

MODERN PYTHONUTVECKLING

Viktor Ahlqvist

OpKoKo 19.2

 /vikahl/modern-python-development

- Python 2 EOL om 1 månad och 22 dagar
- **Inga** uppdateringar efter det
- Migrera nu!



Figur 1: Lisa Roach,
Twitter: @lisroach

- "Moderna" Python-versionerna
 - Python 3.6 – 2016-12-23 (säkerhetsuppdateringar t.o.m. 2021)
 - Python 3.7 – 2018-06-27
 - Python 3.8 – 2019-10-14
- Exempel fungerar på Linux/Mac, oftast också på Windows

AGENDA

Installation

Verktyg och kod-kvalitet

Type hinting


Testing

Paketera som modul

Automatisera tester

Projektmall med cookiecutter

- Paket som kommer användas som exempel under föreläsningen
- Extraherar *TODO* från kod och retunerar JSON
- Bra som exempelmodul, men begränsad "riktig" användbarhet

 /vikahl/todo-extractor

INSTALLATION

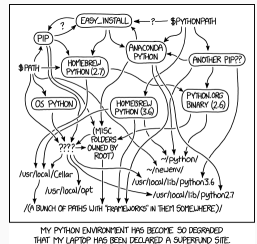
SYSTEMPYTHON OCH VIRTUALENV

Tre korta råd

1. Undvik installation i systemets Python
2. Använd virtuella miljöer *virtual env*
3. Ha system för flera Python versioner installerade samtidigt

Tänk på att

- Kan inte se vad som installerats direkt och vad som är beroenden på dessa
- Kan inte på ett generellt sätt låsa beroenden i flera led



Figur 2: Randall Munroe, xkcd nr 1987

- Lokala paket per miljö/projekt
- Lätt "avinstallation" genom att ta bort mappen
- Skapa i skalet eller via IDE och aktiveras med aktiveringsskript

```
python3 -m venv venv
```

Glöm inte ignorera mappen i Git!

PYENV - HANTERA VERSIONER

- Olika Python-version per projekt
- Installera versioner oberoende av systemets pakethanterare
- Automatisk aktivering i specifika mappar med `.python-version` filer
- Installera med plugins för virtualenv-stöd

```
viktor@aldertech:~$ python3 -V
Python 3.7.3
viktor@aldertech:~$ cd todoextract/
(todo-extractor) viktor@aldertech:~/todoextract$ python3 -V
Python 3.8.0
(todo-extractor) viktor@aldertech:~/todoextract$ pyenv shell 3.5.3
viktor@aldertech:~/todoextract$ python3 -V
Python 3.5.3
—
```

PIPX - HANTERA "BINÄRER"

- Isolerar Python program i separata venv
- Kan både installera och köra direkt
- Enkel uppdatering av allt installerat `pipx upgrade-all`
- *Kan ha flera versioner av samma program, men måste då manuellt symlänka*

Installera med `pip3 install pipx`

Bör vara *den enda* paketet som är installerat i systemets Python.

- Hanterar både beroenden och virtualenv
- Låser beroenden i alla led med *Pipfile* och *Pipfile.lock*

Verkade lovande, men har tappat fart och orsakar lika många problem som det löser.

VERKTYG OCH KOD-KVALITET

- Program som analyserar koden för potentiella fel
- PEP 8 – *Style guide for Python code*

- Stilfel, syntaxfel, docstrings
- Statisk analys
- Gnällig, men fångar mycket
- Kräver nästan alltid konfig.
- Har plugins

```
C0326: No space allowed around keyword  
argument assignment  
C0114: Missing module docstring  
W0102: Dangerous default value [] as  
argument  
C0116: Missing function or method  
docstring  
E0602: Undefined variable 'my_list'
```

Your code has been rated at -12.50/10

```
def add_2(arg = []):  
    arg.append(2)  
    print(arg)  
    return my_list
```

- Mindre gnällig
- Lätt att bara slänga in ett projekt
- Hittar mycket, men inte lika mycket
- Har också plugins

```
E251 unexpected spaces around keyword /  
parameter equals
```

```
F821 undefined name 'my_list'
```

```
def add_2(arg = []):  
    arg.append(2)  
    print(arg)  
    return my_list
```

- Söker efter säkerhetsbrister och bedömer allvarlighet och tillförlitlighet
- Söker både i koden och i inkluderade paket

```
>> Issue: [B502:ssl_with_bad_version] ssl.wrap_socket call with  
insecure SSL/TLS protocol version identified, security issue.  
Severity: High   Confidence: High  
Location: ./ssl-insecure-version.py:4  
More Info: https://bandit.readthedocs.io/en/latest/plugins/b502...
```

```
3
```

```
4         ssl.wrap_socket(ssl_version=ssl.PROTOCOL_SSLv2)
```

```
5         SSL.Context(method=SSL.SSLv2_METHOD)
```


- Formatterar koden automatiskt
- Black – *The uncompromising code formatter*
 - radlängd
 - normalisering av citattecken
- Garanterar samma exekvering av koden innan/efter
- Finns alternativ: YAPF, autoformatter

Mitt råd: Välj Black och fokusera på affärsnytta istället

LINTER OCH AUTOFORMATTERARE

KÖR LINTERS OCH AUTOFORMATTERARE VID PR OCH
KRÄV ATT DE SKA GÅ IGENOM INNAN MERGE

TYPE HINTING

- Python är fortfarande dynamiskt typat
- Underlättar och dokumenterar för utvecklare och IDE
- `def prepare_plot(value: int, unit: str) -> dict`
- Statisk typkontroll med MyPy

- *typing* modulen ger fler möjligheter
- `typing.Union[str, int]`
- `typing.List[str]`
- `typing.Dict[str, set]`
- `typing.Iterable`
- `typing.Any`
- ...
- Klasser är också typer
- Använd strängar för att annotera om typen inte är tillgänglig

- `Literal` definerar unika värden

```
def print_length(length, unit=typing.Literal["m", "ft"])
```

- `Final` för värden som inte *bör* definieras om

```
user_id: typing.Final[int] = 24
```

- `@typing.final` för klasser och metoder som inte *bör* ärvas/ändras av subklasser

```
class OpKoKo(typing.TypedDict):  
    edition: str  
    start_date: datetime.date  
  
current = OpKoKo(edition="19.2", date=datetime.date(2019, 11, 8))  
# ...  
def printable_program(current: OpKoKo) -> str:
```

- Bättre kontroll av typer i en dict
- Objektet blir en vanlig dict
- Jämför: named tuple eller dataclasses

- Static ducktyping
- Specifierar metoder/attribut objektet ska ha

```
class ChessPiece(typing.Protocol):  
    name: str  
    def move(self) -> None:  
        pass  
  
def play(piece: ChessPiece, x: int, y: int) -> None:  
    piece.move(x, y)
```


TESTING

- Fantastiskt testramverk, bättre än Unittest från standardbiblioteket
- Läs Brian Okkens *Python Testing with pytest* (Pragmatic bookshelf)
- Fixturer, plugins, rena asserts, ...
- `assert data is not None`

TESTTÄCKNING MED COVERAGE

- Plugin till pytest: `pytest --cov=todo-extractor`
- Mäter testtäckning
- Genererar rapporter för människor och maskiner
HTML, XML, ...

Name	Stmts	Miss	Cover
------	-------	------	-------

-------	--	--	--

<code>todo_extractor.py</code>	22	7	68
--------------------------------	----	---	----

PAKETERA SOM MODUL

- Finns många sätt men jag rekommenderar Flit
- Starta med `flit init`
- Konfigureras i `pyproject.toml`
PEP 518 – *Specifying minimum build system requirements for Python projects*

- Stödjer inte `requirements.txt` → flytta beroenden
- Installera med symlänk/pth-filer för TDD

- Testa mot `test.pypi.org`
- Konfigurera egna servrar i `/.pypirc`
- `flit --repository testpypi publish`

Installera modulen med `pipx install todo-extractor`

AUTOMATISERA TESTER

AUTOMATISERA MED PRE-COMMIT

- Ramverk för att hantera git pre-commit hooks
- Skrivet i Python, men inte bara för Python
- Konfigureras med `.pre-commit` YAML fil
- Måste manuellt installeras, ersätter inte ci
- Kör linters, autoformattera JSON, ...


- Automatisera tester med definerade testfall
- Unika virtualenv för tester – skilj från utvecklingsvenv
- Möjliggör test i flera Python-versioner
- Passar utmärkt att köra i CI

PROJEKTMALL MED COOKIECUTTER


- Ramverk för att skapa projekt från Jinja2-mallar
- Skrivet i Python, men inte bara för Python
- Kan skapa upp hela mapp- och filstrukturen

/vikahl/python-template

SLUT, FRÅGOR?

/VIKAHL/MODERN-PYTHON-DEVELOPMENT

/VIKAHL/TODO-EXTRACTOR

/VIKAHL/PYTHON-TEMPLATE

MODERN PYTHONUTVECKLING

Viktor Ahlqvist

OpKoKo 19.2

 vikahl/modern-python-development

- Python bytes (podcast)
- Test and code (podcast)
Speciellt avsnitt 80 *From Python script to maintainable package*
och 81 *TDD with Flit*
- Talk Python to me (podcast)
- Real Python (realpython.com)
- docs.python.org