

Lecture 0: Introduction



pythonTM

Python Fundamentals Course

What is a Python?

- ▷ Python – is a very high-level multiparadigm general-purpose programming language that is focused on code quality and development speed
- ▷ Python emerged in 1991 and found its way into almost all areas of programming
- ▷ The definition above sounds complex. Let's break it down



Guido van Rossum aka GvR – the creator of Python. Also known as BDFL – benevolent dictator for life

Python: a very high-level (VHHL) programming language

```
int cmp_lessd(const void *a, const void *b)
{
    return *((double *)a) > *((double *)b);
}

int main(int argc, char *argv[])
{
    double res[argc - 1];
    long cnt = 0;
    for (long i = 1; i < argc; i++) {
        typeof(*res) el = atof(argv[i]);
        if (!fmod(el, 2.0))
            res[cnt++] = el;
    }
    qsort(res, cnt, sizeof(*res), cmp_lessd);
    for (long i = 0; i < cnt; i++)
        printf("%lf\n", res[i]);
    return 0;
}
```

18 lines of C code

Example: lec0/codesize.c

```
print('\n'.join(sorted([i for i in sys.argv[1:]
                        if not float(i) % 2],
                        key=lambda x: float(x))))
```

Just 1 line of Python code

Of course, it does not have to be that concise.
In Python we code for readability and the ease of further support

Being skilled is worth nothing if your skilled colleagues are unable to read your code

Example: lec0/codesize.py

Python: a multiparadigm programming language








- ▷ Object-oriented
- ▷ Procedural
- ▷ Imperative
- ▷ Functional
- ▷ Structured
- ▷ Reflective
- ▷ Declarative
- ▷ Aspect-oriented
- ▷ Asynchronous

Python runs (almost) everywhere

- ▷ On desktops and laptops: CPython, PyPy, Stackless
- ▷ On top of other languages: Jython (Java), IronPython (C#), Cython/Nuitka (C/C++), Emscripten/Pyiodide (JS/WASM), RustPython (Rust)
- ▷ On embedded systems: MicroPython/CircuitPython (MCUs like ESP32), CPython (MPUs like routers)
- ▷ On servers: Django, Flask, Muffin, ... frameworks (WEB); Twisted, Tornado, ... frameworks (general-purpose networking); and many more ...
- ▷ On smartphones and tablets: Kivy, BeeWare, Chaquopy, PyQT
- ▷ In ASIC development: MyHDL
- ▷ And even on blockchain: Vyper for smart contracts development

Case Study: Python popularity (TIOBE index)

[Book a demo](#)

Sep 2022	Sep 2021	Change	Programming Language		Ratings	Change
1	2	▲		Python	15.74%	+4.07%
2	1	▼		C	13.96%	+2.13%
3	3			Java	11.72%	+0.60%
4	4			C++	9.76%	+2.63%
5	5			C#	4.88%	-0.89%
6	6			Visual Basic	4.39%	-0.22%
7	7			JavaScript	2.82%	+0.27%

As of
4 Sep 22

Case Study: Python popularity (PYPL index)

PYPL Index

10 TOP IDE

10 TOP ODE

10 TOP DB

PYPL PopularitY of Programming Language

Worldwide, Sept 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.29 %	-1.8 %
2		Java	17.31 %	-0.7 %
3		JavaScript	9.44 %	-0.1 %
4		C#	7.04 %	-0.1 %
5		C/C++	6.27 %	-0.4 %
6		PHP	5.34 %	-1.0 %

As of
4 Sep 22

The PYPL PopularitY of

Python popularity: What it means

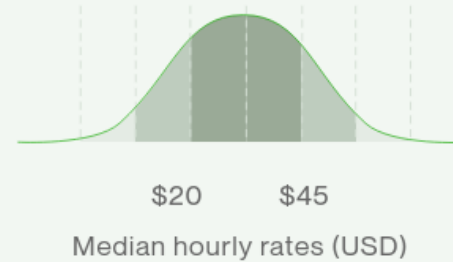
- ▷ In recent years most of universities switched from widespread of outdated programming languages to Python
- ▷ It means that each year they are supplying thousands of young Python developers to the market
- ▷ Supply is correlated with demand. Which results in more projects done using Python and more Software Engineer jobs requiring Python being listed
- ▷ This trend is meant to continue for decades, as we've seen with different programming languages before
- ▷ To be competitive you should consider learning Python

Case Study: Freelance jobs

Python Developers on
Upwork
can earn \$20–\$45/hr.

Learn more below about how you can earn a career
on the world's work marketplace.

Start Earning



$(\$20 \dots \$45) \times 8 \text{ hr} \times 5/7 \times 30 = \$3428 \dots \$7714 \text{ per month}$

As of
4 Sep 22

Case Study: Full-time jobs



United States / Job / Python Developer

Average Python Developer Salary

Pay

Job Listings

\$79,395 / year ▾

Avg. Base Salary (USD)



The average salary for a Python Developer is \$79,395

Base Salary ⓘ	\$51k - \$107k	
Bonus	\$306 - \$15k	
Profit Sharing	\$0 - \$3k	
Total Pay ⓘ	\$43k - \$122k	

Based on 94 salary profiles (last updated Aug 02 2022)

$(\$51k \dots \$107k) / 12 = \$4250 \dots \8916 per month

As of
4 Sep 22



Installing Python: Windows

► Take a look at Windows terminal:

```
Command Prompt
Microsoft Windows [Version 10.0.19042.572]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\thd>dir
Volume in drive C has no label.
Volume Serial Number is A040-CBA4

Directory of C:\Users\thd

08/03/2022  12:15 AM    <DIR>          .
08/03/2022  12:15 AM    <DIR>          ..
11/28/2020  07:39 PM    <DIR>          3D Objects
11/28/2020  07:39 PM    <DIR>          Contacts
08/03/2022  12:15 AM    <DIR>          Desktop
05/07/2021  08:08 PM    <DIR>          Documents
11/28/2020  07:39 PM    <DIR>          Downloads
11/28/2020  07:39 PM    <DIR>          Favorites
11/28/2020  07:39 PM    <DIR>          Links
11/28/2020  07:39 PM    <DIR>          Music
01/29/2022  11:11 PM    <DIR>          Pictures
11/28/2020  07:39 PM    <DIR>          Saved Games
11/28/2020  07:39 PM    <DIR>          Searches
10/30/2021  03:22 PM           2,544 Sti_Trace.log
11/28/2020  07:39 PM    <DIR>          Videos
04/18/2021  05:05 AM    <DIR>          Zotero
               1 File(s)                2,544 bytes
              15 Dir(s)  38,977,695,744 bytes free

C:\Users\thd>
```

It kinda sucks...

*To develop something good
one needs a Linux
enviroment*

– Dev Bible 3:14

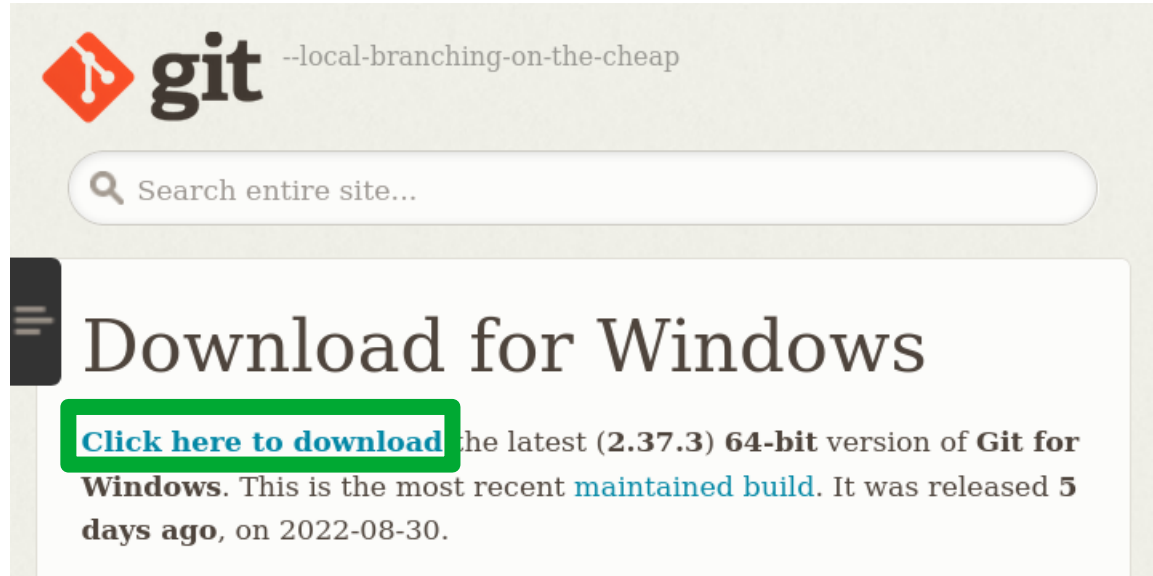


We will fix that

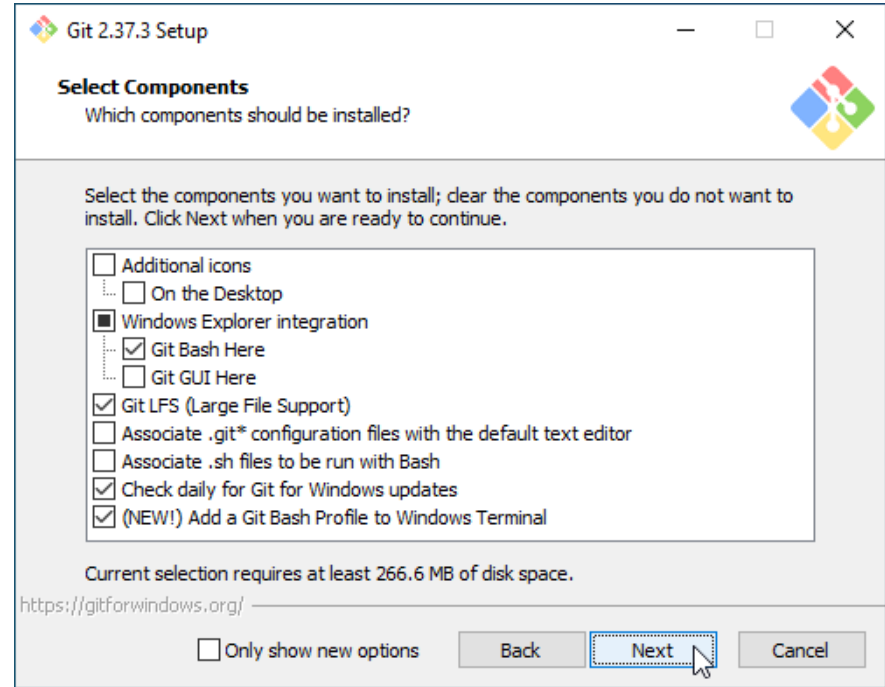
Fixing Windows Terminal

- ▷ One solution is to install Git For Windows
- ▷ It comes with a fancy terminal plus a minimal subset of UNIX environment
- ▷ Since we are going to learn Git, we'll need it anyway

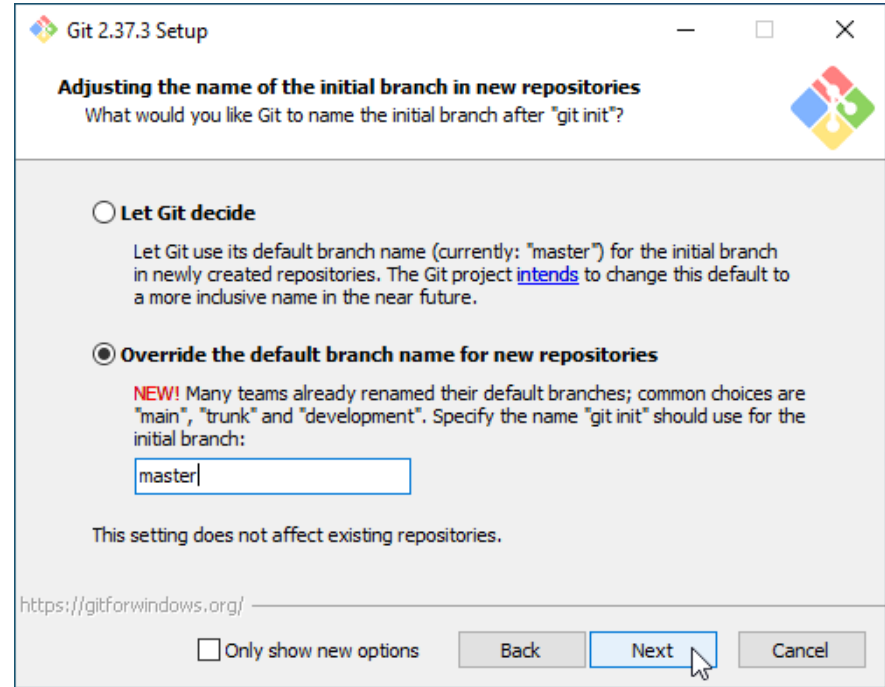
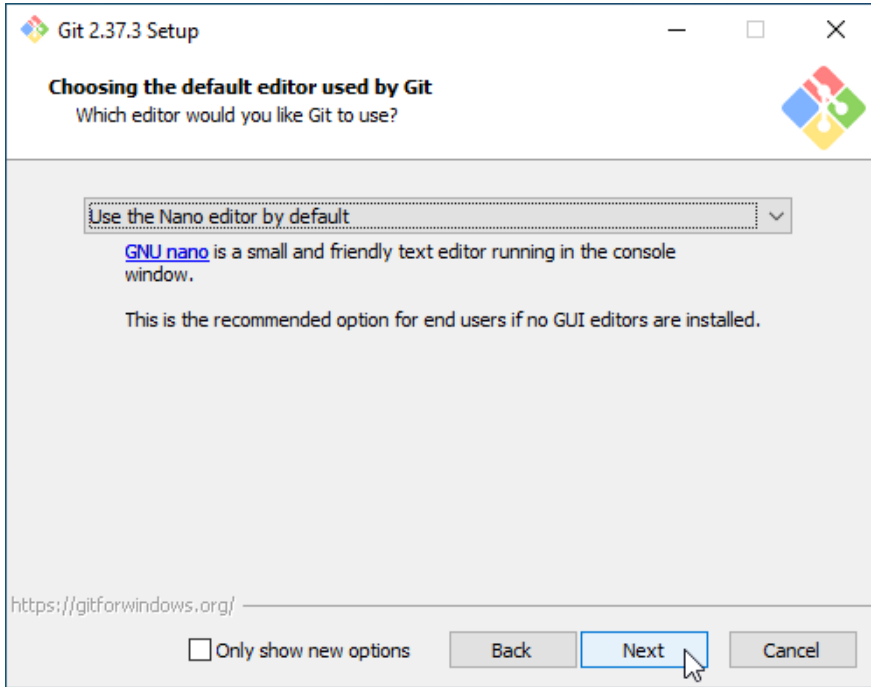
Visit official git-scm.com website and download the [latest Git For Windows](#) distribution



Fixing Windows Terminal...

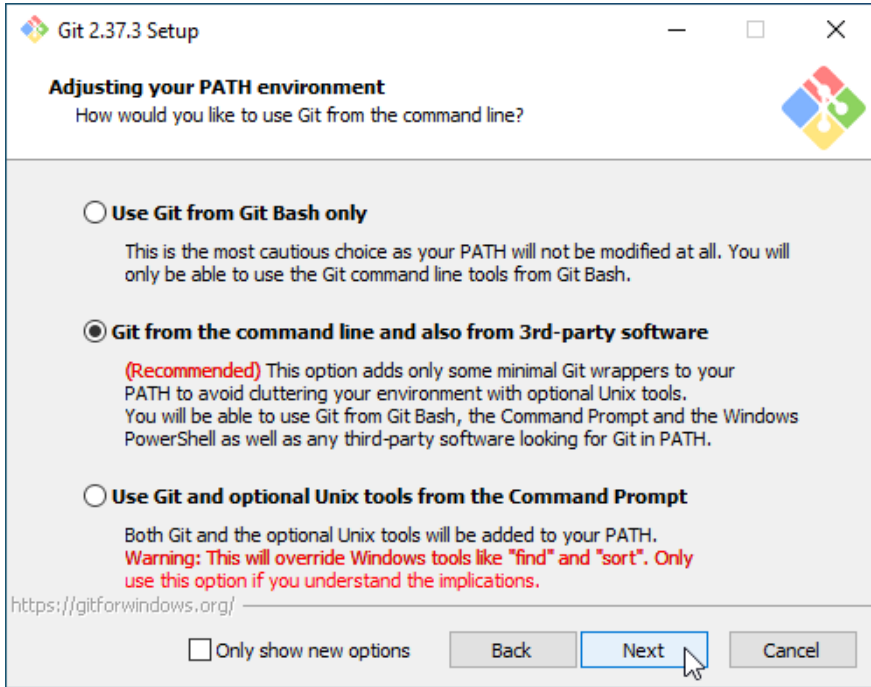


Fixing Windows Terminal...

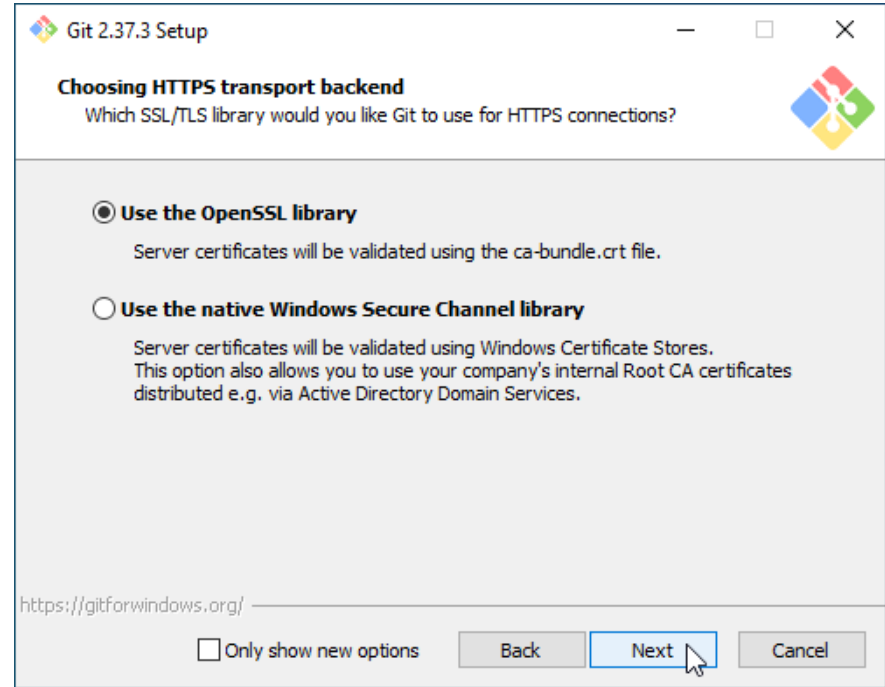


i.e. absolute classics

Fixing Windows Terminal...

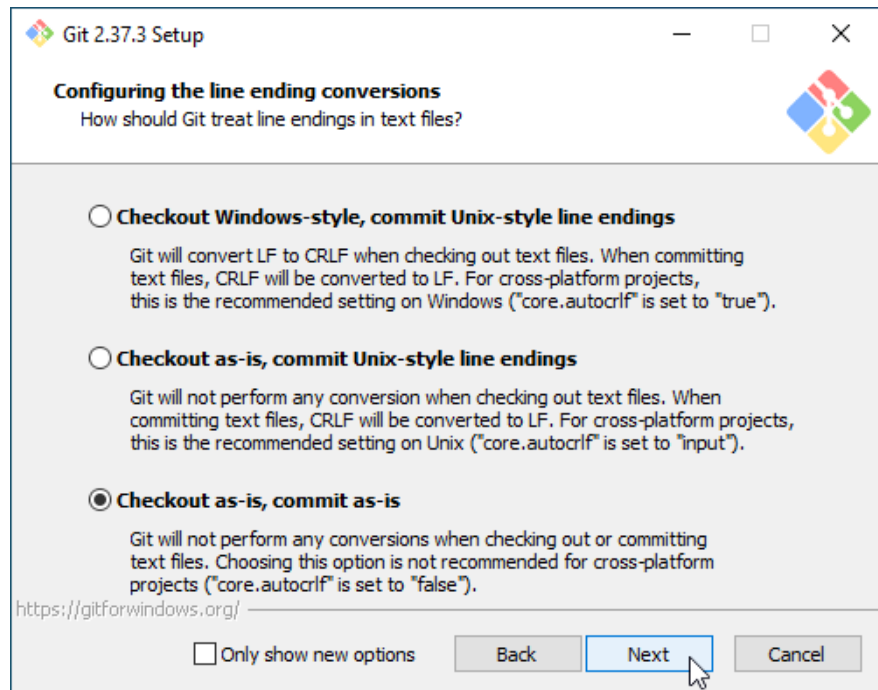


We will call everything else from the Bash

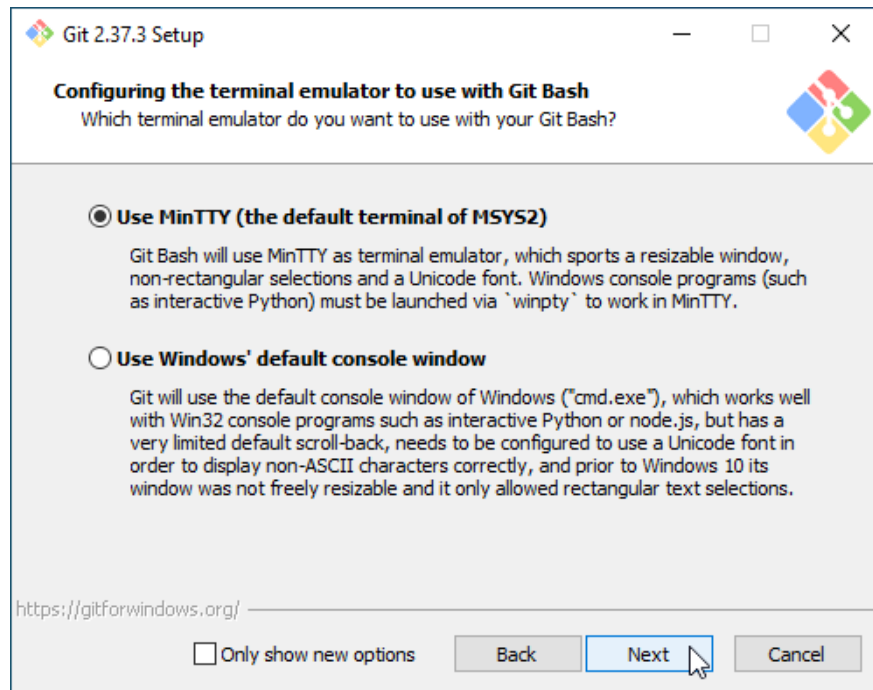


Never trust Windows bundled certificates

Fixing Windows Terminal...

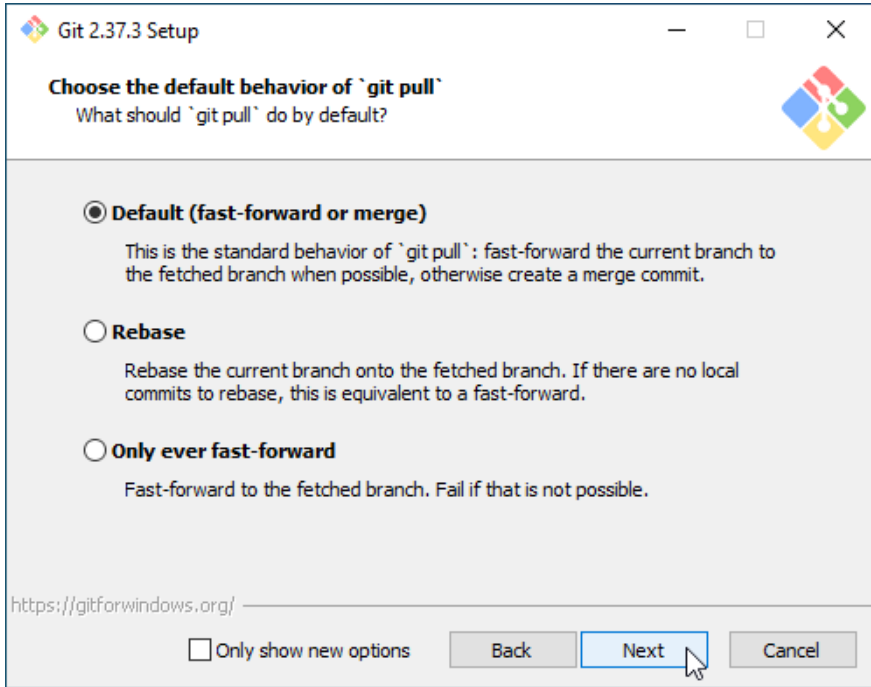


Use a right editor instead of relying on git to fix the Windows carriage-return mess

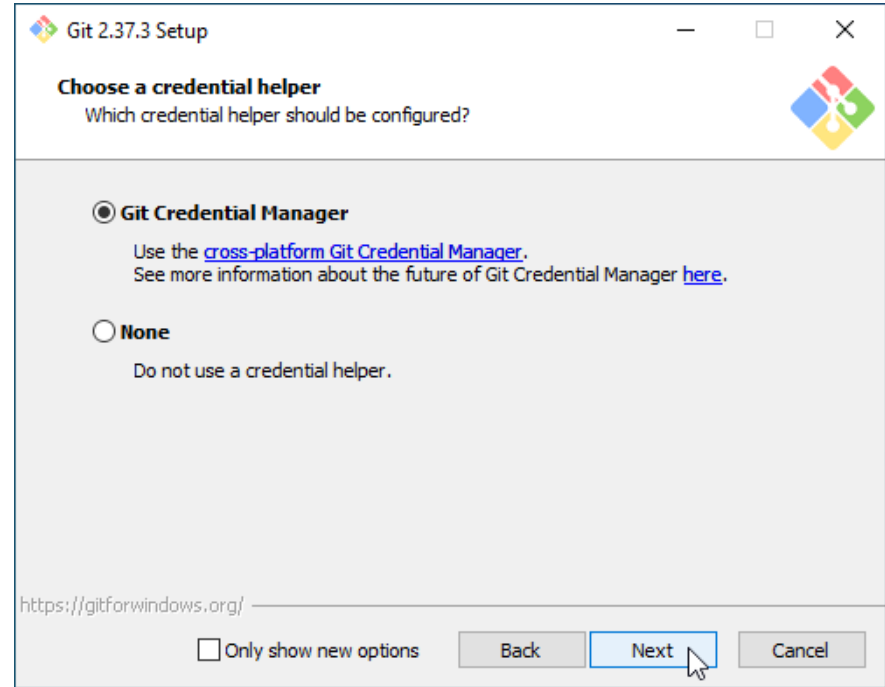


That's the whole purpose of this show

Fixing Windows Terminal...

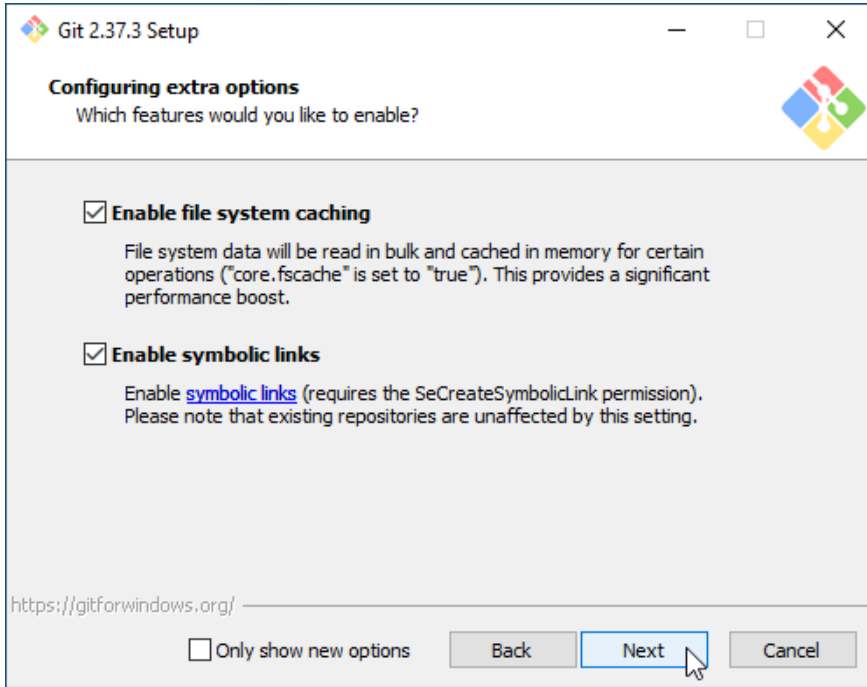


Stick to defaults, it could be changed later

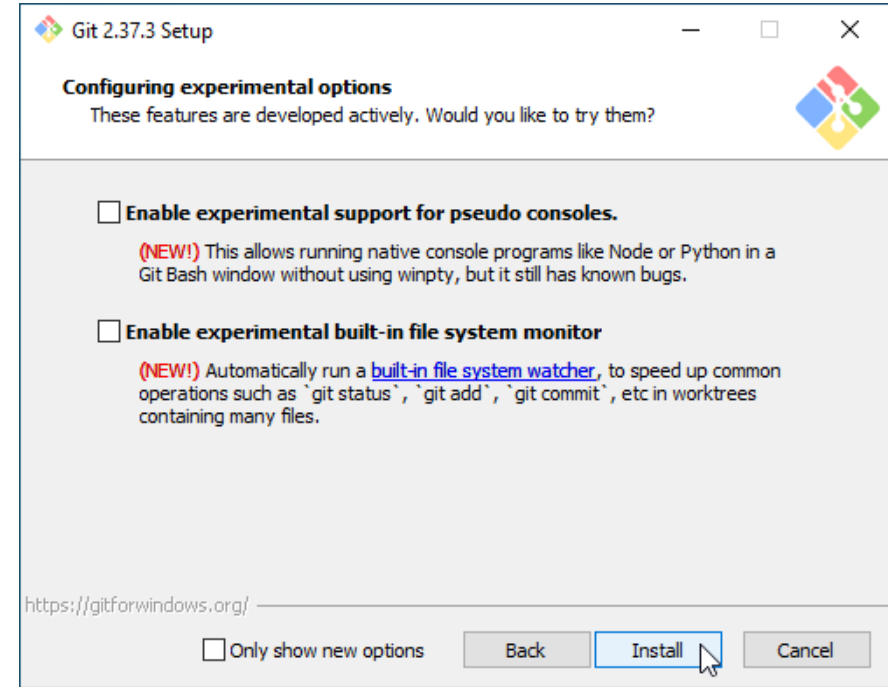


Use safe alternative if you're going to save your login info. If not (ever), select None

Fixing Windows Terminal...

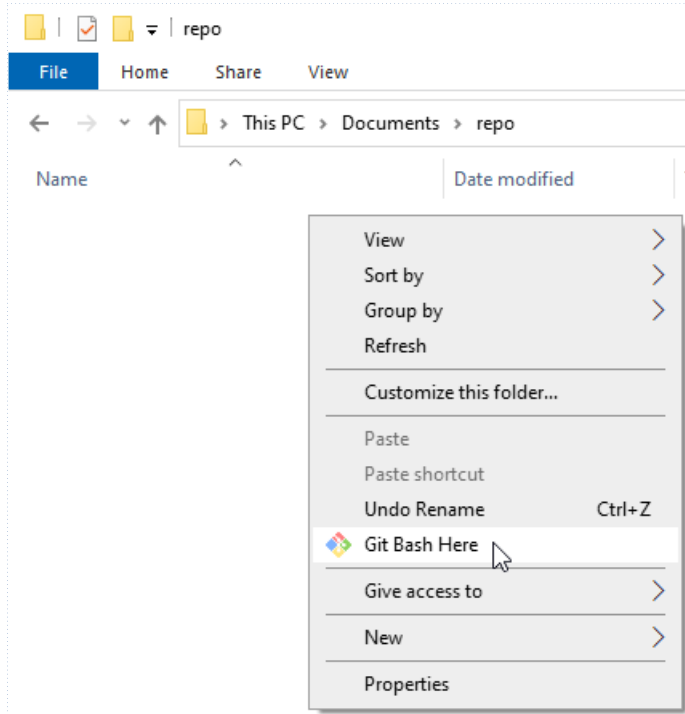


Speed is always nice thing to have, especially with large repos.
Symlinks are heavily used by pro repositories



No experimental stuff until it is stable
But you may still want to try the FS monitor

Fixing Windows Terminal...



```
MINGW64/c

thd@DESKTOP-K8CI5H7 MINGW64 /c/Users/thd/Documents/repo
$ cd /c/

thd@DESKTOP-K8CI5H7 MINGW64 /c
$ ls
'$RECYCLE.BIN'/'      'Program Files'/'      bootmgr
'$WinREAgent'/'      'Program Files (x86)'/'  inetpub/
BOOTNXT              ProgramData/            pagefile.sys
Cadence/             'System Volume Information'/'  swapfile.sys
'Documents and Settings'@  Users/
DumpStack.log.tmp       windows/

thd@DESKTOP-K8CI5H7 MINGW64 /c
$
```

Now that is much much better

Be sure to set your preferred look in terminal Options,
enable the font smoothing and other useful stuff like en_US.UTF8 locale

Fixing Windows Terminal (alternatives)

- ▷ Another solution is to use a full-featured Linux terminal like *xfce4-terminal* from [Cygwin](#) package
- ▷ [Cygwin](#) acts as a GNU/Linux compatibility layer for Windows and provides even more feature-rich subset of Linux tools
- ▷ Go on, try it yourself. And choose what fits you best
- ▷ But do not install a python from Cygwin repo.
It is not a Windows Python, but a GNU/Linux Python relying on Cygwin portability libraries.
Thus it acts in a very different way. Sometimes very inefficient.
But still very good for testing Linux-related Python stuff if you don't want to fire up a Linux VM for some reason
- ▷ WSL may also be used. But WSL generally sucks
- ▷ Or you might want to install a third-party terminal (there's a plenty)

Installing Python: Windows

► Now that we have terminal fixed let's install Python

Visit python.org website and download the latest Python **3** release for Windows (Windows Installer)

Remember:

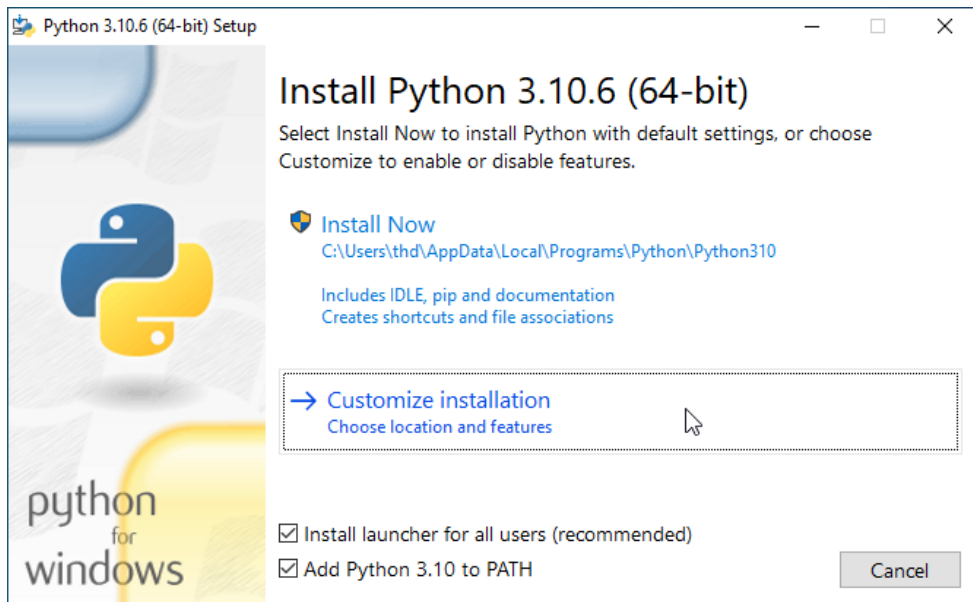
Python 2 is outdated and reached end-of-life on 1 Jan 2020. You may still encounter Python 2 in some old systems (like banks) where it is too expensive to port the codebase to Python 3. But it is really really rare to see now.

Python 3 is the currently actual branch

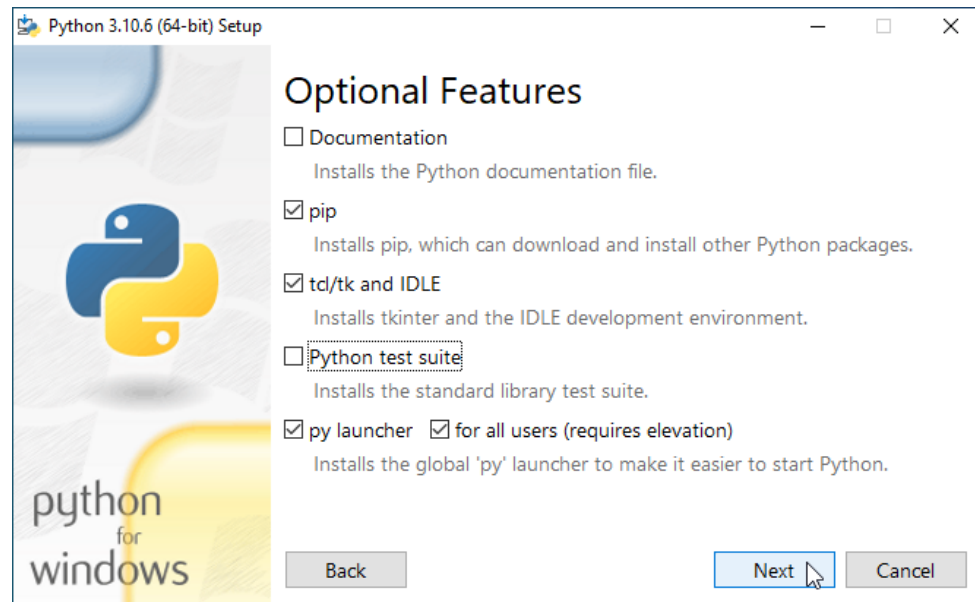
The screenshot shows the Python.org website's 'Downloads' section for Windows. The navigation bar includes links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The breadcrumb trail is 'Python >>> Downloads >>> Windows'. The main heading is 'Python Releases for Windows'. Below this, there are two links: 'Latest Python 3 Release - Python 3.10.6' and 'Latest Python 2 Release - Python 2.7.18'. The first link is highlighted with a green box. To the right, a table provides more details about the recommended version.

Windows	Windows	Recommended
Windows installer (64-bit)		

Installing Python: Windows

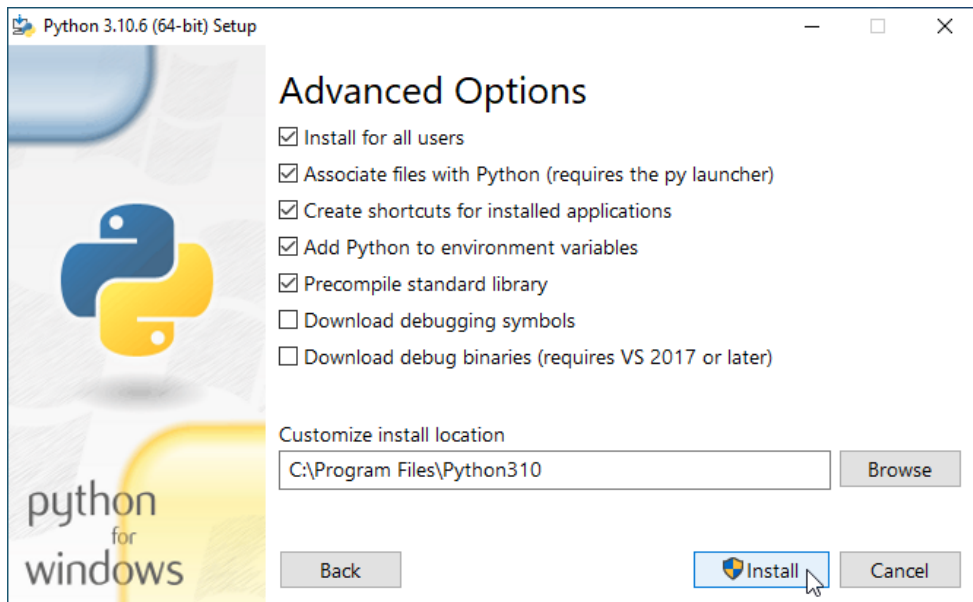


Having Python added to PATH is crucial.
IDLE is where we are going to code, and it
relies on tk.
Python launcher is handsome

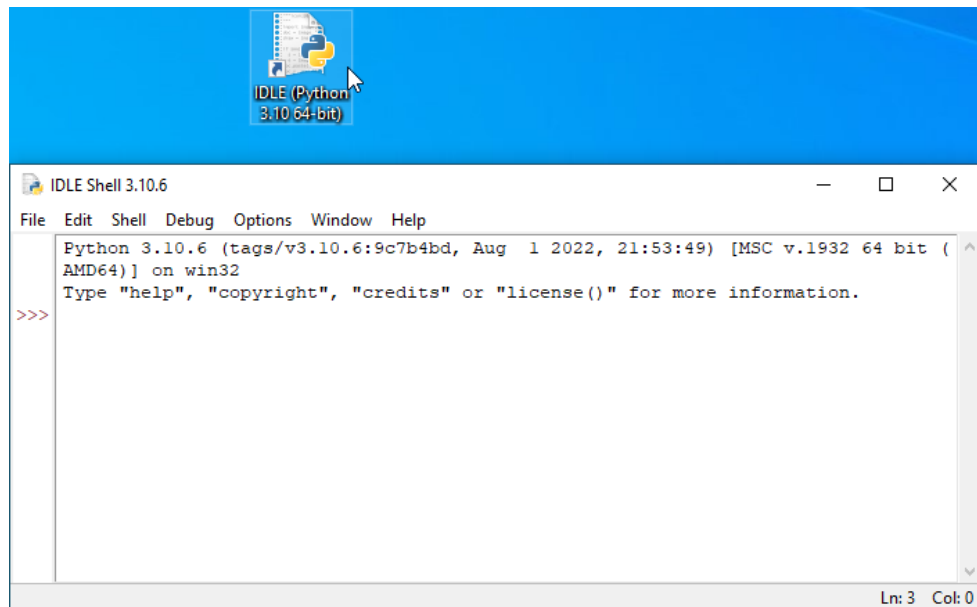


We don't need documentation since it's in
shitty Microsoft CHM format.
Test suite is not needed since we don't
develop the standard library itself
PIP is the absolutely necessary Python's
package manager

Installing Python: Windows



Install globally. Files association is good for using the launcher unless you prefer to have them associated with the editor.
Adding python to env will allow it to be called from everywhere
Having standard lib precompiled will improve the startup time



Create an IDLE shortcut and try running it

Try running “python” in terminal we’ve just set up as well (Ctrl + D is for exit there)

We’ll set up IDLE in a minute

Installing Python: GNU/Linux

▷ Below is an example for Arch Linux:

▷ On most systems
Python3 is already
there
The only thing needed
is to install additional
packages of the dev
env

```
# Update package list
sudo pacman -Syy
# Upgrade system packages
sudo pacman -Su
# Install tk for IDLE to run, as well as PIP for future
sudo pacman -S tk python-pip
# Add user bin to PATH
echo 'export PATH="$(realpath ~/.local/bin):${PATH}"' \
    >> ~/.bashrc
# Relogin & continue
# You should be able to see .local/bin in PATH
echo $PATH
# Upgrade pip
pip install --user --upgrade pip

# Test idle, you may want to create a shortcut for it
idle3
```



What is IDLE?

- ▷ IDLE is a built-in minimalist development environment that is shipped with Python distribution
- ▷ That is, if you reach some new machine and need to fix something ASAP, IDLE is there to help
- ▷ IDLE is NOT a full-featured IDE
- ▷ But you don't need a full-featured Python IDE unless your projects become that huge (at least 10k lines of code)
- ▷ Python is different from many languages you may be familiar with. And as a rich-introspection language, Python does not require traditional debugging instruments (such as debugger or memory watch), but it has them if ever needed (which is a very very rare scenario)

A brief note on IDEs

- ▷ When you begin studying the language, IDEs only tend to distract you from learning
- ▷ Starting having IDE you will either end up frustrated fighting it (auto-formatting, endless warnings, etc.), or will learn nothing about the language itself. IDEs often extend the language functionality with additional features that are not part of the Python itself (but developer may falsely assume they are)
- ▷ I've met a pro-ish developer in my career who was using PyCharm since the very beginning, and didn't even know how to run a Python script on a remote server. What a shame!
- ▷ Avoid using IDEs at any cost, unless you become proficient enough to see their use as absolutely justified
- ▷ In fact what most pro developers need is a decent hackable editor (like Atom/NeoVIM/VSCodium) extended with plugins to meet the developer's needs

Sources of information on Python

- ▷ Shared Course Materials (link attached)
- ▷ docs.python.org/3 – Official documentation
That URL is worth to be learnt by heart
- ▷ The official documentation also has a nice [Tutorial](#). It is concise and can make you up and running in no time. But make sure you follow all the links in the text there
- ▷ [StackOverflow](#). An unmatched source of useful information. You may just scroll around the top-rated questions under the [python tag](#), ask your own questions or even try answering the others'. A high-karma profile on StackOverflow will aid you during the employment
- ▷ Books. They are slow to read. But by the end of the day inevitable, if you wish to become a real pro developer. Make sure you not only read them (e.g. getting the theory), but also follow the examples trying your ideas in the meanwhile (e.g. getting the practice)

Sources of information on Python...

- ▷ The PyDoc system

When you're developing, it is much faster to type `help(something)` than to look for the information elsewhere. Built-in documentation is one of the powerful Python features

And you may also run `pydoc -b` from the terminal. The browser will open up providing you with all the automatically generated documentation.

It is a handy feature in case you ever end up developing where there is simply no internet connection

- ▷ Sources to **AVOID** (*based on personal experience*):
Other videos; forums without a review system; Habr; blogs of switchers and complete amateurs

Recommended Literature

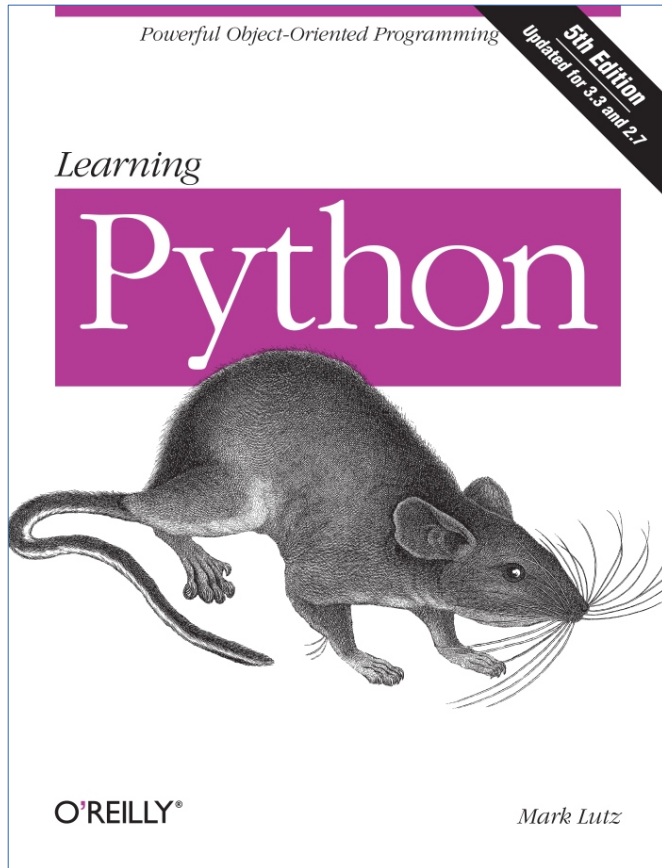
**HIGHLY
RECOMMENDED**

Mark Lutz. *Learning Python* (5th ed.). Sebastopol, CA: O'Reilly Media, 2013

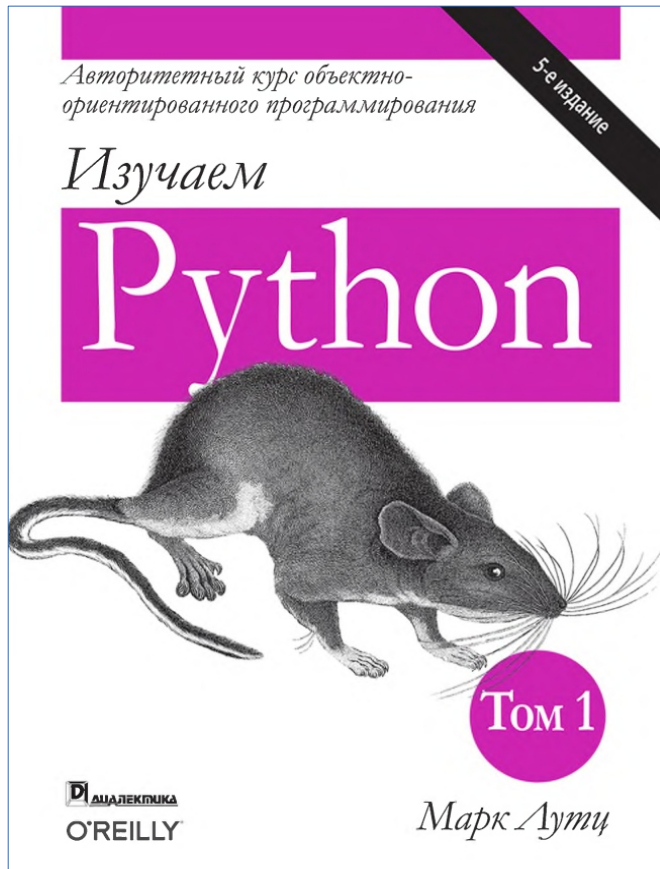
An absolutely super-heroic book. It is much like a Thinking in Java by Bruce Eckel, but for Python
It teaches you not only the language, but also the philosophy behind the Python itself thus allowing to write really pythonic code

Note: The Book covers Python 2 and Python 3. Python 2 reached its end-of-life on 1 Jan 2020 and is dead for now. Skip everything related to Python 2 while reading

It covers Python up to 3.3 version (the latest at the moment is 3.10). 99% you'll learn for Python 3.3 is valid for Python 3.10, but there have been new useful language features added that are worth getting acquainted with



Recommended Literature



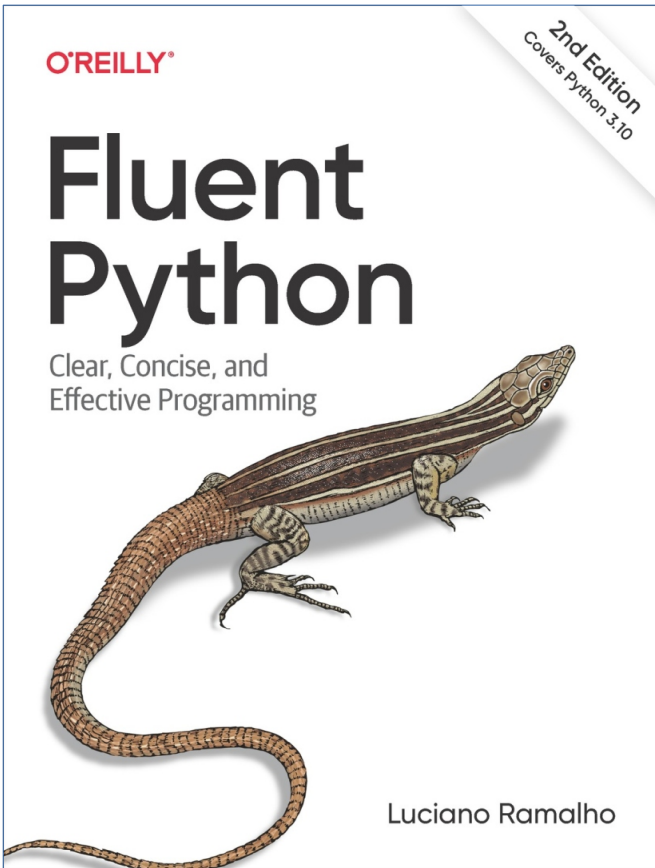
Translation available

Лутц, Марк. Изучаем Python, 5-е изд.: Пер. с англ. — СПб.: ООО “Дialeктика”, 2019, 2020.

If you are ok with learning in English, avoid translations at any cost. There are some language terms that simply can not be directly translated without introducing the new obscure terminology.

A Software Engineer needs to be good at working with the technical documentation in English (which means both reading and writing it). Otherwise it is truly a “professional suicide”

Recommended Literature



Luciano Ramalho. Fluent Python (2nd ed.). Sebastopol, CA: O'Reilly Media, 2022

This book is oriented towards more professional, e.g. Middle developers, who already know the Python language fundamentals and want to become more proficient with it. As well as towards switchers coming from other programming languages willing to make their code more pythonic.

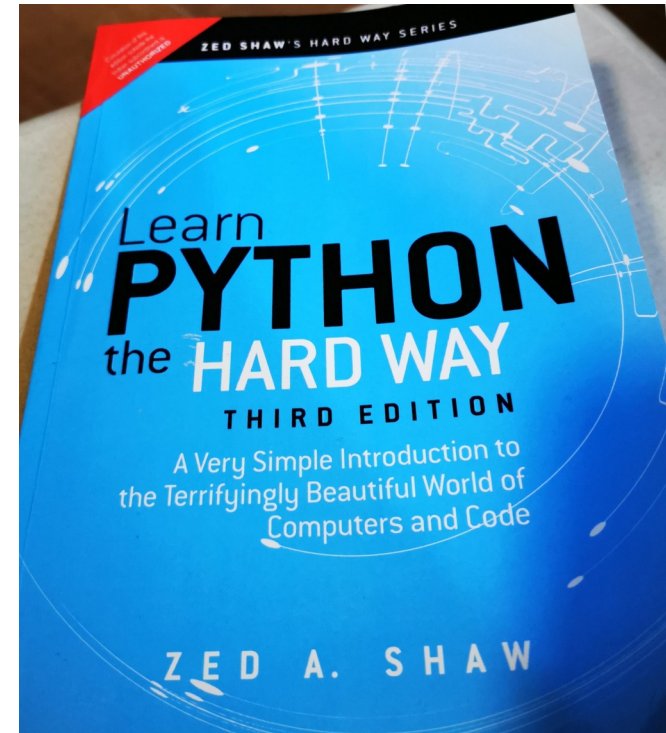
Anyway, it is a good read worth considering sooner or later

Don't get fooled

With Python it is very easy to get the task solved fast, because the language was specifically designed that way.

But it is extremely hard to write good, truly pythonic code with all the modern features put to work. It takes time and practice to become a real pro.

It is similar to learning how to drive a car. First you screw up all the time. Then it is starting to work out. At this moment it feels like “ok, I’ve learnt how to do it”. But it only seems so. And becoming a skilled driver requires much more practice



Other books

- ▷ Below is a list of other books
(exactly as they are named on my
computer).

Many of them are faster to read. But on the other hand,
they won't provide you with a knowledge deep enough
to truly master Python. So are they worth spending time?
Probably. It depends on what style of learning you like most

- 1_Python Crash Course, 2nd Edition A Hands-On, Project-Based Introduction to Programming by Eric Matthes.pdf
- 2_Automate the boring stuff with python 2nd edition by Al Sweigart.pdf
- 3_Python Flash Cards Syntax, Concepts, and Examples by Eric Matthes.pdf
- 4_Impractical Python Projects Playful Programming Activities to Make You Smarter by Lee Vaughan.pdf
- 5_Real-World Python A Hackers Guide to Solving Problems with Code by Lee Vaughan.pdf
- 6_Python One-Liners Write Concise, Eloquent Python Like a Professional by Christian Mayer.pdf
- 7_Serious Python Black-Belt Advice on Deployment, Scalability, Testing, and More by Julien Danjou.pdf