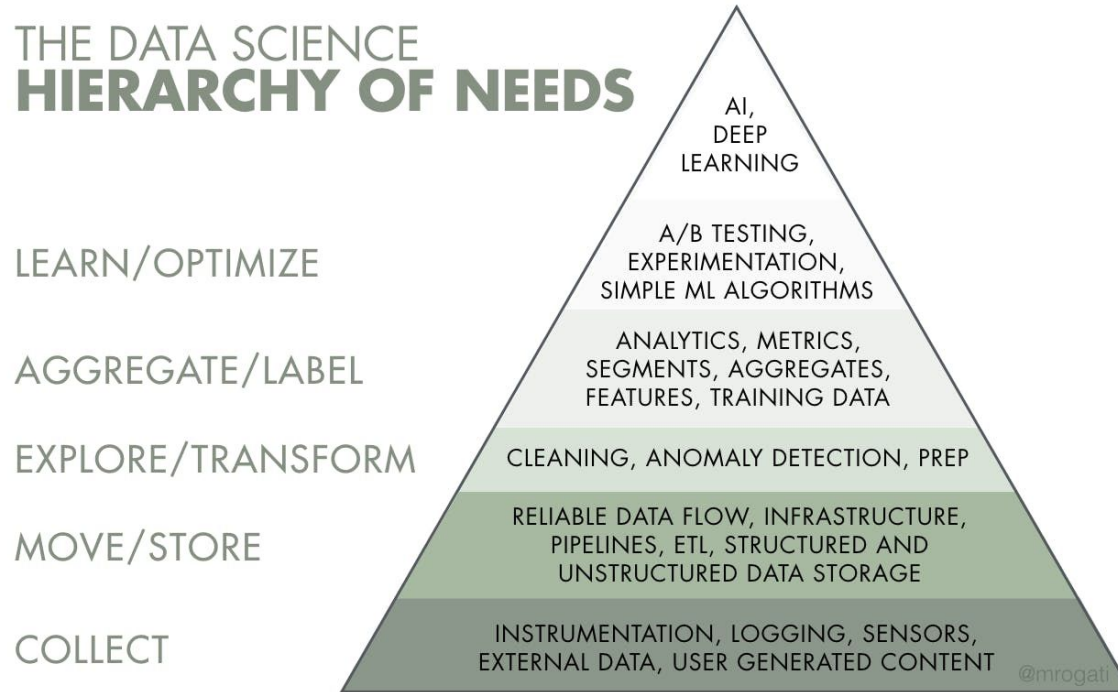


**Innovating on
software
development**

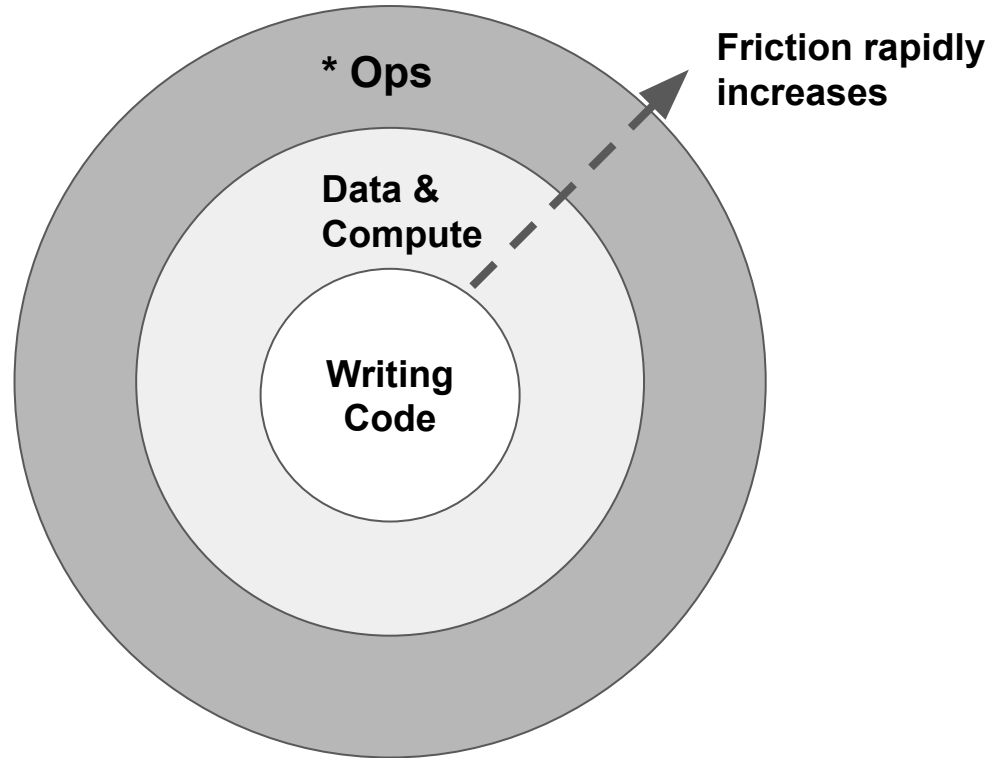
Some observations about
the current state of software
development

We need to write code at all stages

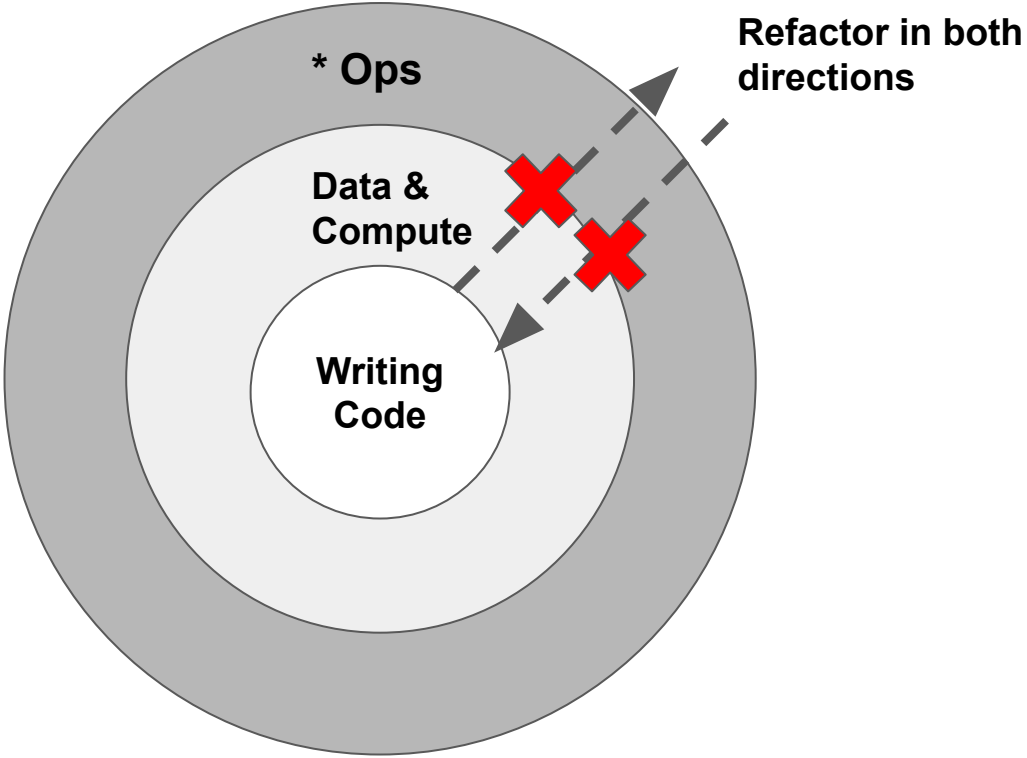


Monica Rogati's Data Science Hierarchy of Needs

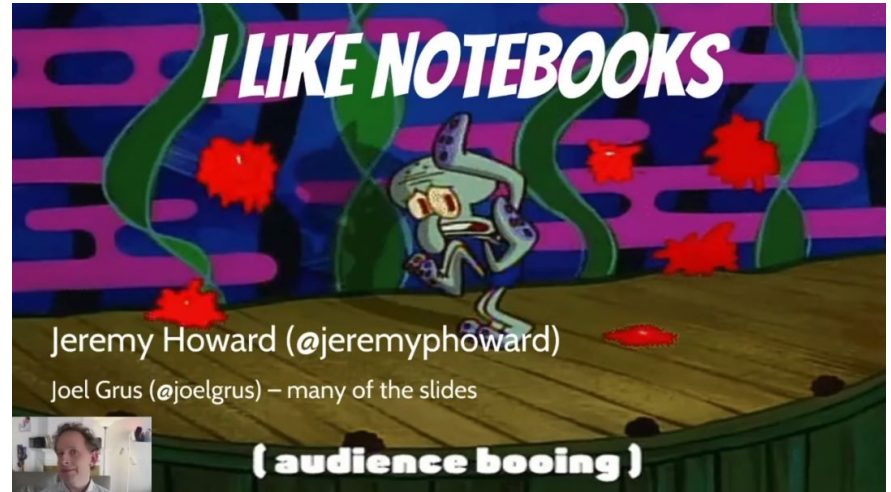
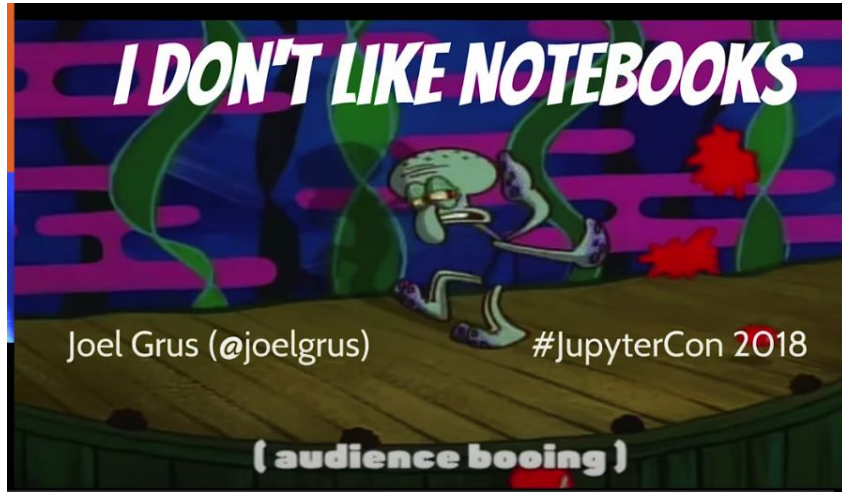
Innovation on tools are focused on the outer loops



We Live in Two Worlds



How We Write Code is Hotly Debated



How We Write Code is Hotly Debated

Operationalizing Machine Learning: An Interview Study

Shreya Shankar*, Rolando Garcia*, Joseph M. Hellerstein, Aditya G. Parameswaran

University of California, Berkeley

{shreyashankar,rogarcia,hellerstein,adityagp}@berkeley.edu

*Co-first authors

completely unaligned.

Strong Opinions on Jupyter Notebooks. Participants described strongly opinionated and different philosophies with respect to how to use Jupyter notebooks in their workflows. Jupyter notebooks were heavily used in development to support high velocity, which we did not find surprising. However, we were surprised that although participants generally acknowledged worse code quality in notebooks, **some participants preferred to use them in production** to minimize the differences between their development and production environments. P6 mentioned that they could debug quickly when locally downloading, executing, and manipulating data from a production notebook run. P18 remarked on the modularization benefits of a migration from a single codebase of scripts to notebooks:

We put each component of the pipeline in a notebook, which has made my life so much easier. Now [when debugging], I can run only one specific component if I want, not the entire pipeline... I don't need to focus on all those other components, and this has also helped with iteration.

On the other hand, some participants strongly disliked the idea of notebooks in production (P10, P15). **P15 even went as far as to philosophically discourage the use of notebooks in the development environment:** "Nobody uses notebooks. Instead, we all work in a shared code base, which is both the training and serving code base and people kick off jobs in the cloud to train models." Similarly, P10 recounted a shift at their company to move any work they wanted to reproduce or deploy out of notebooks:

How We Write Code is Hotly Debated

Operationalizing Machine Learning: An Interview Study

Shreya Shankar*, Rolando Garcia*, Joseph M. Hellerstein, Aditya G. Parameswaran

University of California, Berkeley

{shreyashankar,rogarcia,hellerstein,adityagp}@berkeley.edu

*Co-first authors

The anecdotes on notebooks identified conflicts between competing priorities: (1) Notebooks support high velocity and therefore need to be in development environments, (2) Similar development and production environments prevents new bugs from being introduced, and (3) It's easy to make mistakes with notebooks in production, e.g., running with the wrong inputs; copy-pasting instead of reusing code. Each organization had different rankings of these priorities, ultimately indicating whether or not they used notebooks in production.

Two Axes Of Innovation

Literate Programming

Combines programming and documentation.
Software is written to be read by humans.
Natural language interspersed with code.

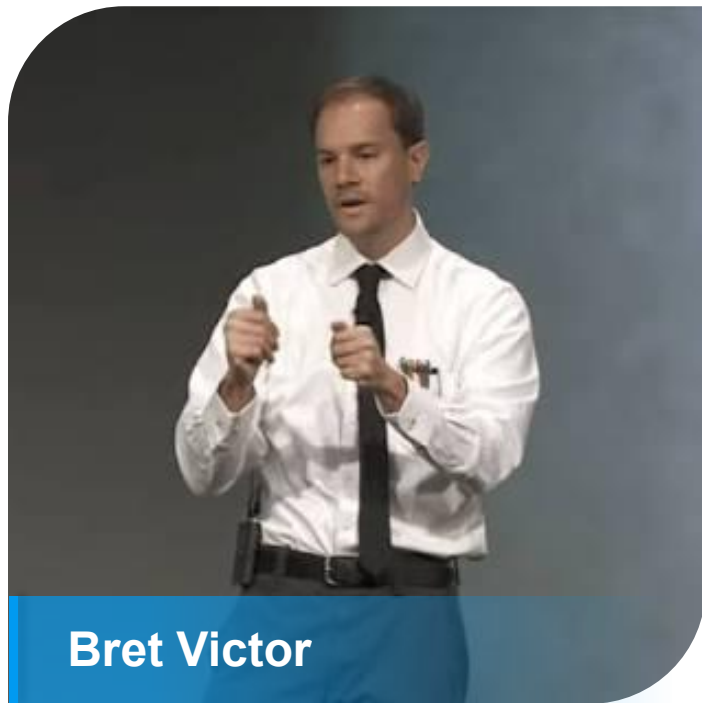
Exploratory Programming

Having an immediate connection and feedback to your code. Interactive, dynamic development

Prior Art



Donald Knuth



Bret Victor

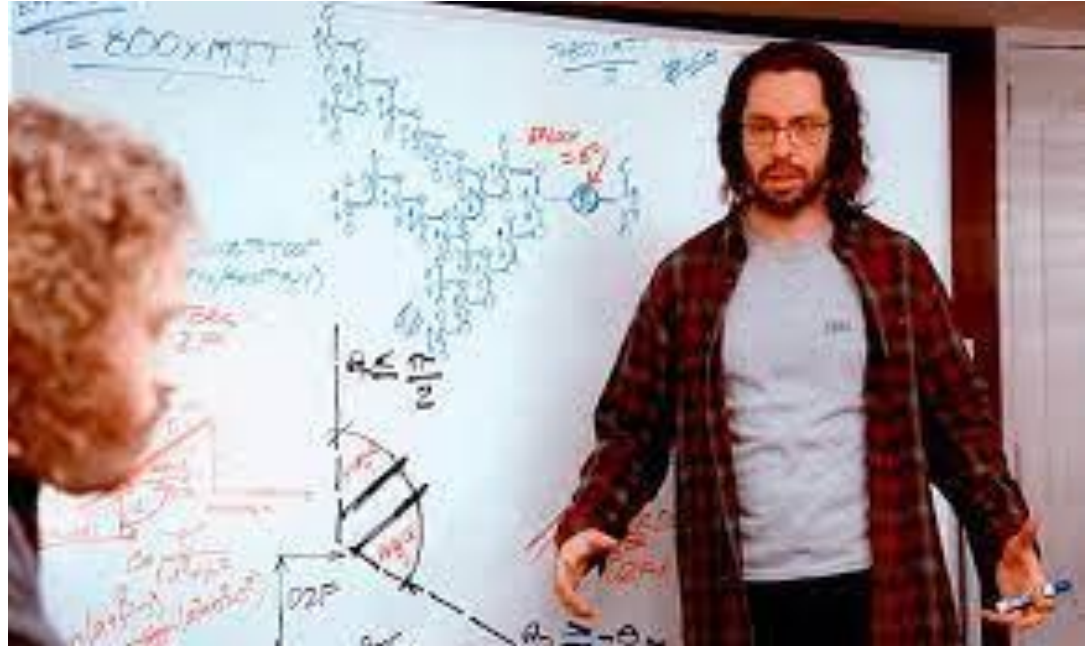
Literate Programming

- Combines programming and documentation.
- Software is written to be read by humans.
- Natural language interspersed with code.

Knuth: “The Yoda Of Silicon Valley”



The basis for an engineering rite of passage



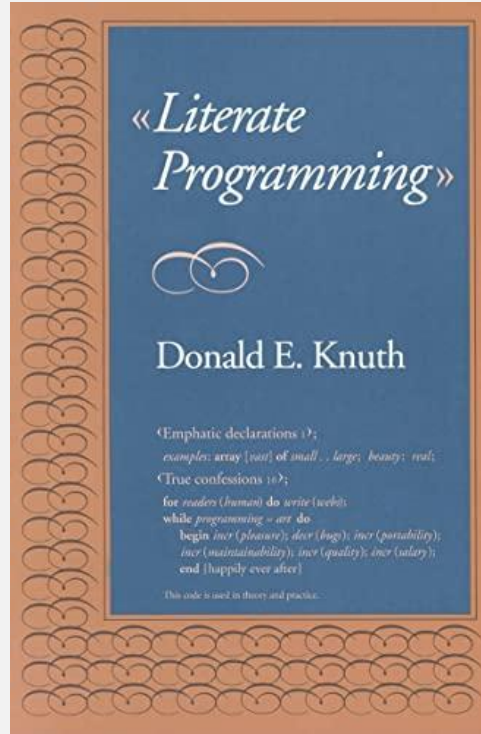
One Of The Most Celebrated Works In Computer Science

If you think you're a really good programmer . . . read [Knuth's] Art of Computer Programming. . . . You should definitely send me a résumé if you can read the whole thing.

—Bill Gates

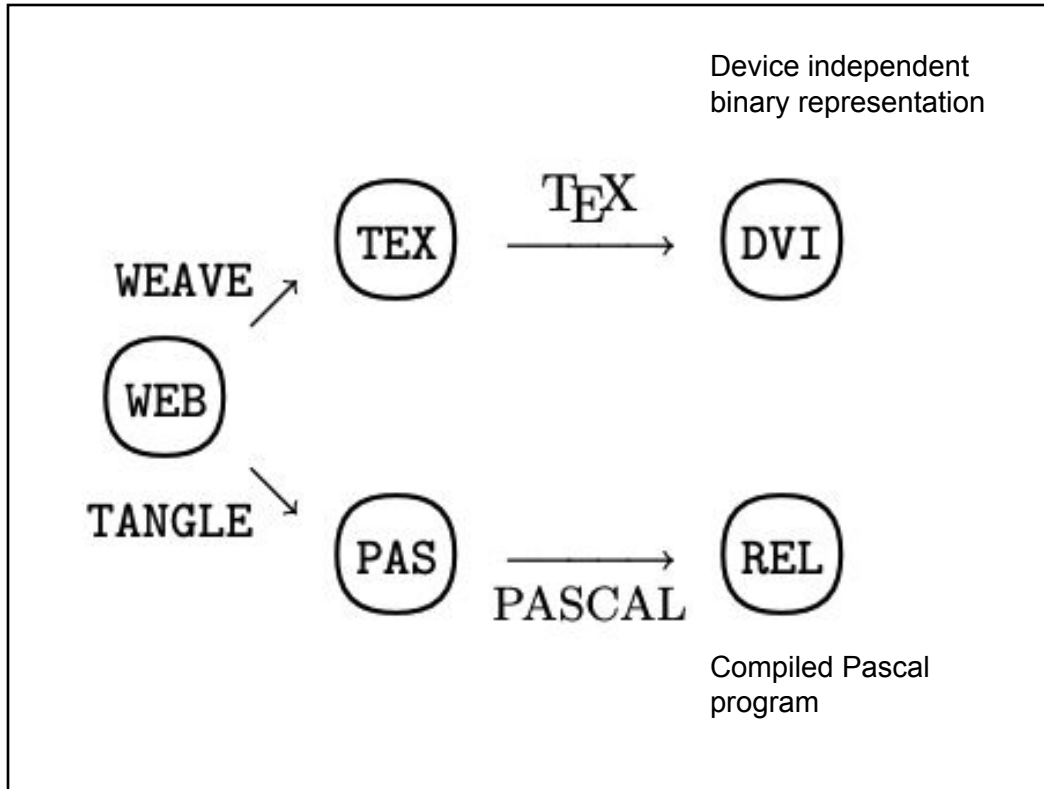
With more than one million copies in print, “The Art of Computer Programming” is the Bible of its field. “Like an actual bible, it is long and comprehensive; no other book is as comprehensive,” said Peter Norvig, a director of research at Google. After 652 pages,

Lesser Known: Coined The Term “Literate Programming”



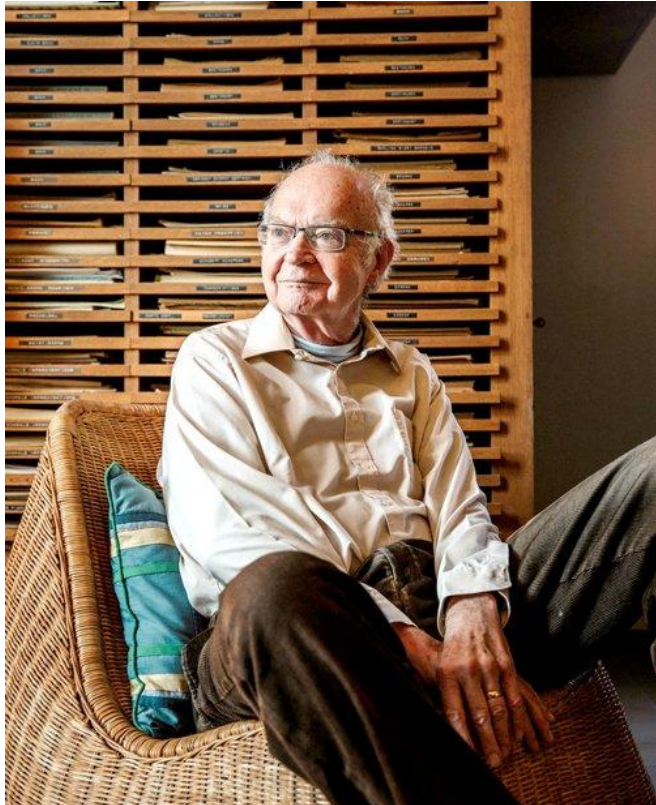
A collection of
work and papers
discussing Literate
Programming.

WEB: A Literate Programming Language



One file generates source code (Pascal) and documentation (Tex) from it

Knuth on Literate Programming

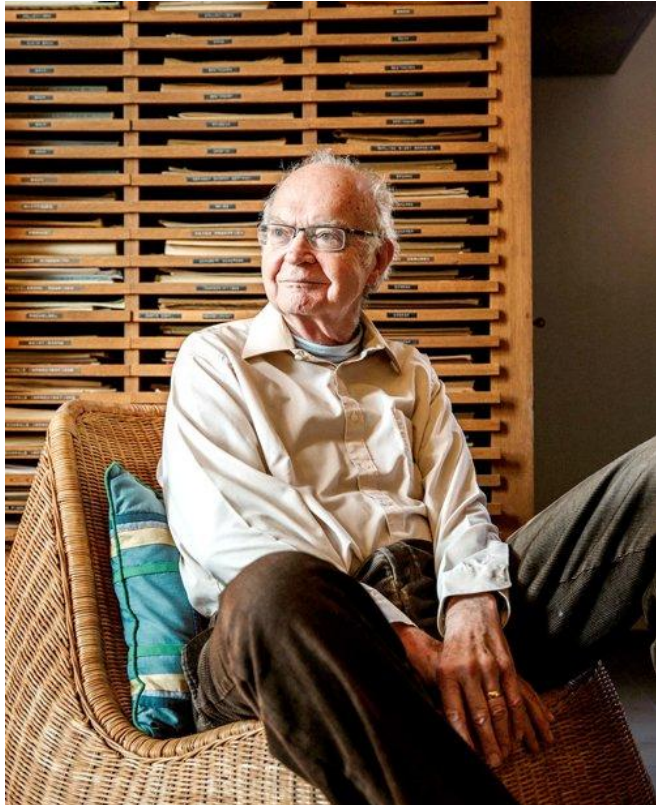


“

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

”

Knuth on Literate Programming

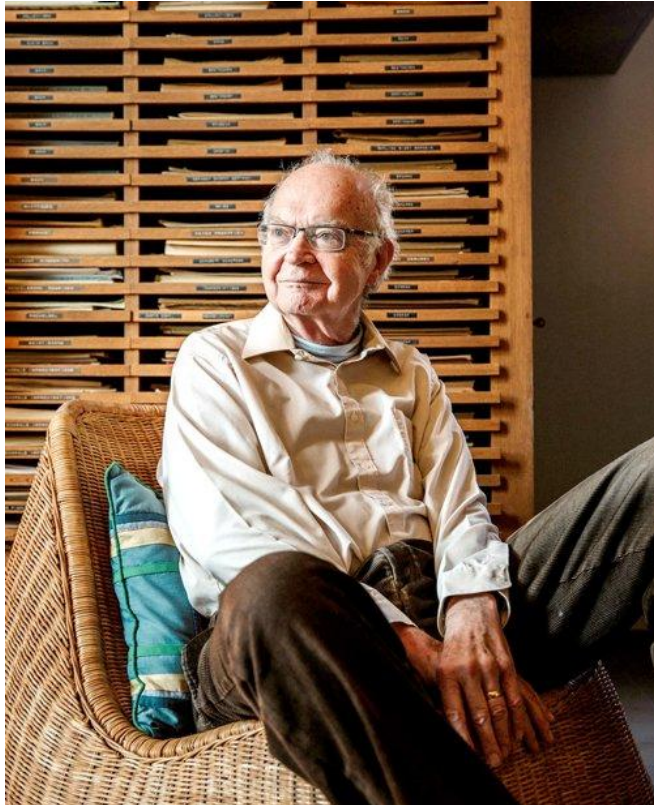


“

By coining the phrase “literate programming,” I am imposing a moral commitment on everyone who hears the term; surely nobody wants to admit writing an illiterate program.

”

Knuth on Literate Programming

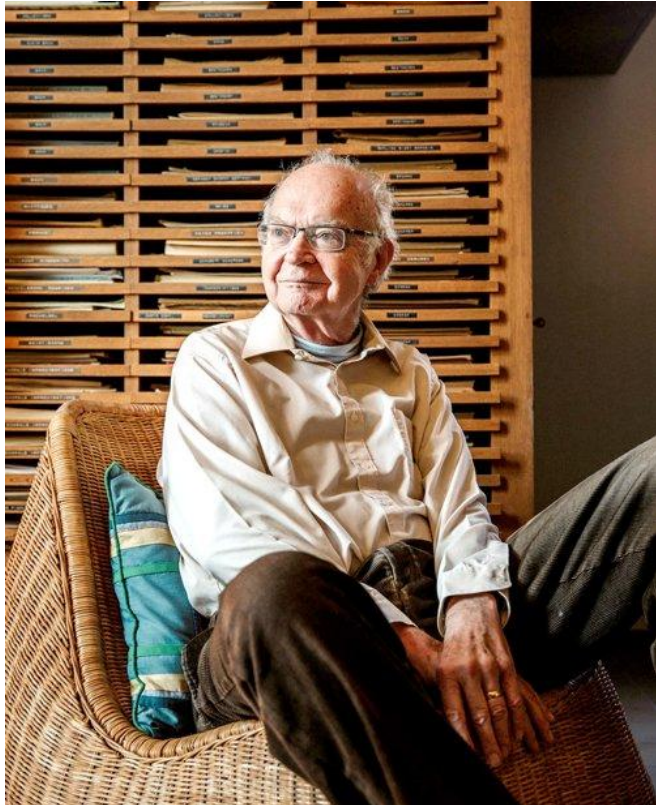


“

All of the major problems associated with computer programming ...are ameliorated when programs and their dialogs with users become more literate.

”

Knuth on Literate Programming



“

Programming is best regarded as the process of creating works of literature, which are meant to be read.

”

Knuth on Literate Programming

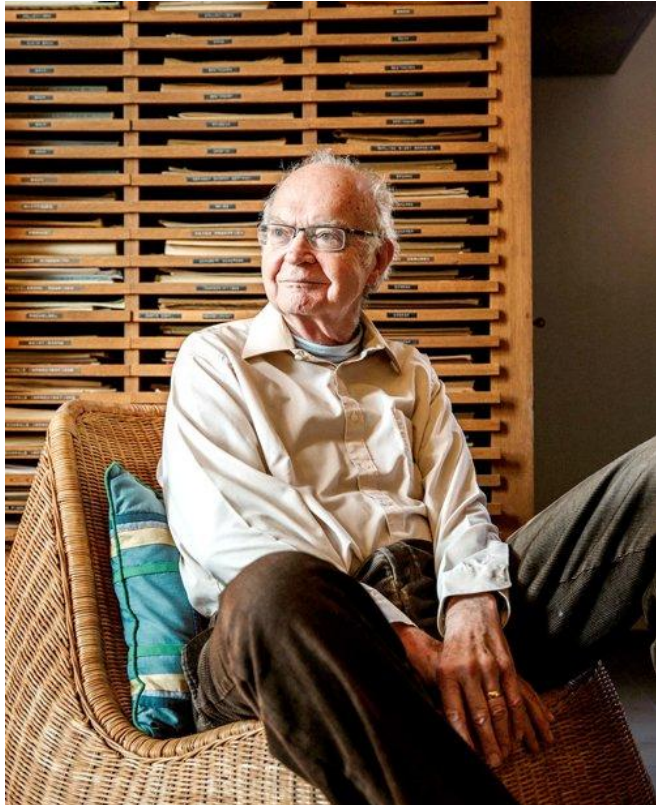


“

In retrospect, the fact that a “literate” program takes much less time to debug is not surprising, because the WEB language encourages a discipline that I was previously unwilling to impose on myself.

”

Knuth on Literate Programming



“

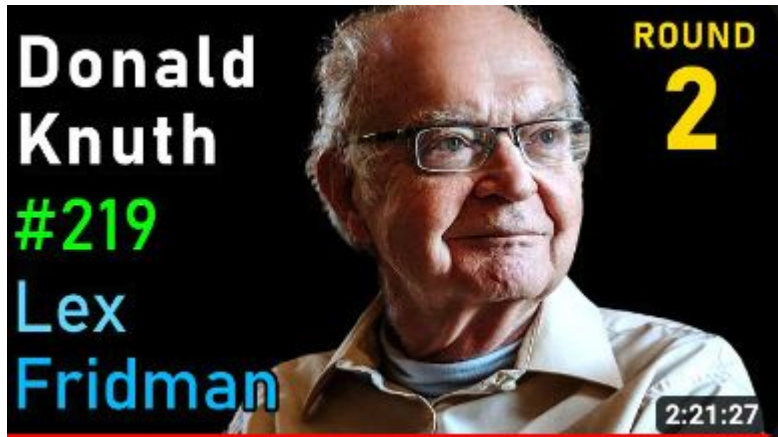
I have found to my surprise that the total time of writing and debugging a WEB program is no greater than the total time of writing and debugging an ALGOL or PASCAL program, even though my WEB programs are much better, and [have] substantially more documentation

”

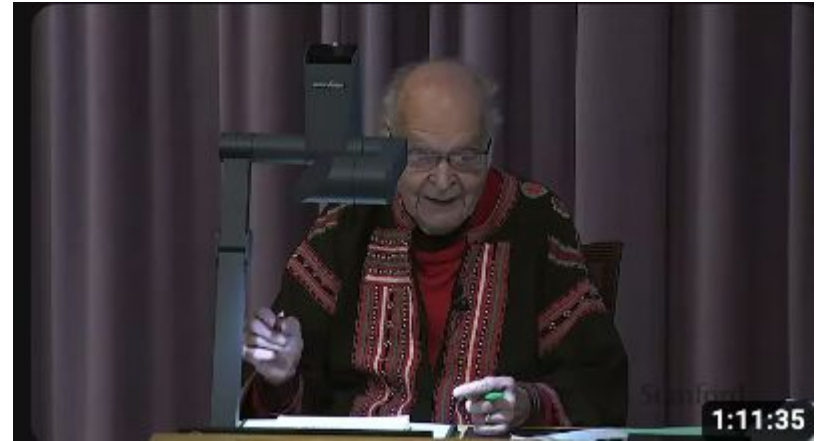
Knuth: We Need To Change The Way We Think About Programming



He hasn't changed his mind



September 2021: Interview



December 2022: Algorithms Lecture

Nod To Literate Programming: R Markdown

The screenshot shows the RStudio interface with a notebook titled "image-graphics.Rmd". The code in the editor includes:

```
16  
17 ## Using Terrain Colors  
18  
19 ```{r}  
20 image(x, y, volcano, col=terrain.colors(100),axes=FALSE)  
21 contour(x, y, volcano, levels=seq(90, 200, by=5), add=TRUE, col="brown")  
22 axis(1, at=x.at)  
23 axis(2, at=y.at)  
24 box()  
25 title(main="Maunga Whau Volcano", sub = "col=terrain.colors(100)", font.main=4)  
26 ```
```

The plot displays a topographic map of Maunga Whau Volcano with contour lines and a color gradient from green to brown. The x-axis is labeled 'x' and ranges from 0 to 800. The y-axis is labeled 'y' and ranges from 0 to 600. Below the plot, the code `col=terrain.colors(100)` is visible.

The right-hand side of the RStudio window shows the Environment pane with a list of functions and the Files pane showing a list of installed packages.

Name	Description	Version
assertthat	Easy pre and post assertions.	0.1
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.60.0-2
biclust	BiCluster Algorithms	1.2.0
bitops	Bitwise Operations	1.0-6
bookdown	Authoring Books with R Markdown	0.0.71
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-18
brew	Templating Framework for Report Generation	1.0-6
broom	Convert Statistical Analysis Objects into Tidy Data Frames	0.4.0
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
class	Functions for Classification	7.3-14
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.4
codetools	Code Analysis Tools for R	0.2-14

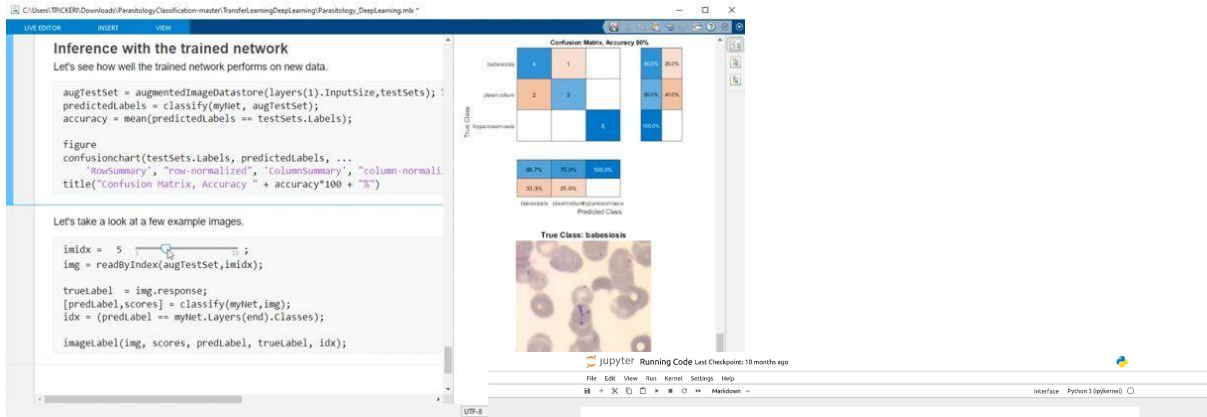
A collage of R Markdown documents and plots. The top document is titled "Diversity gradient" and features a map of the United States with a red dot in the Northeast. Below it is a document titled "More Examples" which contains the following text:

The rest of this document consists of a few test cases to make sure everything still works well in slightly more complicated scenarios. First we generate two plots in one figure environment with the chunk option `fig.show = 'hold'`:

```
p <- ggplot(etcars2, aes(hp, mpg, color = am)) +  
  geom_point()  
p  
p + geom_smooth()
```

The bottom plot is a scatter plot of mpg vs hp, with a legend for 'am' (automatic and manual). The plot shows a positive correlation between horsepower and miles per gallon, with manual transmission cars generally having higher horsepower and lower mpg.

Nod To Literate Programming: Notebooks



The screenshot shows a Jupyter Notebook window with the following content:

```
Inference with the trained network
Let's see how well the trained network performs on new data.

augTestSet = augmentedImageDataset(layers(1).InputsSize, testSets);
predictedLabels = classify(myNet, augTestSet);
accuracy = mean(predictedLabels == testSets.Labels);

figure
confusionchart(testSets.Labels, predictedLabels, ...
    'rowsSummary', 'row-normalized', 'columnSummary', 'column-normalized',
    title('Confusion Matrix, Accuracy ' + accuracy*100 + '%'))

Let's take a look at a few example images.

indx = 5;
img = readbyIndex(augTestSet, indx);
trueLabel = img.response;
[predLabel, scores] = classify(myNet, img);
idx = (predLabel == myNet.Layers(end).Classes);
imageLabel(img, scores, predLabel, trueLabel, idx);
```

The notebook displays a confusion matrix with the following data:

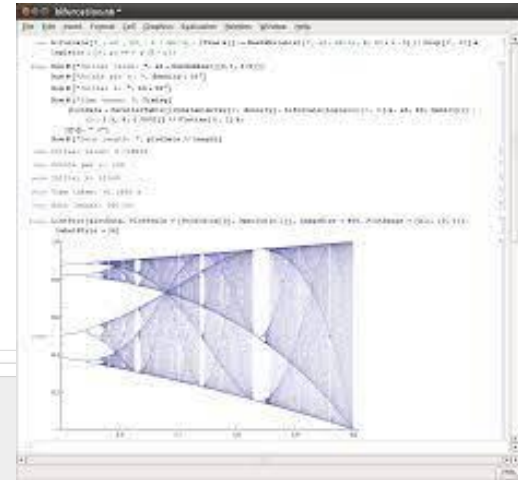
True Class \ Predicted Class	babesiosis	leishmaniasis	trypanosomiasis
babesiosis	1	1	0
leishmaniasis	2	3	0
trypanosomiasis	0	0	5

Summary statistics: Accuracy: 80%. Confusion Matrix: 94.7% (True Positives), 75.0% (True Negatives), 100.0% (True Negatives), 33.3% (False Positives), 20.0% (False Negatives).

True Class: babesiosis

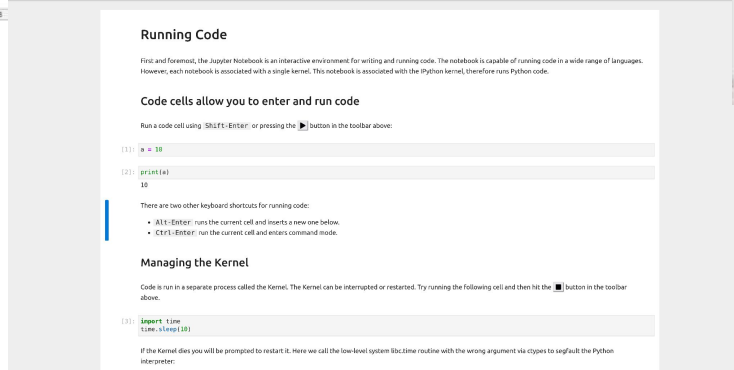
The interface includes a toolbar with options like 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', 'Help', and a status bar indicating 'interface Python 3 (ipykernel)'.

Matlab



The screenshot shows the Mathematica software interface with a 3D plot of a complex, multi-faceted geometric shape. The plot is rendered in a light blue color and is set against a dark background. The interface includes a menu bar at the top with options like 'File', 'Edit', 'Format', 'Cell', 'Graphics', 'Application', 'System', 'Window', and 'Help'. The plot is displayed in a 3D view, showing its intricate structure and shading.

Mathematica



The screenshot shows a Jupyter Notebook page with the following content:

Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using **SHIFT+ENTER** or pressing the **Run** button in the toolbar above:

```
(1) | a = 10
(2) | print(a)
10
```

There are two other keyboard shortcuts for running code:

- **ALT+ENTER**: runs the current cell and inserts a new one below.
- **CTRL+ENTER**: run the current cell and enters command mode.

Managing the Kernel

Code is run in a separate process called the Kernel. The Kernel can be interrupted or restarted. Try running the following cell and then hit the **Restart** button in the toolbar above.

```
(3) | import time
| time.sleep(10)
```

If the Kernel dies you will be prompted to restart it. Here we call the low-level system `time.sleep` routine with the wrong argument via `ctypes` to segfault the Python interpreter:

Jupyter

Writing prose + “production” code is painful

```
sklearn > metrics > sklearn.py > anycode > auc
46
47 def auc(x, y):
48     """Compute Area Under the Curve (AUC) using the trapezoidal rule.
49
50     This is a general function, given points on a curve. For computing the
51     area under the ROC-curve, see :func:`roc_auc_score`. For an alternative
52     way to summarize a precision-recall curve, see
53     :func:`average_precision_score`.
54
55     Parameters
56     -----
57     x : ndarray of shape (n,)
58         x coordinates. These must be either monotonic increasing or monotonic
59         decreasing.
60     y : ndarray of shape, (n,)
61         y coordinates.
62
63     Returns
64     -----
65     auc : float
66
67     See Also
68     -----
69     roc_auc_score : Compute the area under the ROC curve.
70     average_precision_score : Compute average precision from prediction scores
71     precision_recall_curve : Compute precision-recall pairs for different
72     probability thresholds.
73
74     Examples
75     -----
76     >>> import numpy as np
77     >>> from sklearn import metrics
78     >>> y = np.array([1, 1, 2, 2])
79     >>> pred = np.array([0.1, 0.4, 0.35, 0.8])
80     >>> fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=2)
81     >>> metrics.auc(fpr, tpr)
82     0.75
83     """
84     check_consistent_length(x, y)
85     x = column_or_1d(x)
86     y = column_or_1d(y)
87
88     if x.shape[0] < 2:
89         raise ValueError(
90             "At least 2 points are needed to compute area under curve, but x is
```

sklearn.metrics.auc

sklearn.metrics.auc(x, y)

[source]

Compute Area Under the Curve (AUC) using the trapezoidal rule.

This is a general function, given points on a curve. For computing the area under the ROC-curve, see [roc_auc_score](#). For an alternative way to summarize a precision-recall curve, see [average_precision_score](#).

Parameters:	x : <i>ndarray of shape (n,)</i> x coordinates. These must be either monotonic increasing or monotonic decreasing.
	y : <i>ndarray of shape, (n,)</i> y coordinates.
Returns:	auc : <i>float</i>

See also:

roc_auc_score

Compute the area under the ROC curve.

average_precision_score


Compute average precision from prediction scores.

precision_recall_curve

Compute precision-recall pairs for different probability thresholds.

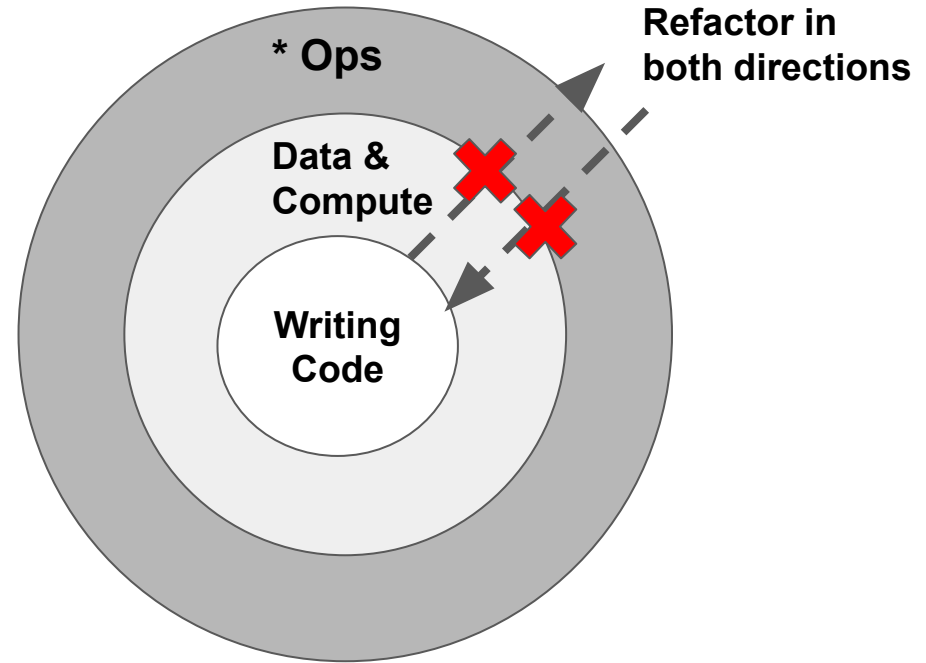
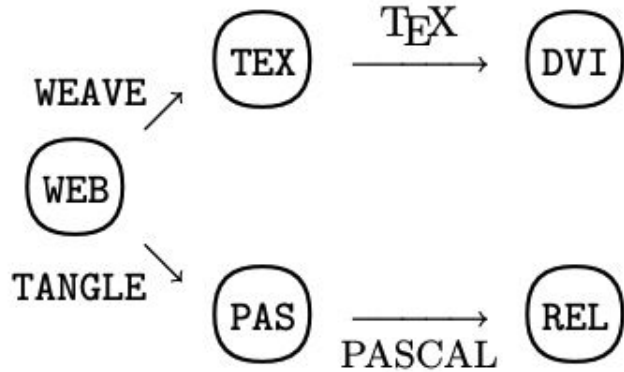
Examples

```
>>> import numpy as np
>>> from sklearn import metrics
>>> y = np.array([1, 1, 2, 2])
>>> pred = np.array([0.1, 0.4, 0.35, 0.8])
>>> fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=2)
>>> metrics.auc(fpr, tpr)
0.75
```



This looks like a notebook, even though its not.

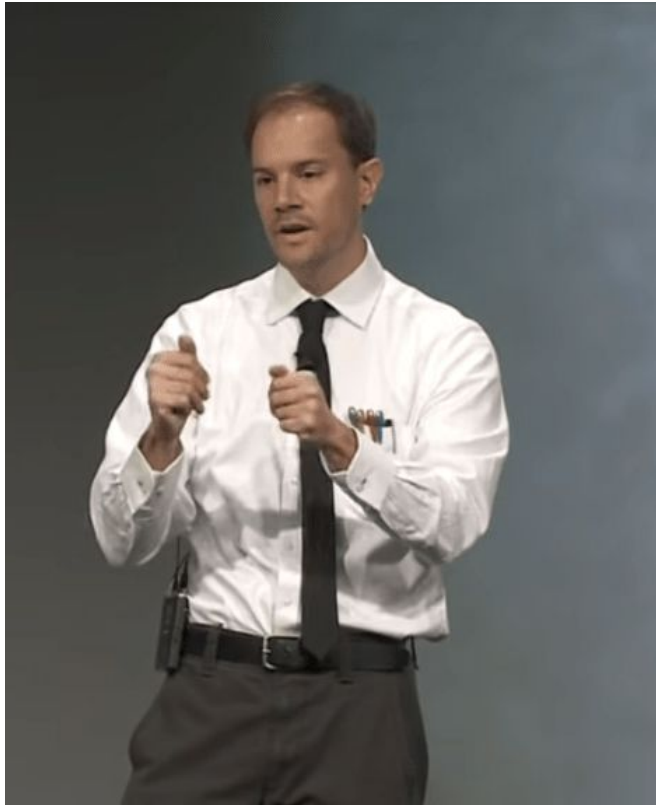
Literate Programming vs Notebooks: Still A Gap



Exploratory Programming

- Having an immediate connection and feedback to your code.
- Interactive, dynamic development

Bret Victor: Champion of Exploratory Programming

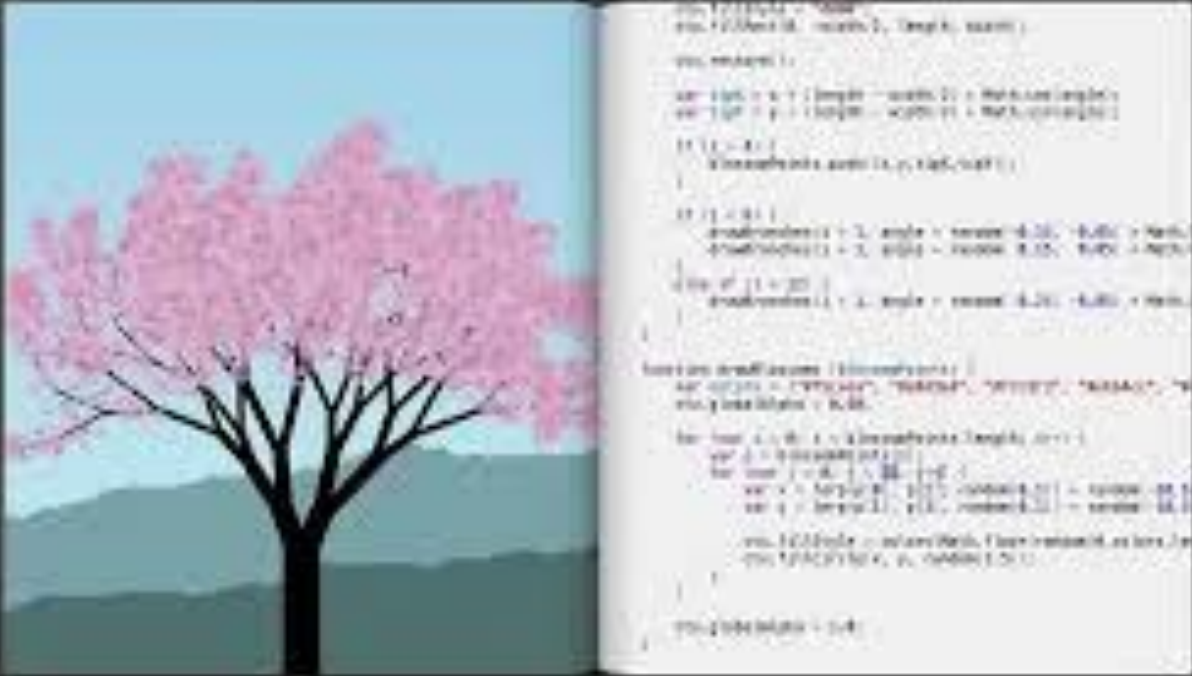


“

I wanna make a new MatLab or a cross between MatLab and Garageband.

”

Bret Victor: Inventing On Principle



What If We Combined Literate
& Exploratory Programming?

nbdev.fast.ai

Create delightful software with Jupyter Notebooks

Write, test, document, and distribute software packages and technical articles — all in one place, your notebook.

Get started

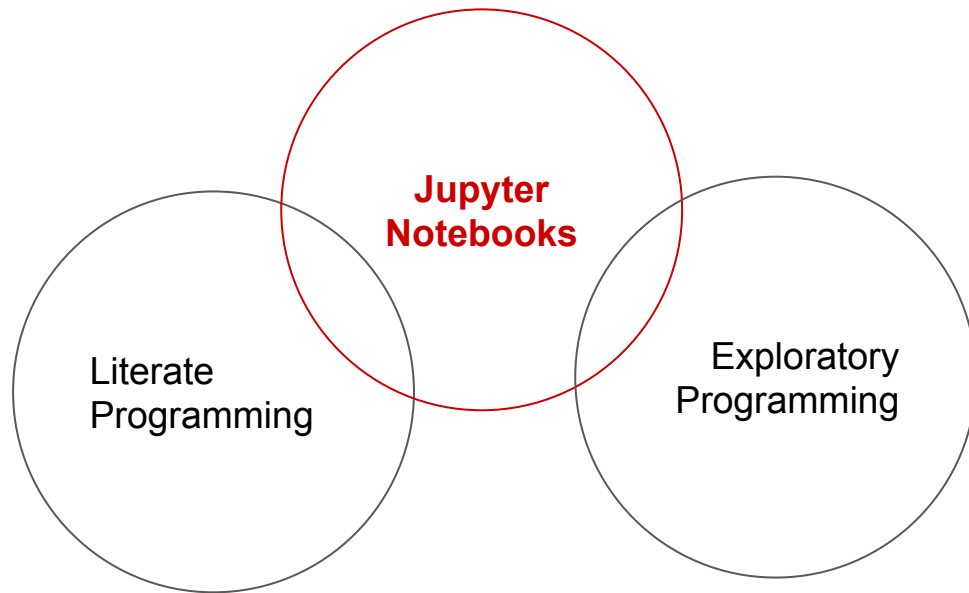
The central Jupyter notebook displays the following content:

```
A Deck of cards is a collection of Card objects.  
deck = Deck()  
deck  
deck  
There are 52 cards in a deck.  
test_eq(len(deck), 52)  
#import  
from IPython  
def pop(self):  
    """Returns and removes card from the deck"""  
    return self.deck.pop(index)  
deck.pop()  
deck  
There are 51 cards left in the deck now.  
test_eq(len(deck), 51)
```

The surrounding panels show:

- GitHub:** A repository named `pages-build-deployment` with a `build` workflow and a `report-build-status` action.
- ANACONDA.ORG:** A repository for `fastai/packages/cards-deck 0.0.3` with a table of versions and a `pip install cards-deck` command.
- quarto:** A notebook `Deck.qmd` with a table showing the output of `deck.pop()`.
- pypi:** A package page for `cards-deck 0.0.3` with a `pip install cards-deck` command and a project description.

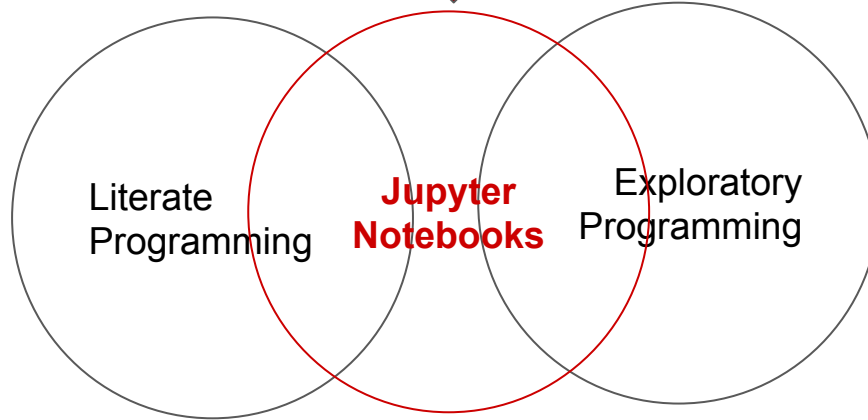
nbdev



nbdev



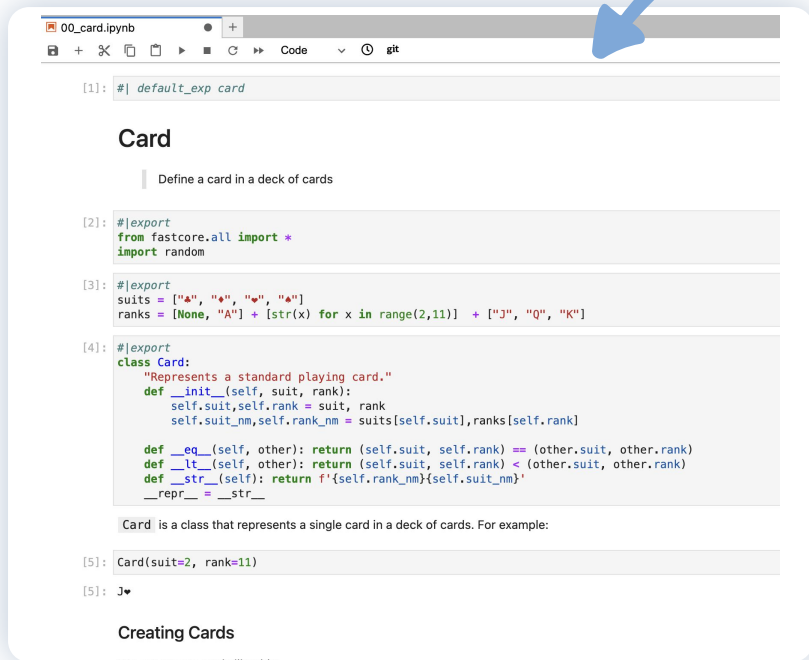
Nbdev extends jupyter notebooks to do more



What is nbdev?

1. Generate Python Modules

2-way sync



```
[1]: #| default_exp card

Card
    | Define a card in a deck of cards

[2]: #|export
from fastcore.all import *
import random

[3]: #|export
suits = ["♠", "♥", "♦", "♣"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]

[4]: #|export
class Card:
    """Represents a standard playing card."""
    def __init__(self, suit, rank):
        self.suit, self.rank = suit, rank
        self.suit_nm, self.rank_nm = suits[self.suit], ranks[self.rank]

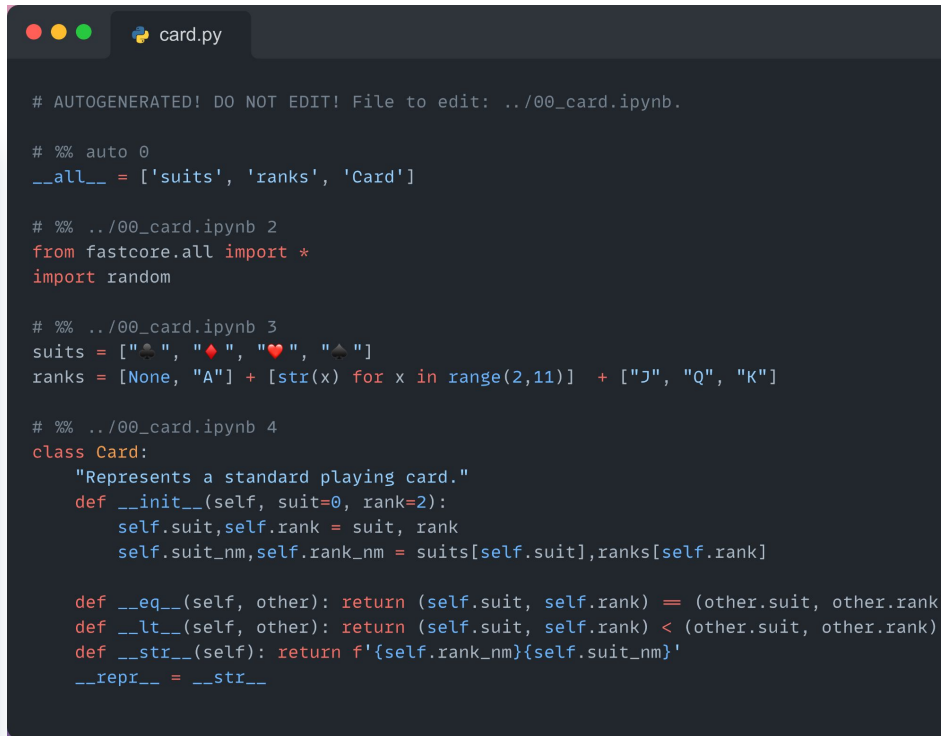
    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__

Card is a class that represents a single card in a deck of cards. For example:

[5]: Card(suit=2, rank=11)

[5]: J♥

Creating Cards
    We can create cards like this:
```



```
card.py

# AUTOGENERATED! DO NOT EDIT! File to edit: ../00_card.ipynb.

# %% auto 0
__all__ = ['suits', 'ranks', 'Card']

# %% ../00_card.ipynb 2
from fastcore.all import *
import random

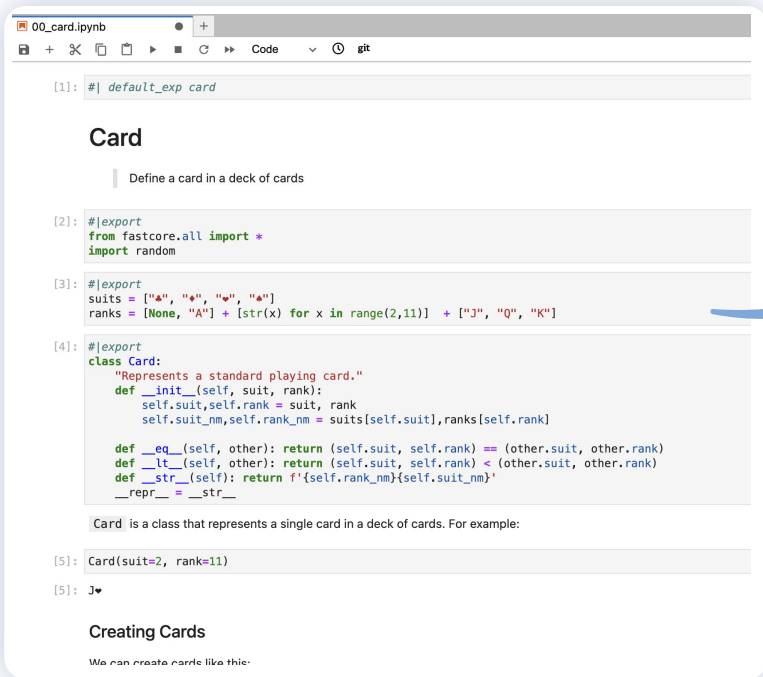
# %% ../00_card.ipynb 3
suits = ["♠", "♥", "♦", "♣"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]

# %% ../00_card.ipynb 4
class Card:
    """Represents a standard playing card."""
    def __init__(self, suit=0, rank=2):
        self.suit, self.rank = suit, rank
        self.suit_nm, self.rank_nm = suits[self.suit], ranks[self.rank]

    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__
```

Use your favorite IDE for nav & edits

2. Develop Modules From Notebooks



```
[1]: #| default_exp card
```

Card

Define a card in a deck of cards

```
[2]: #|export
from fastcore.all import *
import random
```

```
[3]: #|export
suits = ["♠", "♥", "♦", "♣"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]
```

```
[4]: #|export
class Card:
    """Represents a standard playing card."""
    def __init__(self, suit, rank):
        self.suit, self.rank = suit, rank
        self.suit_nm, self.rank_nm = suits[self.suit], ranks[self.rank]

    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__
```

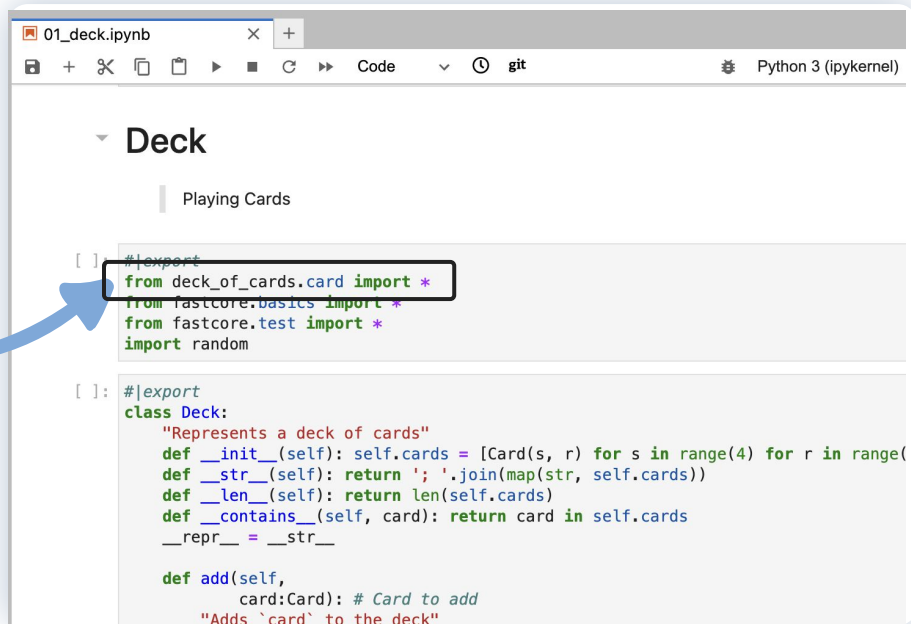
Card is a class that represents a single card in a deck of cards. For example:

```
[5]: Card(suit=2, rank=11)
```

```
[5]: J♥
```

Creating Cards

We can create cards like this:



```
01_deck.ipynb
```

Deck

Playing Cards

```
[ ]: #|export
from deck_of_cards.card import *
from fastcore.basics import *
from fastcore.test import *
import random
```

```
[ ]: #|export
class Deck:
    """Represents a deck of cards"""
    def __init__(self): self.cards = [Card(s, r) for s in range(4) for r in range(
    def __str__(self): return '; '.join(map(str, self.cards))
    def __len__(self): return len(self.cards)
    def __contains__(self, card): return card in self.cards
    __repr__ = __str__

    def add(self,
            card:Card): # Card to add
        """Adds `card` to the deck"""
```

3. Documentation w/Quarto

```
00_card.ipynb
[1]: #| default_exp card

Card
  Define a card in a deck of cards

[2]: #|export
from fastcore.all import *
import random

[3]: #|export
suits = ["A", "S", "H", "D"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]

[4]: #|export
class Card:
    """Represents a standard playing card."""
    def __init__(self, suit, rank):
        self.suit,self.rank = suit, rank
        self.suit_nm,self.rank_nm = suits[self.suit],ranks[self.rank]

    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__

Card is a class that represents a single card in a deck of cards. For example:

[5]: Card(suit=2, rank=11)

[5]: J♥

Creating Cards
We can create cards like this:
```



deck_of_cards

Deck of Cards
Card
Deck

Card

Define a card in a deck of cards

On this page
Creating Cards
Comparing Cards
Report an issue

Card

Card (suit=0, rank=2)

Represents a standard playing card.

Card is a class that represents a single card in a deck of cards. For example:

```
Card(suit=2, rank=11)
```

J♥

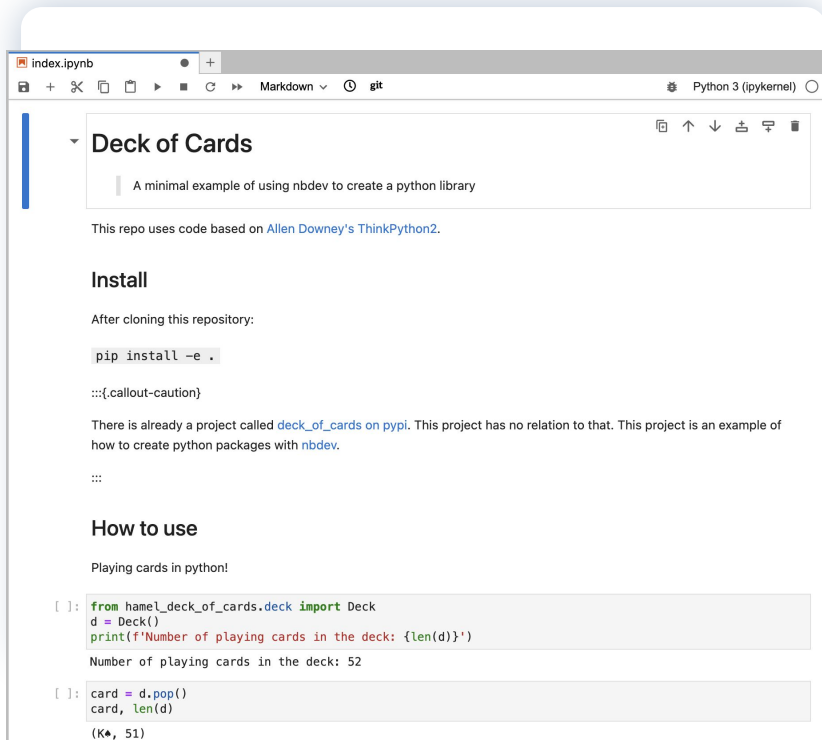
Creating Cards

We can create cards like this:

```
c = Card(suit=1, rank=3)
c
```

3 of Diamonds

4. README with Notebooks, Too



The screenshot shows a Jupyter Notebook window titled 'index.ipynb' with a Python 3 (ipykernel) environment. The notebook content is a README for 'Deck of Cards', which includes a description, installation instructions, a warning about a conflicting project, and a code cell demonstrating how to use the Deck class.

Deck of Cards

A minimal example of using nbdev to create a python library

This repo uses code based on [Allen Downey's ThinkPython2](#).

Install

After cloning this repository:

```
pip install -e .
```

!!!callout-caution

There is already a project called [deck_of_cards on pypi](#). This project has no relation to that. This project is an example of how to create python packages with [nbdev](#).

How to use

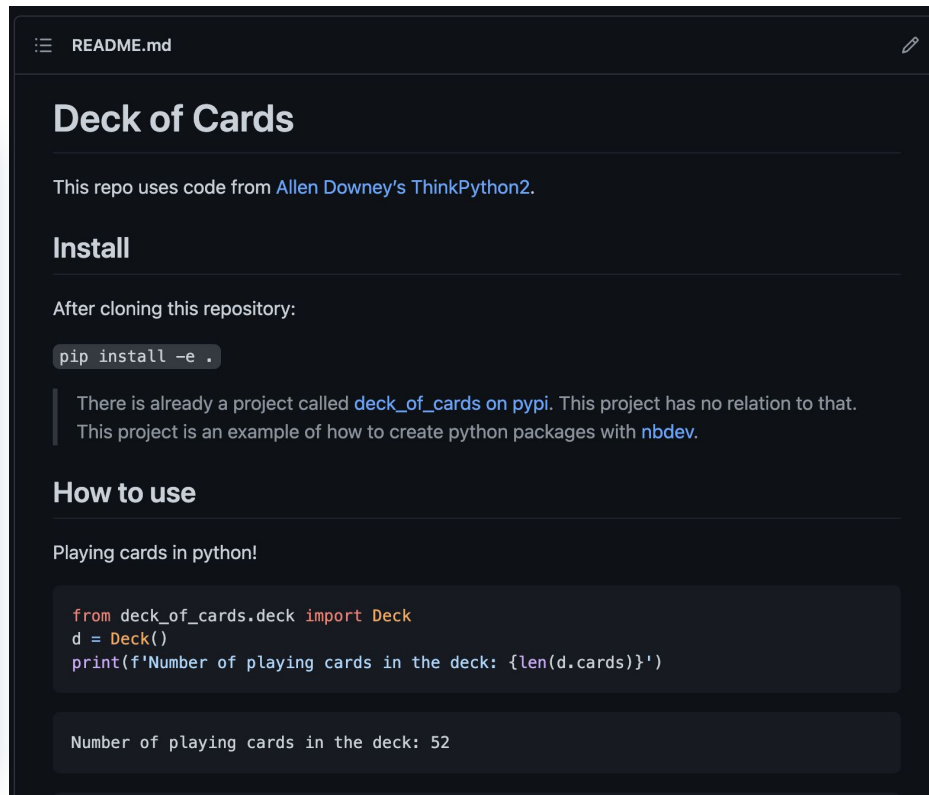
Playing cards in python!

```
[ ]: from hame_deck_of_cards.deck import Deck
d = Deck()
print(f'Number of playing cards in the deck: {len(d)}')
```

Number of playing cards in the deck: 52

```
[ ]: card = d.pop()
card, len(d)
```

(K, 51)



The screenshot shows a dark-themed editor window titled 'README.md'. The content is identical to the Jupyter Notebook screenshot, but with a code cell that shows the output of the Deck class: 'Number of playing cards in the deck: 52'. A blue arrow points from the bottom of the Jupyter Notebook screenshot to this output line.

Deck of Cards

This repo uses code from [Allen Downey's ThinkPython2](#).

Install

After cloning this repository:

```
pip install -e .
```

There is already a project called [deck_of_cards on pypi](#). This project has no relation to that. This project is an example of how to create python packages with [nbdev](#).

How to use

Playing cards in python!

```
from deck_of_cards.deck import Deck
d = Deck()
print(f'Number of playing cards in the deck: {len(d.cards)}')
```

Number of playing cards in the deck: 52

5. Tests: CI & Local

```
00_card.ipynb
[1]: #| default_exp card

Card
Define a card in a deck of cards

[2]: #|export
from fastcore.all import *
import random

[3]: #|export
suits = ["A", "♠", "♥", "♣"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]

[4]: #|export
class Card:
    "Represents a standard playing card."
    def __init__(self, suit, rank):
        self.suit,self.rank = suit, rank
        self.suit_nm,self.rank_nm = suits[self.suit],ranks[self.rank]

    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__

Card is a class that represents a single card in a deck of cards. For example:

[5]: Card(suit=2, rank=11)
[5]: J♣

Creating Cards
We can create cards like this-
```

Test Notebooks

```
1 ▶ Run nbdev_test
7 WARNING:root:AssertionError in
/home/runner/work/deck_of_cards/deck_of_cards/00_card.ipynb:
8 =====
9
10 While Executing Cell #12:
11 -----
12 AssertionError Traceback (most recent call last)
13 Input In [1], in <cell line: 1>()
14 ----> 1 test_eq(str(c), '3 of Diamonds')
15
16 File /opt/hostedtoolcache/Python/3.10.5/x64/lib/python3.10/site-
packages/fastcore/test.py:36, in test_eq(a, b)
17     34 def test_eq(a, b):
18         35 "test" that `a==b`
19 ----> 36 test(a,b,equals, '==')
20
21 File /opt/hostedtoolcache/Python/3.10.5/x64/lib/python3.10/site-
packages/fastcore/test.py:24, in assert_that(a, b, cmp, cname)
22     24 "assert" that `a` is equal to `b`
23     25 if cname is None:
24 ----> 26 assert cmp(a, b)
25
26 AssertionError: ==:
27 3♣
28 3 of Diamonds
```

test

- Set up job
- Run actions/checkout@v3
- Run actions/setup-python@v3
- Install Dependencies
- Check if all notebooks are cleaned
- Test Notebooks
- Post Run actions/setup-python@v3
- Post Run actions/checkout@v3
- Complete job

```
(base) ~/deck_of_cards % nbdev_test
WARNING:root:AssertionError in /Users/home1/github/deck_of_cards/00_card.ipynb:
While Executing Cell #12:
-----
AssertionError Traceback (most recent call last)
<ipython-input-1-fba952eda60d> in <module>
----> 1 test_eq(str(c), '3 of Diamonds')

~/github/fastcore/fastcore/test.py in test_eq(a, b)
    34 def test_eq(a, b):
    35     "test" that `a==b`
----> 36     test(a,b,equals, '==')
    37
    38 # Cell

~/github/fastcore/fastcore/test.py in test(a, b, cmp, cname)
    24     "assert" that `cmp(a,b)`; display inputs and `cname or cmp.__name__` if it fails"
    25     if cname is None: cname=cmp.__name__
----> 26     assert cmp(a,b), f'({cname}):{n(a)}\n{b}'
    27
    28 # Cell

AssertionError: ==:
3♣
3 of Diamonds
```

6. Python Packaging

```
00_card.ipynb
[1]: #| default_exp card

Card
    Define a card in a deck of cards

[2]: #|export
from fastcore.all import *
import random

[3]: #|export
suits = ["A", "S", "H", "D"]
ranks = [None, "A"] + [str(x) for x in range(2,11)] + ["J", "Q", "K"]

[4]: #|export
class Card:
    """Represents a standard playing card."""
    def __init__(self, suit, rank):
        self.suit,self.rank = suit, rank
        self.suit_nm,self.rank_nm = suits[self.suit],ranks[self.rank]

    def __eq__(self, other): return (self.suit, self.rank) == (other.suit, other.rank)
    def __lt__(self, other): return (self.suit, self.rank) < (other.suit, other.rank)
    def __str__(self): return f'{self.rank_nm}{self.suit_nm}'
    __repr__ = __str__

Card is a class that represents a single card in a deck of cards. For example:

[5]: Card(suit=2, rank=11)

[5]: J♦

Creating Cards
We can create cards like this-
```



Search projects

Help

hamel-deck-of-cards 0.0.1

`pip install hamel-deck-of-cards`

An example repository for nbdev

Navigation

- [Project description](#)
- [Release history](#)
- [Download files](#)

Project links

- [Homepage](#)

Project description

Deck of Cards

This repo uses code from [Allen Downey's ThinkPython2](#).

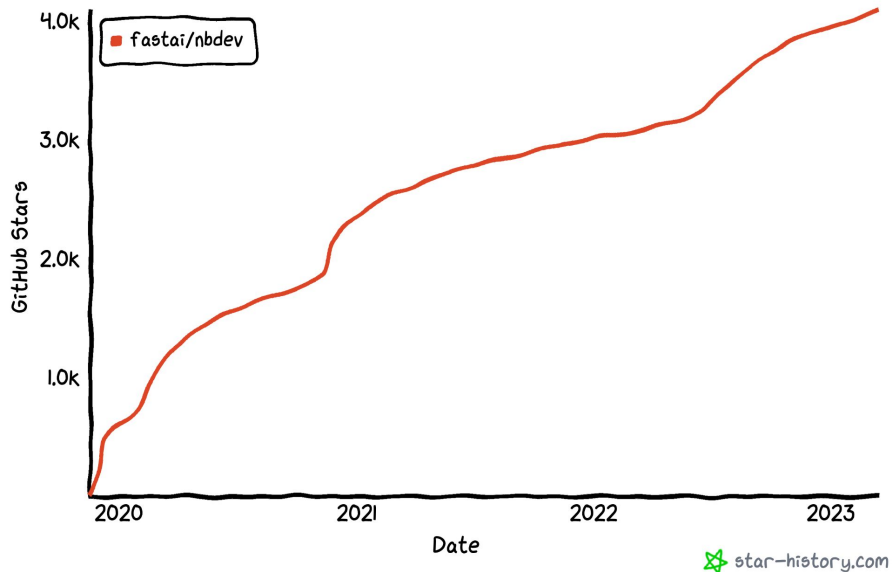
Install

After cloning this repository:

```
pip install -e .
```

Results

Star History



fastai Public

The fastai deep learning library

Jupyter Notebook 22.6k 7.3k

fastpages Public template

An easy to use blogging platform, with enhanced support for Jupyter Notebooks.

Jupyter Notebook 3.3k 760

ghapi Public

A delightful and complete interface to GitHub's amazing API

Jupyter Notebook 364 Apache-2.0 40 32 0
Updated 11 minutes ago



Chris Lattner

Inventor of Swift and LLVM

I really do think [nbdev] is a huge step forward for programming environments.

Notebooks As Syntactic Sugar For Source Code

▲ erikgaas on Nov 20, 2020 | parent | context | favorite | on: Nbdev: A literate programming environment that dem...

So I use this in production at my company. It's an awesome tool. Personally when I'm coding in python I like to prototype in jupyter, copy code over, and then reimport anyway. Nbdev streamlines everything so I can write docs, tests, and code all in one place. And since the docs are just a jekyll site I can copy it to our documentation aws bucket in continuous integration. And with one command I can run all the notebook tests in CI as well.

The packaging is also really well thought out. I don't have to stress out about connecting setup.py with whatever publishing system we have. The settings.ini makes things sane and I can bump the version whenever I want.

A get a lot of skeptical looks when I say the source code is in notebooks, but that's just syntactic sugar for the raw source code. You still get to edit the raw code files and with one command sync everything with the notebooks. From my point of you it is close to a pareto improvement over traditional python library development.

▲ mloncode on Nov 20, 2020 | next [-]

Really interesting! Do you mind sharing what your company is? (I am the author of the blog post)

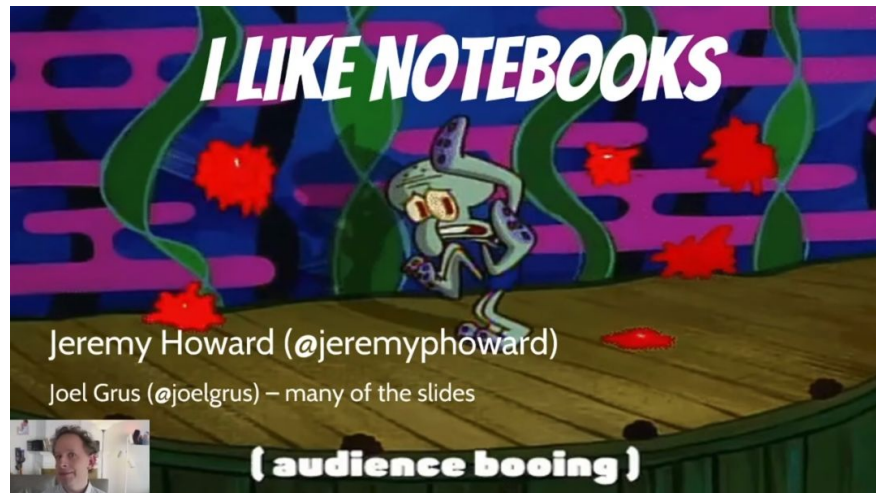
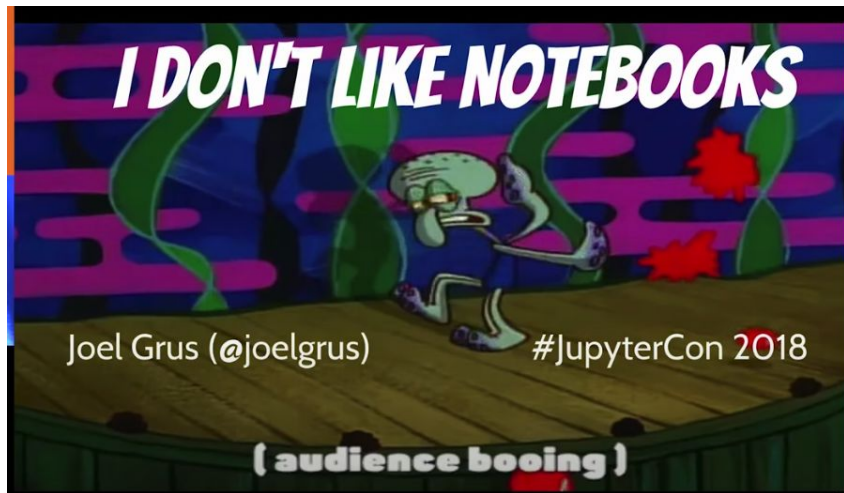
▲ erikgaas on Nov 23, 2020 | parent | next [-]

I work for Lyft's self driving car division, Level 5! Nbdev has been great. I use it a lot. Thank you for all of the work you've put into it!

Challenges With nbdev

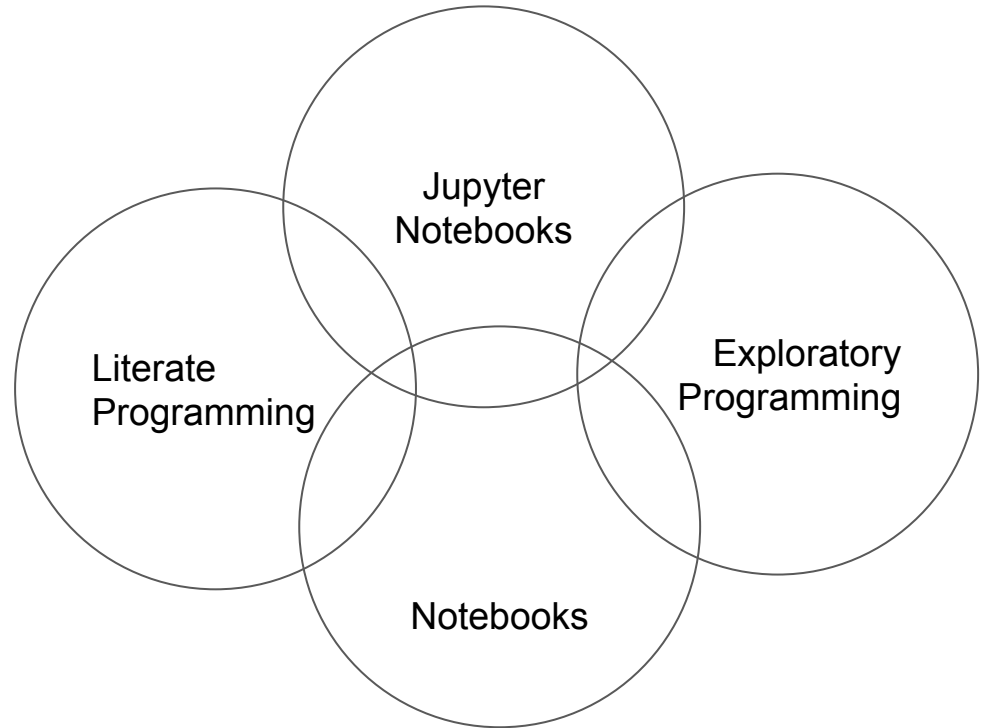
- Source of truth isn't plain text -> diffing on GitHub
- It's effectively a new programming language in a sneaky way!
- It's bootstrapped with notebooks, rather than building an IDE from the ground up: rough edges / limitations.

How We Write Code is Hotly Debated




We Should See The Forest For The Trees



- Literate programming and exploratory programming are not synonymous with Jupyter notebooks.
- We can innovate on notebooks
- We can use something else
- We are empowered to make tools in this space







These Ideas Are Now More Important Than Ever

Model: Code Interpreter **ALPHA**

 Thank you for uploading the file "plugins-demo.mp4." How can I assist you with this file?

 Can you extract the first 5 s of the video?

 Working...  Show work 

Stop generating

```
end_time = 5
extracted_clip = video_clip.subclip(start_time, end_time)

# Save the extracted clip
output_path = "/mnt/data/extracted-video.mp4"
extracted_clip.write_videofile(output_path, codec='h264')

output_path
```

STDOUT/STDERR

```
Moviepy - Building video /mnt/data/extracted-video.mp4
MoviePy - Writing audio in extracted-videoTEMP_MF_H87643
chunk: 0%|          | 0/111 [00:00<?, ?it/s, n
```

Stop generating

These Ideas Are Now More Important Than Ever

%%ai_ask

Can you write a function that takes a `list` of numbers `and` returns the sum of the numbers?

✓ 7.3s

Sure! Here's a simple Python function that takes a list of numbers as input and returns their sum:

```
def sum_of_numbers(numbers):  
    return sum(numbers)
```

This function uses the built-in `sum()` function to add up all the numbers in the input list. You can call this function and pass in a list of numbers like this:

```
my_list = [1, 2, 3, 4, 5]  
total = sum_of_numbers(my_list)  
print(total) # Output: 15
```

Of course, you can modify or customize this function depending on your requirements.

%%ai_continue

Can you make the return value be a product of the numbers `and not` a sum?

✓ 11.0s

Sure, I can easily modify the previous function to return the product of the numbers instead of their sum. Here's the updated function:

```
def product_of_numbers(numbers):  
    product = 1  
    for num in numbers:  
        product *= num  
    return product
```

This function initializes a variable called `product` to 1 and then uses a `for` loop to iterate through the input list of numbers. For each number in the list, it multiplies it with the current value of `product`. Finally, it returns the resulting product.

You can call this function and pass in a list of numbers like this:

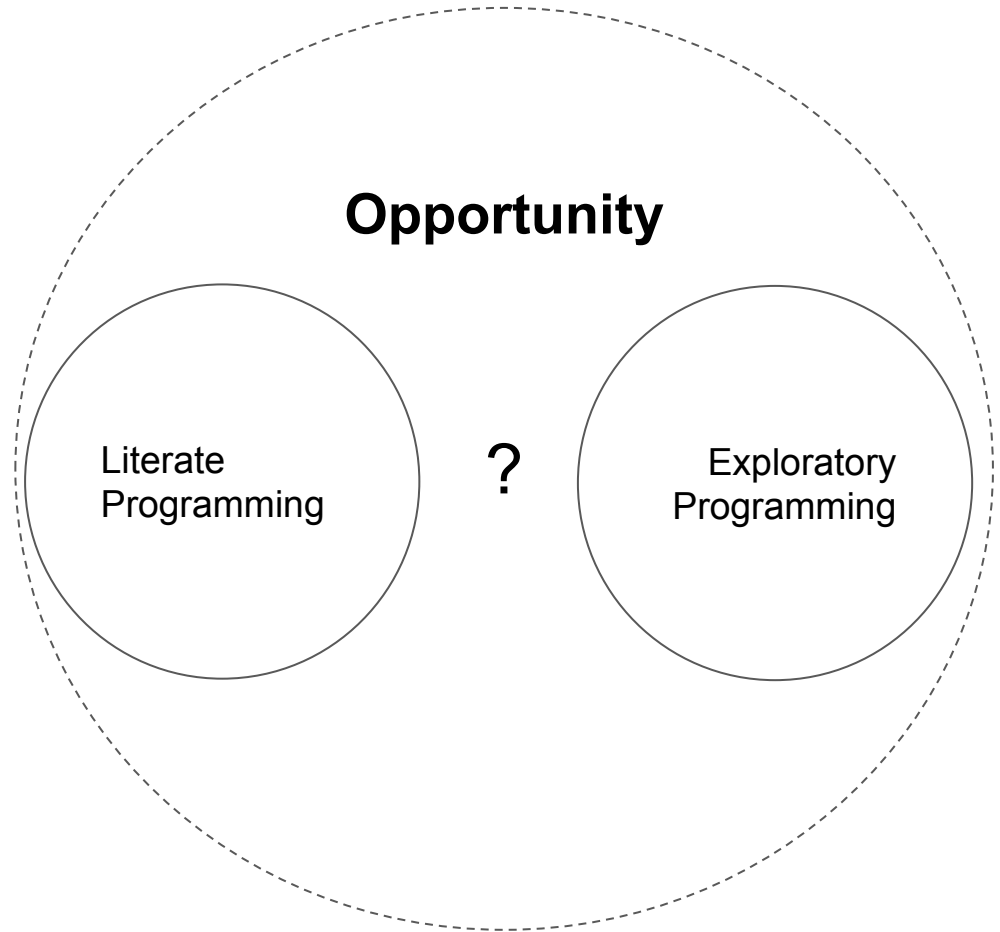
```
my_list = [1, 2, 3, 4, 5]  
total = product_of_numbers(my_list)  
print(total) # Output: 120
```

Again, you can modify this function according to your specific needs.

Source: Radek Osmulski @radekosmulski

Call To Action

- Let's build tools.
- Think outside the box, don't get stuck in "Notebooks for analytics" or "Hosted collaborative notebooks".
- Few people are working on this problem.



Get In Touch

<https://hamel.dev>