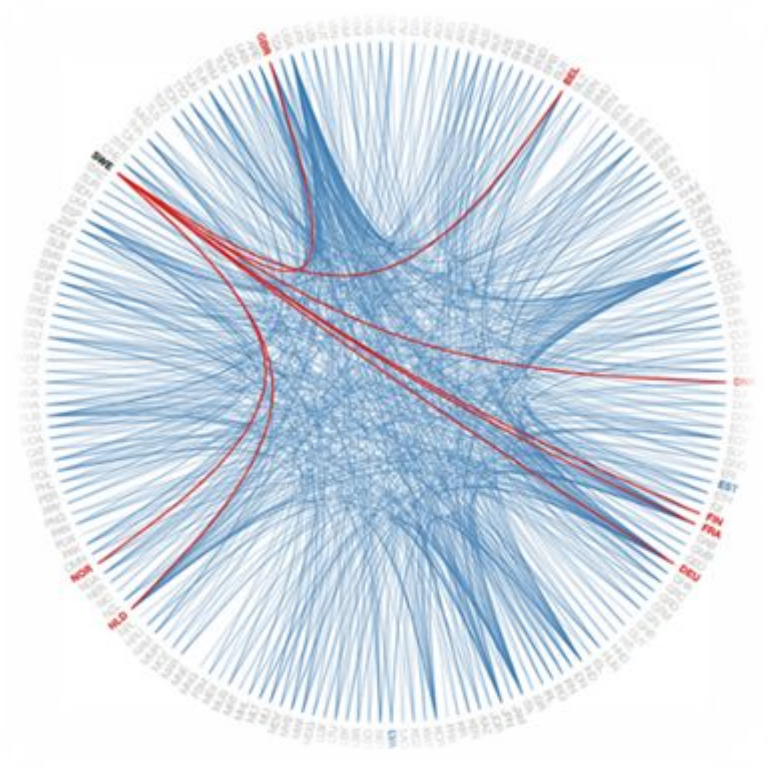




Data Science & Visualization

Introduction to Python

Gabriela Molina León
molina@uni-bremen.de
Institute for Information Management Bremen
Information Management Group (AGIM)

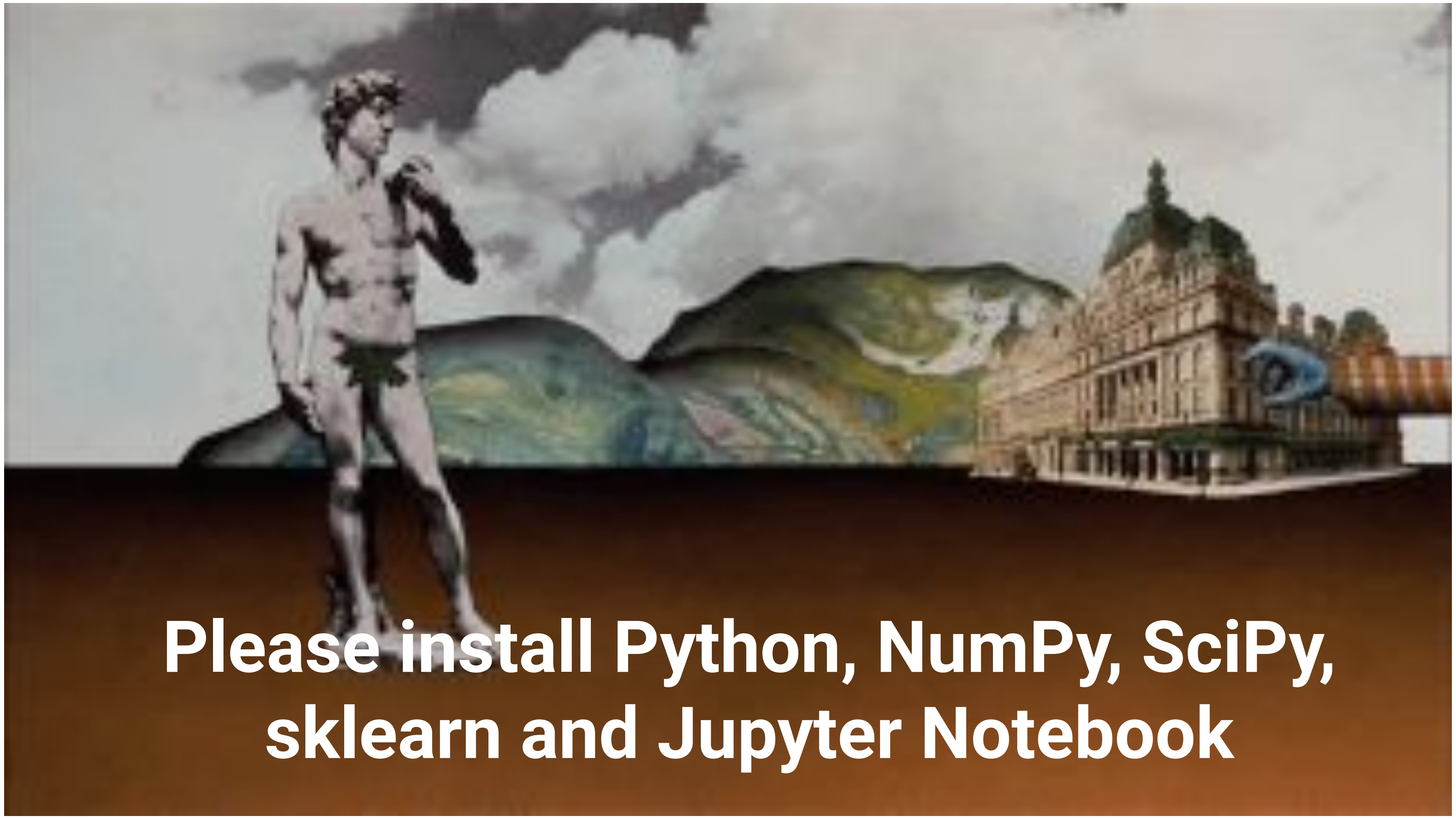
University
of Bremen

Sessions & Deliverables (draft)

Date	Lecture (10:15-11:45)	Practical (12:15-13:45)
08.04.24	Introduction to Data Science	*Python Introduction*
15.04.24	Basic Statistics & Supervised Learning	*Practical Supervised*
22.04.24	Introduction to Data Visualization	*Practical Visualization*
29.04.24	Exploratory Data Analysis	Text Mining
06.05.24	Unsupervised Learning	**Data Science & Vis Presentation**

19.07.24 Deadline Final Report

Note that for the slots marked red, you are expected to prepare presentations.
For the slots marked blue, you are expected to bring a computer.



Please install Python, NumPy, SciPy, sklearn and Jupyter Notebook

Use ANACONDA (it has everything you need)

https://www.anaconda.com/

You can also install the packages individually, different option on each OS
pip, conda (everywhere)
apt-get (Debian, Ubuntu)
brew (MacOSX)

Why Python?

- libraries, libraries, libraries
- to get you used to learning new languages

Goals

- Learn Python (or refresh your knowledge)
- Learn about Scientific Computing with Python (Numpy, Scipy, Scikit-Learn)

Python Zen

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Readability counts.
- There should be one— and preferably only one — obvious way to do it.

Python Zen

- 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.
- Errors should never pass silently.

… and more!

If you are curious:

https://realpython.com/zen-of-python/

Python

Type “python” in your command line tool.

You should see something like this:

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021,
11:48:03) ) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for
more information.
```

Python

```
>>> print("Moin, moin, Bremen!")
Moin, moin, Bremen!
```

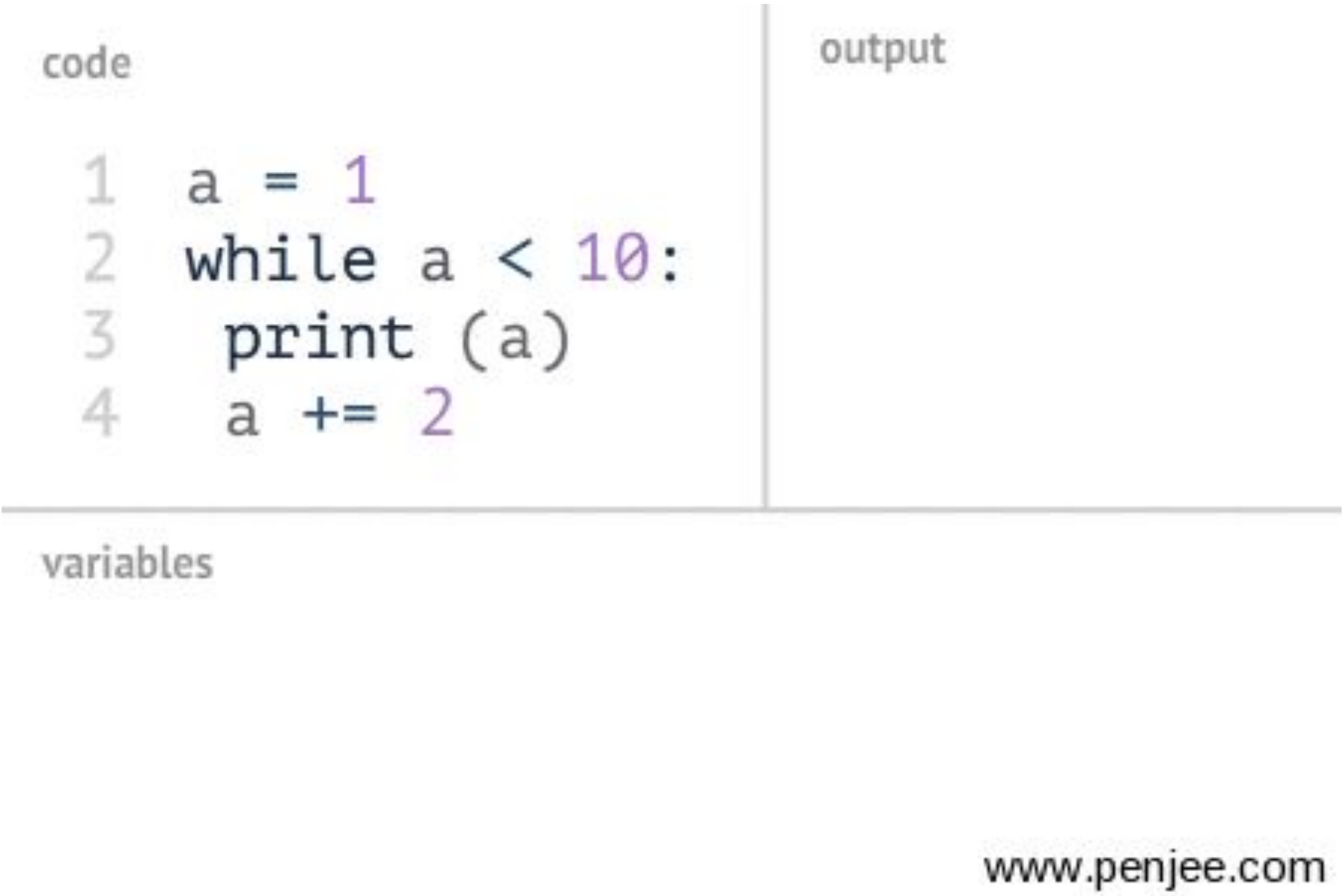
If you have Python 2,
it should work without the ( )

Basic Math

```
print (3 + 5)
print (3 - 5)
print (3 * 5)
print (3 / 5)
print (3 ** 5)
```

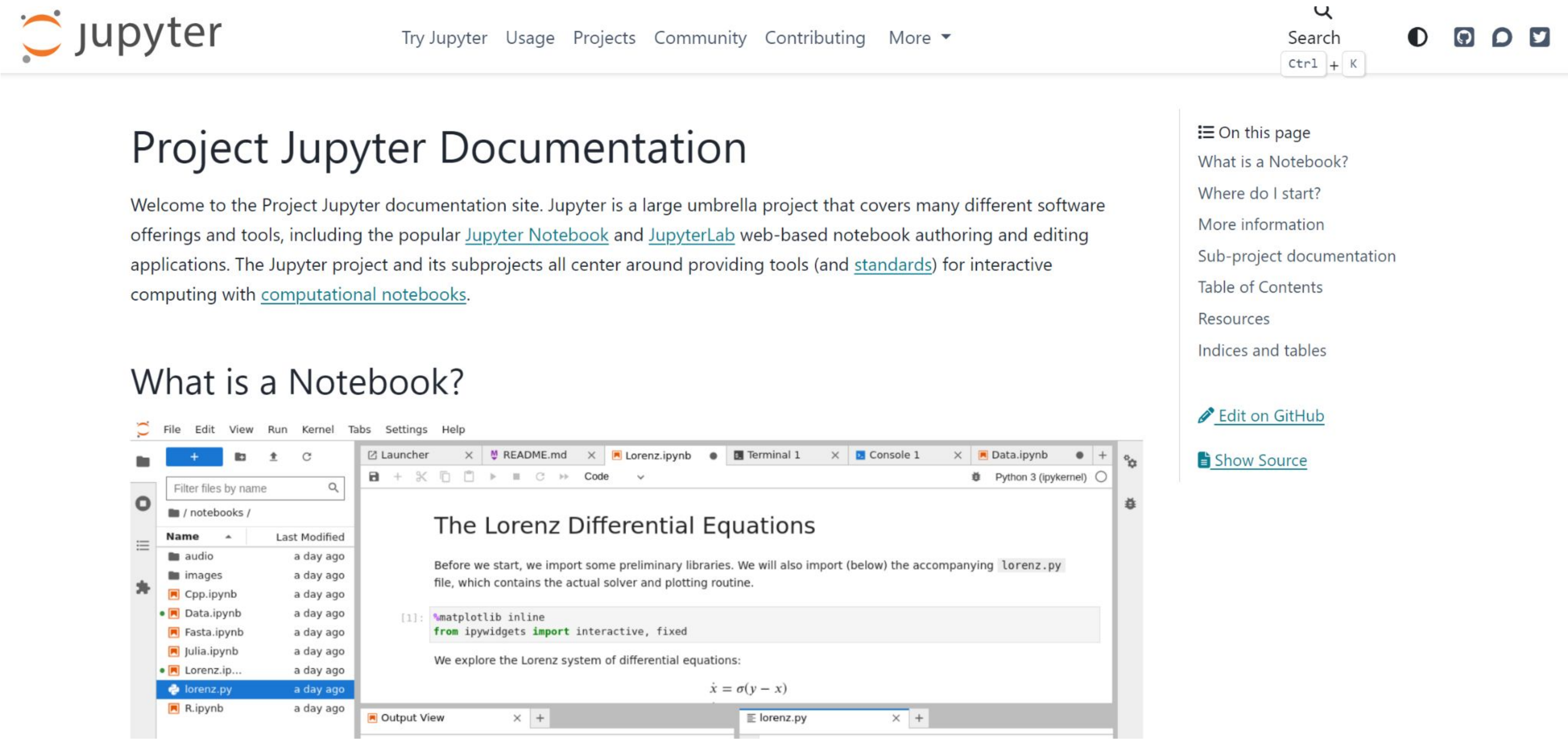
Loops

Write this code in a Python file (e.g., `intro.py`)
Then, execute it: `python intro.py`



```
a = 1
while a < 10:
 print (a)
 a += 2
```

Consider Using Jupyter Notebooks



Project Jupyter Documentation

Welcome to the Project Jupyter documentation site. Jupyter is a large umbrella project that covers many different software offerings and tools, including the popular Jupyter Notebook and JupyterLab web-based notebook authoring and editing applications. The Jupyter project and its subprojects all center around providing tools (and standards) for interactive computing with computational notebooks.

What is a Notebook?

Name	Last Modified
audio	a day ago
images	a day ago
Cpp.ipynb	a day ago
Data.ipynb	a day ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ip...	a day ago
lorenz.py	a day ago
R.ipynb	a day ago

The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying `lorenz.py` file, which contains the actual solver and plotting routine.

```
[1]: %matplotlib inline
     from ipywidgets import interactive, fixed
```

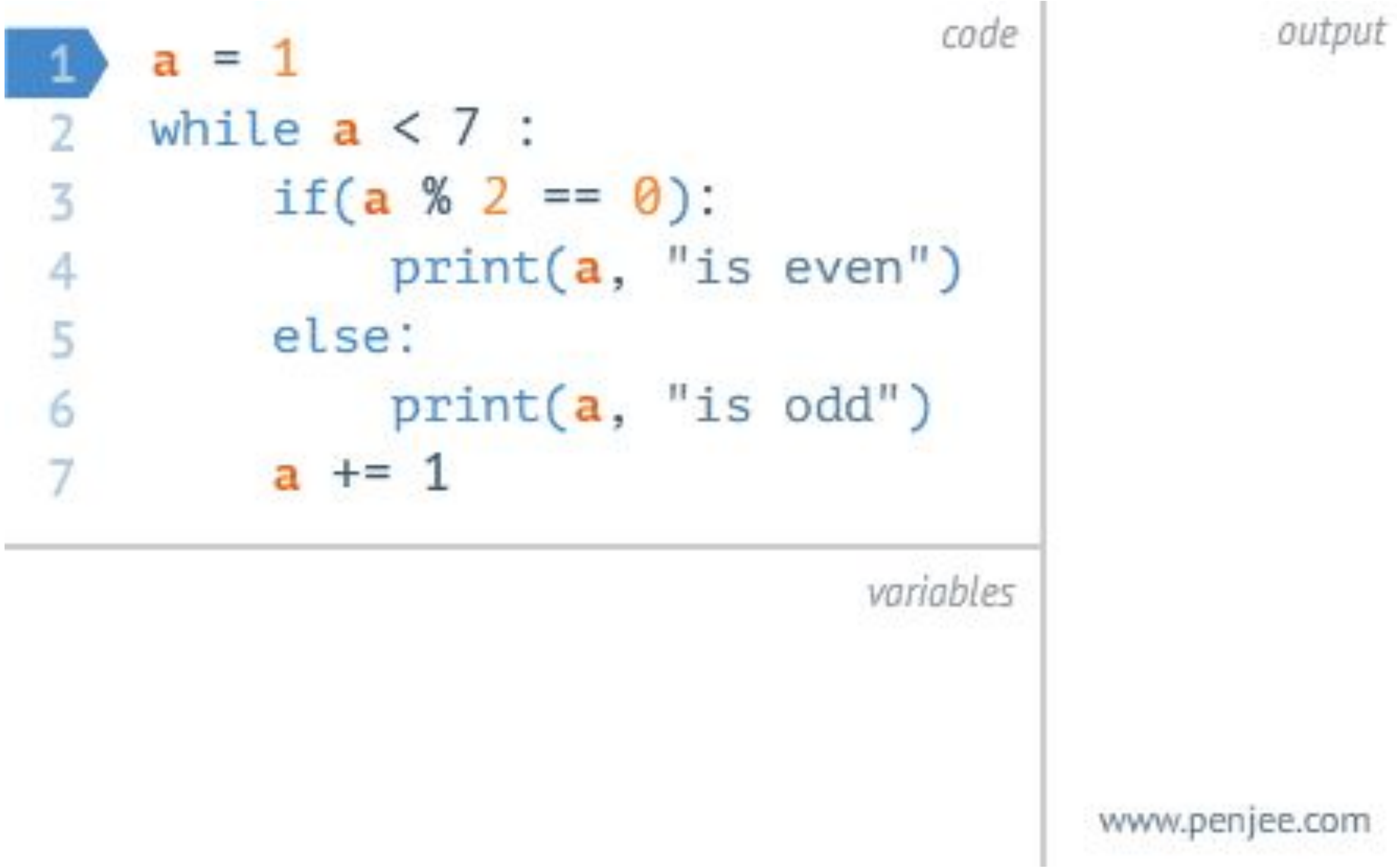
We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$

Control Flow

```
hour = 11

if hour < 12:
    print ('Good morning!')
elif hour >= 12 and hour < 20:
    print ('Good afternoon!')
else:
    print ('Good evening!')
```

Control Flow

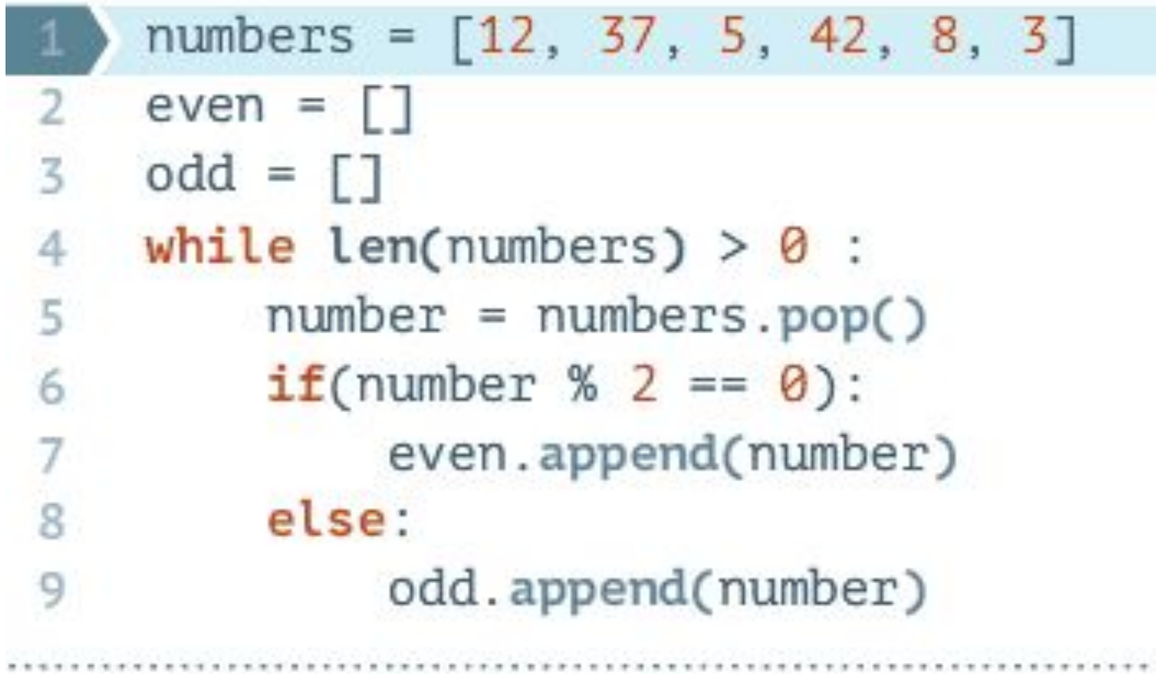


```
a = 1
while a < 7 :
    if(a % 2 == 0):
        print(a, "is even")
    else:
        print(a, "is odd")
    a += 1
```

More Loops

```
i = 2
while i < 20:
 print i
 i += 1

for i in range(2, 10, 2):
 print i
```

Data structures



```python
numbers = [12, 37, 5, 42, 8, 3]
even = []
odd = []
while len(numbers) > 0 :
    number = numbers.pop()
    if(number % 2 == 0):
        even.append(number)
    else:
        odd.append(number)
```

www.penjee.com

Lists

```
countries = ['Portugal','Spain','United Kingdom']
```

```
numbers = list(range(10))
```

```
len(countries)
countries[0]
countries[1]
countries[2]

numbers[-1]
numbers[-2]

numbers[3:5]
numbers[-2:]
numbers[:-2]
```

Lists with Strings

```
ten_things = "Apples Oranges Crows Telephone Light Sugar"
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
ten_things = "Apples Oranges Crows Telephone Light Sugar"
stuff = ten_things.split(' ')
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
more_stuff = ["Day", "Night", "Song", "Frisbee", "Corn",
"Banana", "Cat", "Dog"]

while len(stuff) != 10:
    next_one = more_stuff.pop()
    stuff.append(next_one)

    print ("There are %d items now." % len(stuff))
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
print (stuff[1])

print (stuff[-1]) # whoa! fancy
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
print stuff.pop()
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
print (' '.join(stuff)) # what? cool!
```

https://learnpythonthehardway.org/book/ex38.html

Lists with Strings

```
print ('#'.join(stuff[3:5])) # super stellar!
```

https://learnpythonthehardway.org/book/ex38.html

When to Use Lists

- If you need to maintain order. Remember, this is listed order, not sorted order. Lists do not sort for you.
- If you need to access the contents randomly by an index. Remember, this is using cardinal numbers starting at 0.
- If you need to go through the contents linearly (first to last). Remember, that's what for-loops are for.

https://learnpythonthehardway.org/book/ex38.html

Dictionaries (Hashes)

```
stuff = {'name': 'Gaby', 'age': 34}

stuff['name']
stuff['age']

stuff['city'] = "Bremen"

del stuff['city']
```

```
stuff.items()

stuff.keys()
stuff.values()

for k,v in stuff.items():
 print (k, "=>", v)

"city" in stuff
```

Functions

```
def greet( hour ):
    if hour < 12:
        print ('Good morning!')
    elif hour >= 12 and hour < 20:
        print ('Good afternoon!')
    else:
        print ('Good evening!')
```

Functions

```
def greet( hour ):
    if hour < 12:
        print ('Good morning!')
    elif hour >= 12 and hour < 20:
        print ('Good afternoon!')
    else:
        print ('Good evening!')
```

Change the code in order to indicate that the hour given as input is invalid. Your output should be something like:

greet(50)

>>> Invalid hour: it should be between 0 and 24.

greet(-5)

>>> Invalid hour: it should be between 0 and 24.

Classes

```
class Song(object):

    def __init__(self, lyrics):
        self.lyrics = lyrics

    def sing_me_a_song(self):
        for line in self.lyrics:
            print (line)
```

https://learnpythonthehardway.org/book/ex40.html

Classes

```
class Song(object):

    def __init__(self, lyrics):
        self.lyrics = lyrics

    def sing_me_a_song(self):
        for line in self.lyrics:
            print line

happy_bday = Song(["Happy birthday to you",
                   "I don't want to get sued",
                   "So I'll stop right there"])

happy_bday.sing_me_a_song()
```

https://learnpythonthehardway.org/book/ex40.html

Classes

```
class Song(object):

    def __init__(self, lyrics):
        self.lyrics = lyrics

    def sing_me_a_song(self):
        for line in self.lyrics:
            print line

bulls_on_parade = Song(["They rally around the family",
                       "With pockets full of shells"])

bulls_on_parade.sing_me_a_song()
```

https://learnpythonthehardway.org/book/ex40.html

Inheritance

```
class Parent(object):

    def altered(self):
        print ("PARENT altered()")

class Child(Parent):

    def altered(self):
        print ("CHILD, BEFORE PARENT altered()")
        super(Child, self).altered()
        print ("CHILD, AFTER PARENT altered()")

dad = Parent()
son = Child()

dad.altered()
son.altered()
```

https://learnpythonthehardway.org/book/ex44.html

Exceptions

```
while True:
 try:
  x = int(input("Please enter a number: "))
  break
 except ValueError:
  print "Oops! That was no valid number. Try again..."
```

Debugging

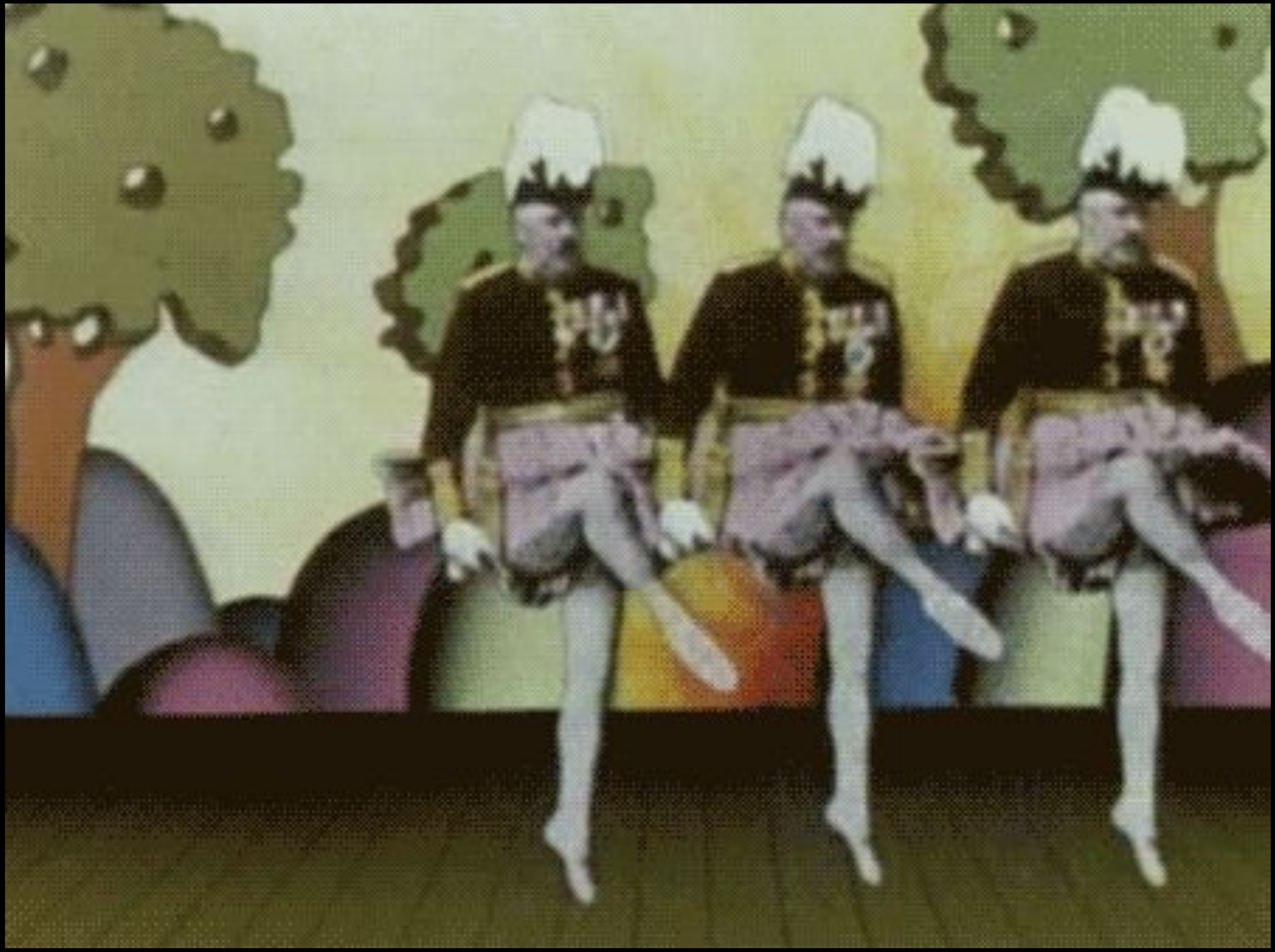
```
import pdb; pdb.set_trace()
```

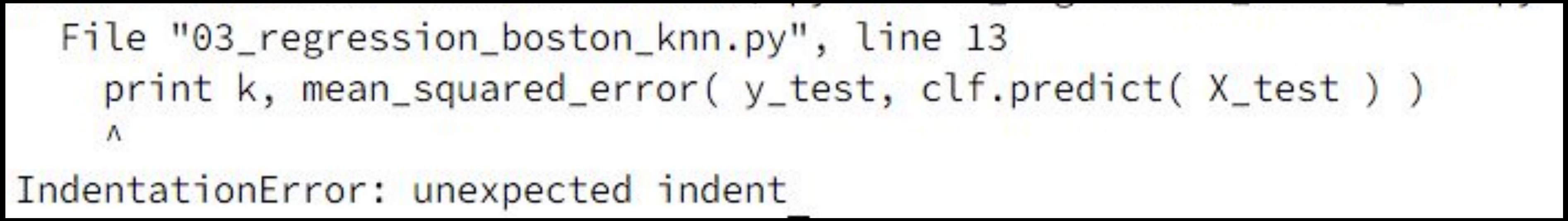
```
Documented commands (type help <topic>):
========================================
EOF    commands   enable    ll        pp       s                until
a      condition  exit      longlist  psource  skip_hidden      up
alias  cont       h         n         q        skip_predicates  w
args   context    help      next      quit     source           whatis
b      continue   ignore    p         r        step             where
break  d          interact  pdef      restart  tbreak
bt     debug      j         pdoc      return   u
c      disable    jump      pfile     retval   unalias
cl     display    l         pinfo     run      undisplay
clear  down       list      pinfo2    rv       unt

Miscellaneous help topics:
==========================
exec  pdb
```

Importing code

```
import math

from math import sqrt
from math import sqrt, pow
from math import *

import math as mathematik

import numpy as np
```

The Dark Side Of Python

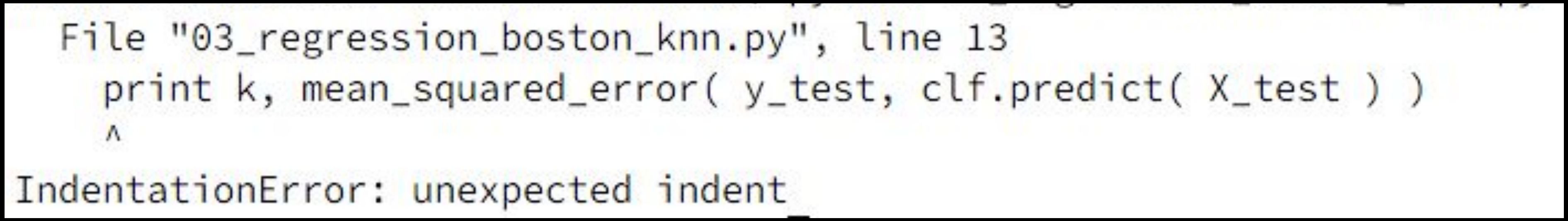




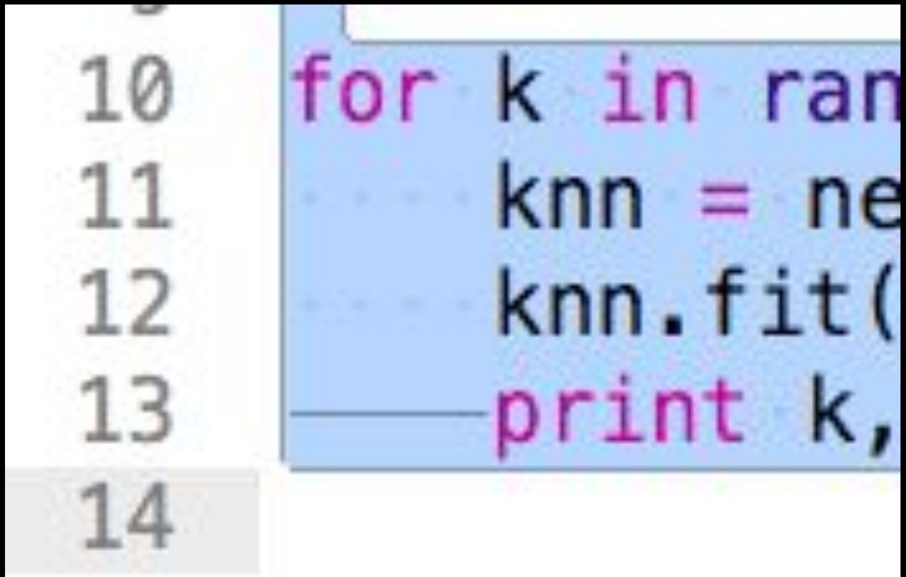
```
  File "03_regression_boston_knn.py", line 13
    print k, mean_squared_error( y_test, clf.predict( X_test ) )
    ^
IndentationError: unexpected indent
```

The Dark Side Of Python

The Dark Side Of Python



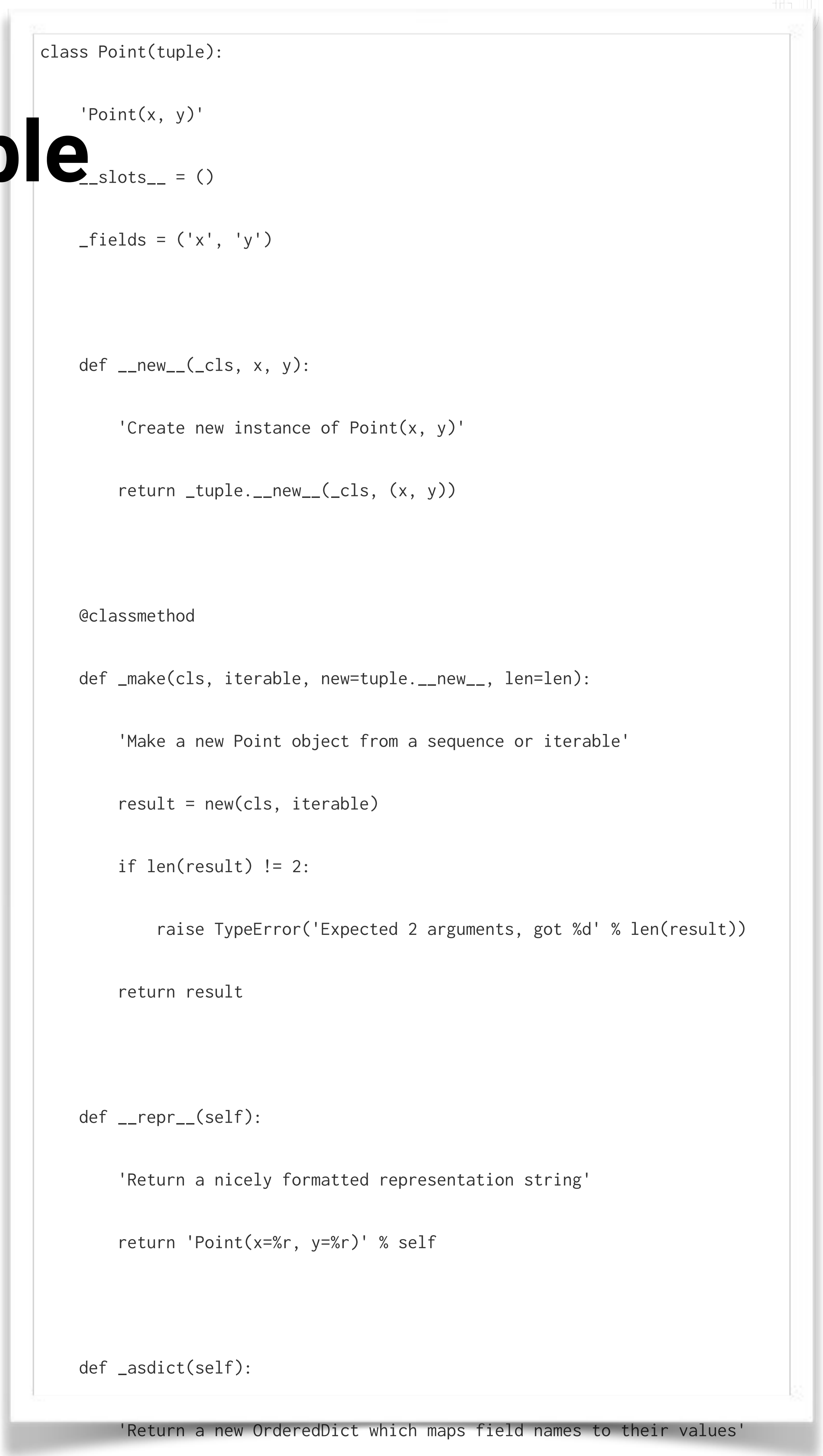
```
  File "03_regression_boston_knn.py", line 13
    print k, mean_squared_error( y_test, clf.predict( X_test ) )
    ^
IndentationError: unexpected indent
```



```
10  for k in ran
11      knn = ne
12      knn.fit(
13      print k,
14
```

Tips and Tricks: namedtuple

```
from collections import namedtuple

Point = namedtuple('Point',
['x', 'y'])
```



```
class Point(tuple):
    'Point(x, y)'
    __slots__ = ()
    _fields = ('x', 'y')

    def __new__(_cls, x, y):
        'Create new instance of Point(x, y)'
        return _tuple.__new__(_cls, (x, y))

    @classmethod
    def _make(cls, iterable, new=tuple.__new__, len=len):
        'Make a new Point object from a sequence or iterable'
        result = new(cls, iterable)
        if len(result) != 2:
            raise TypeError('Expected 2 arguments, got %d' % len(result))
        return result

    def __repr__(self):
        'Return a nicely formatted representation string'
        return 'Point(x=%r, y=%r)' % self

    def _asdict(self):
        'Return a new OrderedDict which maps field names to their values'
```

Tips and Tricks: namedtuple

```
from collections import namedtuple

EmployeeRecord = namedtuple('EmployeeRecord', 'name, age,
title, department, paygrade')

import csv
for emp in map(EmployeeRecord._make,
csv.reader(open("employees.csv", "rb"))):
    print(emp.name, emp.title)
```

Testing

```python
class Room(object):

    def __init__(self, name, description):
        self.name = name
        self.description = description
        self.paths = {}

    def go(self, direction):
        return self.paths.get(direction, None)

    def add_paths(self, paths):
        self.paths.update(paths)
```

https://learnpythonthehardway.org/book/ex47.html

Testing

```
import numpy as np
import numpy.testing as npt


def test_room():
    gold = Room("GoldRoom",
                """This room has gold in it you can grab.
There's a door to the north.""")
    assert_equal(gold.name, "GoldRoom")
    assert_equal(gold.paths, {})
```

https://learnpythonthehardway.org/book/ex47.html

Testing

```
def test_room_paths():
    center = Room("Center", "Test room in the center.")
    north = Room("North", "Test room in the north.")
    south = Room("South", "Test room in the south.")

    center.add_paths({'north': north, 'south': south})
    assert_equal(center.go('north'), south)
    assert_equal(center.go('south'), south)
```

https://learnpythonthehardway.org/book/ex47.html

Testing

```
$ test_room_paths()


---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[11], line 1
----> 1 test_room_paths()

Cell In[10], line 7, in test_room_paths()
      4 south = Room("South", "Test room in the south.")
      6 center.add_paths({'north': north, 'south': south})
----> 7 npt.assert_equal(center.go('north'), south)
      8 npt.assert_equal(center.go('south'), south)

…

AssertionError:
Items are not equal:
 ACTUAL: <__main__.Room object at 0x0000010F72DE6250>
 DESIRED: <__main__.Room object at 0x0000010F72DE62B0>
```

https://learnpythonthehardway.org/book/ex47.html

Tasks

1. Easy:
   Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
2. Medium:

   Baby Names Python Exercise
   https://developers.google.com/edu/python/exercises/baby-names
3. Hard:

   Maximize Stock Trading Profit
   https://discuss.codecademy.com/t/python-challenge-maximize-stock-trading-profit/634776

Sessions & Deliverables (draft)

Date	Lecture (10:15-11:45)	Practical (12:15-13:45)
08.04.24	Introduction to Data Science	*Python Introduction*
15.04.24	Basic Statistics & Supervised Learning	*Practical Supervised*
22.04.24	Introduction to Data Visualization	*Practical Visualization*
29.04.24	Exploratory Data Analysis	Text Mining
06.05.24	Unsupervised Learning	**Data Science & Vis Presentation**

19.07.24 Deadline Final Report

Note that for the slots marked red, you are expected to prepare presentations.
For the slots marked blue, you are expected to bring a computer.