Jupyter Notebooks

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 1

Writing and running code

- Manual
  - Text editor to write code
  - Run at the command line

emacs@mmcarbon

File Edit Options Buffers Tools Python Help

```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)

def prime(n):
  return(factors(n) == [1,n])

primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

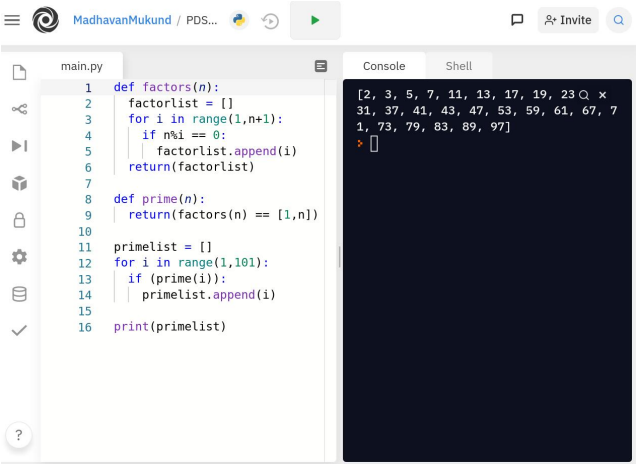
-:--- main.py All L1 (Python ElDoc)

File Edit View Search Terminal Help

```
$ python3
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from main import *
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
 79, 83, 89, 97]
>>>
```

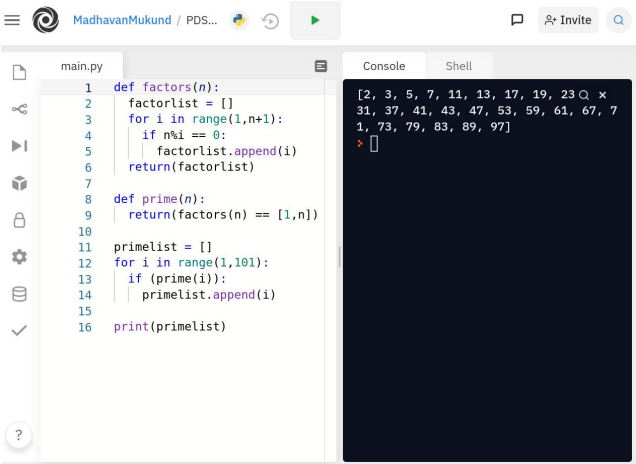
Writing and running code

- Manual
  - Text editor to write code
  - Run at the command line
- Integrated Development Environment (IDE)
  - Single application to write and run code
  - On desktop or online, `replit`
  - Quick update-run cycle
  - Debugging, testing, . . .



Writing and running code

- Manual
  - Text editor to write code
  - Run at the command line
- Integrated Development Environment (IDE)
  - Single application to write and run code
  - On desktop or online, `replit`
  - Quick update-run cycle
  - Debugging, testing, . . .
- What more could one want?



Collaboration

- Share your code
  - Collaborative development
  - Report your results

jupyter PDSA Week 1 Lecture 1 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

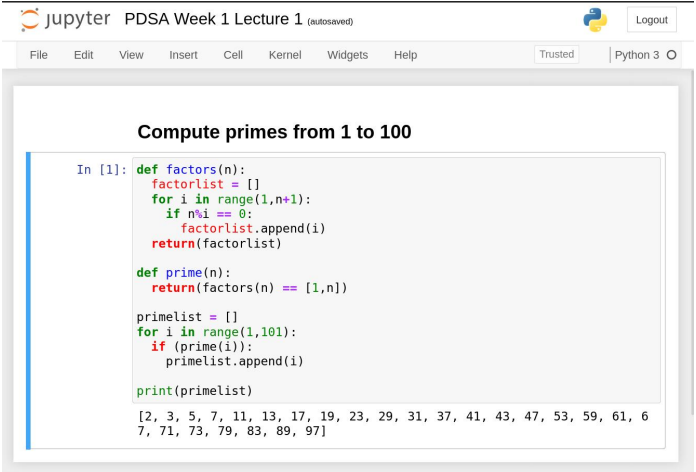
Compute primes from 1 to 100

In [1]:
```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)

def prime(n):
  return(factors(n) == [1,n])

primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 6
7, 71, 73, 79, 83, 89, 97]
```

Collaboration

- Share your code
  - Collaborative development
  - Report your results
- Documentation
  - Interleave with the code



Collaboration

- Share your code
  - Collaborative development
  - Report your results
- Documentation
  - Interleave with the code
- Switch between different versions of code

jupyter PDSA Week 1 Lecture 1 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Compute primes from 1 to 100

In [1]:
```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)
```

In [3]:
```
def prime(n):
  return(factors(n) == [1,n])
```

In [2]:
```
def prime(n):
  return(len(factors(n)) == 2)
```

In [4]:
```
primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 6
7, 71, 73, 79, 83, 89, 97]
```

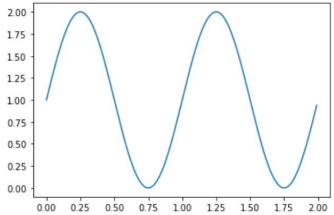
Collaboration

- Share your code
  - Collaborative development
  - Report your results
- Documentation
  - Interleave with the code
- Switch between different versions of code
- Export and import your project
- Preserve your output

jupyter PDSA Week 1 Lecture 1 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [6]:
```
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

plt.show()
```



Jupyter notebook

- A sequence of cells
  - Like a one dimensional spreadsheet

jupyter PDSA Week 1 Lecture 1 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Compute primes from 1 to 100

In [1]:
```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)
```

In [3]:
```
def prime(n):
  return(factors(n) == [1,n])
```

In [2]:
```
def prime(n):
  return(len(factors(n)) == 2)
```

In [4]:
```
primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 6
7, 71, 73, 79, 83, 89, 97]
```

Jupyter notebook

- A sequence of cells
  - Like a one dimensional spreadsheet
- Cells hold code or text
  - Markdown notation for formatting
  - `https://www.markdownguide.org/`

jupyter PDSA Week 1 Lecture 1 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Compute primes from 1 to 100

In [1]:
```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)
```

In [3]:
```
def prime(n):
  return(factors(n) == [1,n])
```

In [2]:
```
def prime(n):
  return(len(factors(n)) == 2)
```

In [4]:
```
primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 6
7, 71, 73, 79, 83, 89, 97]
```

Jupyter notebook

- A sequence of cells
  - Like a one dimensional spreadsheet
- Cells hold code or text
  - Markdown notation for formatting
  - `https://www.markdownguide.org/`
- Edit and re-run individual cells to update environment

jupyter PDSA Week 1 Lecture 1 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Compute primes from 1 to 100

In [1]:
```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist.append(i)
  return(factorlist)
```

In [3]:
```
def prime(n):
  return(factors(n) == [1,n])
```

In [2]:
```
def prime(n):
  return(len(factors(n)) == 2)
```

In [4]:
```
primelist = []
for i in range(1,101):
  if (prime(i)):
    primelist.append(i)

print(primelist)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 6
7, 71, 73, 79, 83, 89, 97]
```

Jupyter notebook . . .

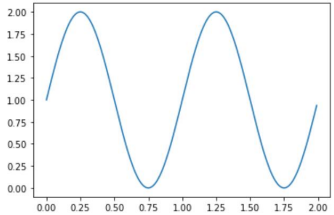
- Supports different kernels
  - Julia, Python, R
  - We will use it only for Python

jupyter PDSA Week 1 Lecture 1 (unsaved changes)

Logout

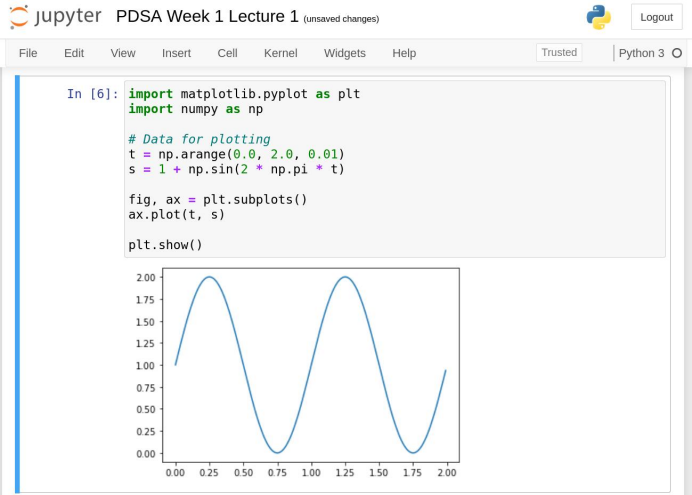
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [6]:
```
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

plt.show()
```



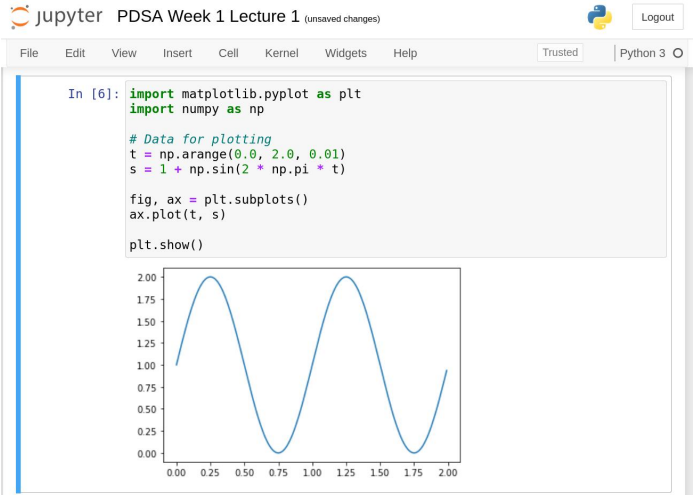
Jupyter notebook . . .

- Supports different kernels
  - Julia, Python, R
  - We will use it only for Python
- Widely used to document and disseminate ML projects
  - Solutions to problems posed on platforms like Kaggle `https://www.kaggle.org`



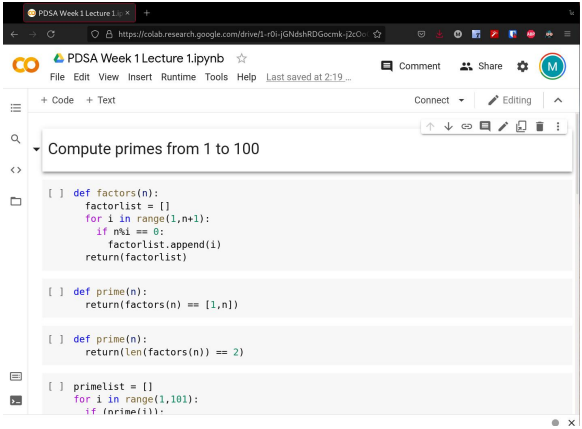
Jupyter notebook . . .

- Supports different kernels
  - Julia, Python, R
  - We will use it only for Python
- Widely used to document and disseminate ML projects
  - Solutions to problems posed on platforms like Kaggle `https://www.kaggle.org`
- ACM Software Systems Award 2017



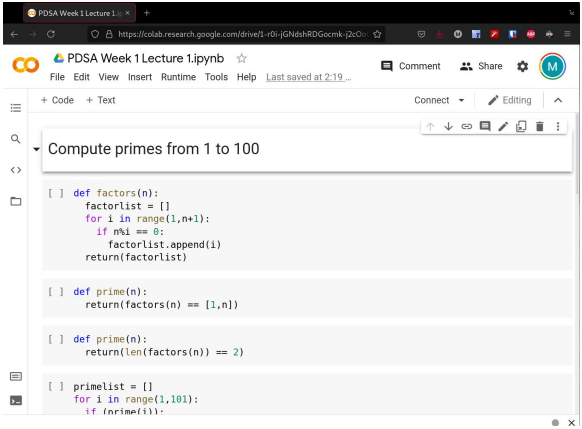
Google Colab

- Google Colaboratory (Colab)
  - `colab.research.google.com`
  - Free to use



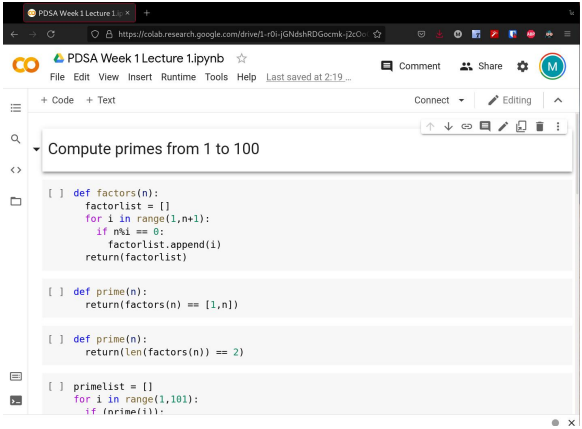
Google Colab

- Google Colaboratory (Colab)
  - `colab.research.google.com`
  - Free to use
- Customized Jupyter notebook



Google Colab

- Google Colaboratory (Colab)
  - `colab.research.google.com`
  - Free to use
- Customized Jupyter notebook
- All standard packages required for ML are preloaded
  - `scikit-learn`, `tensorflow`
  - Access to GPU hardware



Summary

- Jupyter notebook is a convenient interface to develop Python code

Summary

- Jupyter notebook is a convenient interface to develop Python code
- Incrementally update and run

Summary

- Jupyter notebook is a convenient interface to develop Python code
- Incrementally update and run
- Embed documentation using Markdown

Summary

- Jupyter notebook is a convenient interface to develop Python code
- Incrementally update and run
- Embed documentation using Markdown
- Preserve outputs when exporting

Summary

- Jupyter notebook is a convenient interface to develop Python code
- Incrementally update and run
- Embed documentation using Markdown
- Preserve outputs when exporting
- Useful for collaboration, sharing

Summary

- Jupyter notebook is a convenient interface to develop Python code
- Incrementally update and run
- Embed documentation using Markdown
- Preserve outputs when exporting
- Useful for collaboration, sharing
- Google Colab — free to use version configured for ML