exactpro
EXITUS ACTA PROBAT

Python Configuration Management

2023.10.20 – Viacheslav Ermakov

BUILD SOFTWARE TO TEST SOFTWARE
exactpro.com

Who is this presentation for?

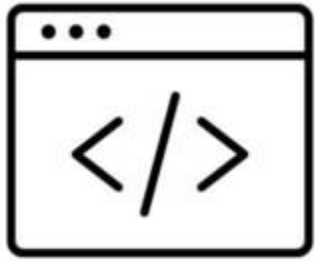
Who is this presentation for?

exactpro
EXITUS ACTA PROBAT

1. For python beginners
2. For people who write some scripts from time to time and dealt with configuration scripts
3. For people who just want to have some guide and configuration templates (some boilerplate code)

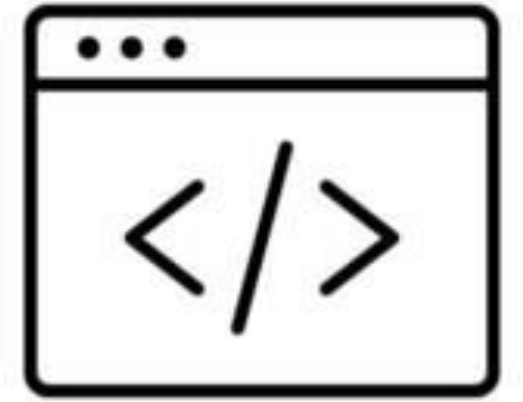
Problem description

Problem description



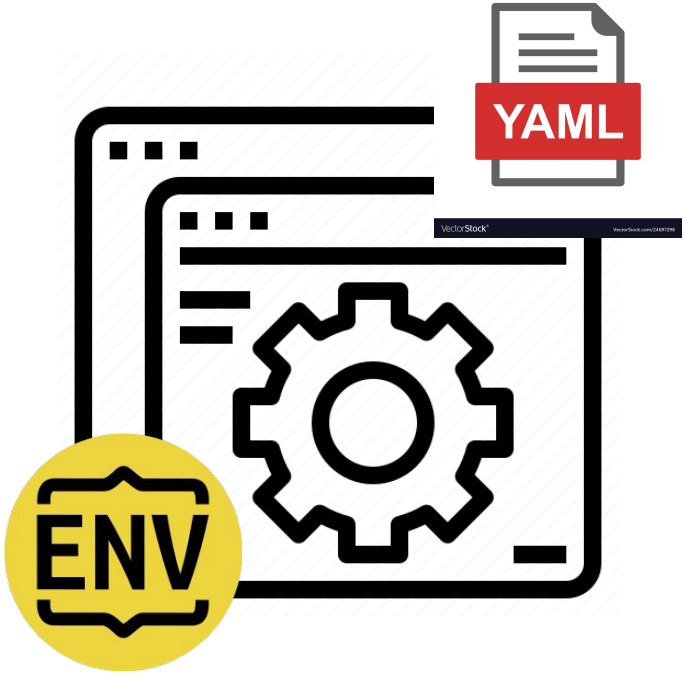
We are writing some kind of script.

Problem description

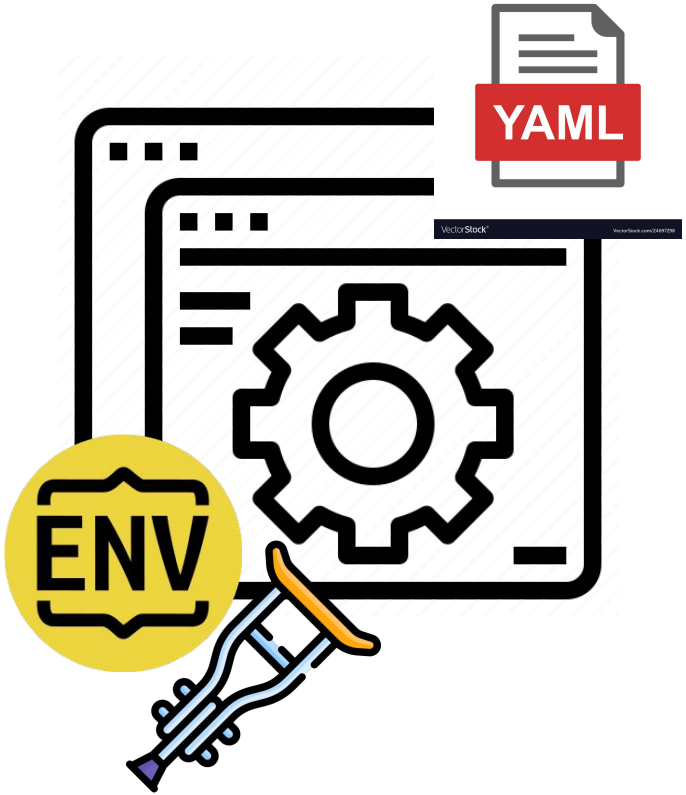


Then it gets bigger, and we need to somehow configure it without changing the code.

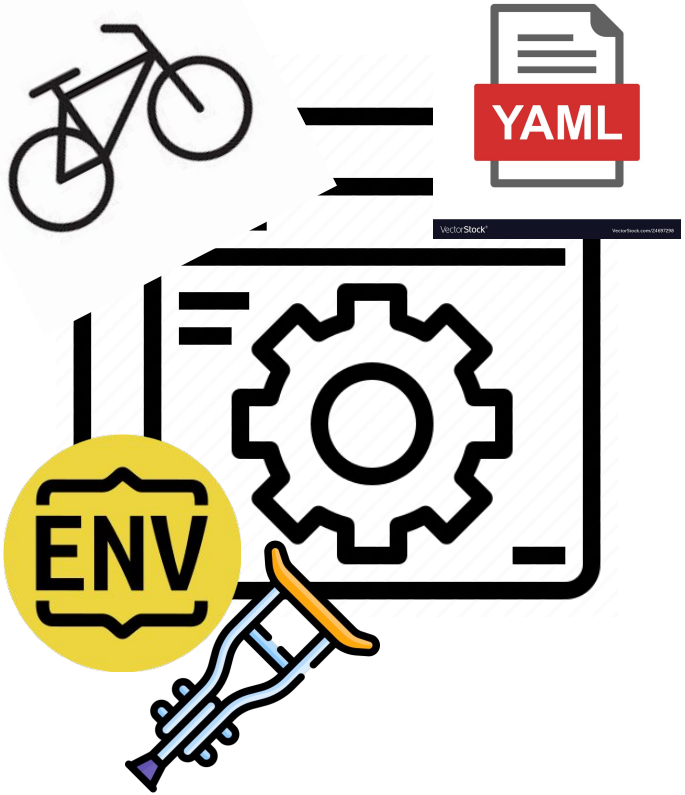
Problem description



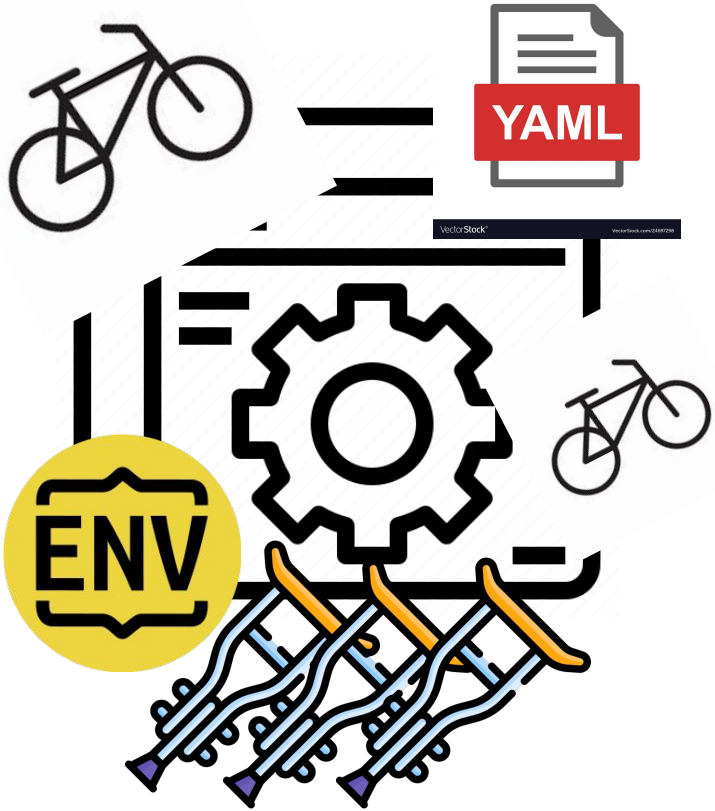
Problem description



Problem description



Problem description



Sounds familiar?

Are you familiar with this problem? Do you know how to solve it?

1. no, I have no idea
2. yes, I don’t know
3. yes, I wrote my own configuration manager
4. yes, I know the secret knowledge of how to solve this

Existing solutions

Existing solutions

exactpro
EXITUS ACTA PROBAT

1. **Configparser** – https://docs.python.org/3/library/configparser.html – python built-in package for INI configs
2. **PyYAML** – https://github.com/yaml/pyyaml – canonical tool for working with Yaml files
3. **Dynaconf** – https://www.dynaconf.com/ – when you have very dynamic configuration, when user provide nested structures or if you want to have fast start
4. **Pydantic-settings** – https://docs.pydantic.dev/latest/concepts/pydantic_settings/ – when you want to have strict variable names and configuration parameters hints in IDE.
5. **Hydra** – https://hydra.cc/ – Facebook framework to build CLI tools. It solves configuration problem inside. Configuration management API can be used separately
6. **oslo.config** – https://docs.openstack.org/oslo.config/latest/ – when you want to use INI files + CLI
7. **OmegaConf** – https://github.com/omry/omegaconf

What to choose?

Recommendations

What to choose?

What's the best solution?

There is no universal solution

Who’s that program?

??

???

?

Short scripts

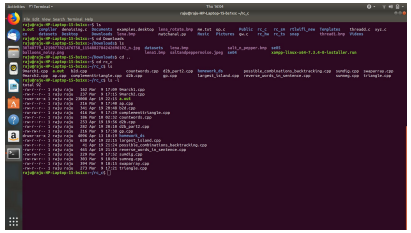
Short scripts
up to 100 lines

???

?

Short scripts

Short scripts
up to 100 lines



with configuration via CLI

without configuration via CLI

Short scripts

https://github.com/ConnectDIY/python-best-practices/tree/master/config_management/simple_scripts_configuration

exactpro
EXITUS ACTA PROBAT

pretty_print_json_cli.py

```python
import sys
import json

"""..."""

if __name__ == '__main__':
    args = sys.argv[1:]  # 0 item is a script name.

    if not args:
        print("Json string isn't provided")
        exit(-1)

    json_str = args[0]

    json_object = json.loads(json_str)
    json_formatted_str = json.dumps(json_object, indent=2)
    print(json_formatted_str)
```

Use **sys.argv** to get CLI arguments

with CLI

backup.py

```python
from zipfile import ZipFile

COMPRESSED_FILENAME = 'backup.zip'
LIST_OF_FILES_TO_COMPRESS = [
    'pretty_print_json_cli.py',
]

if __name__ == '__main__':
    with ZipFile(COMPRESSED_FILENAME, 'w') as myzip:
        for fname in LIST_OF_FILES_TO_COMPRESS:
            myzip.write(fname)
```

Use **Constants** to configure simple scripts

without CLI

Who’s that program?

Short scripts
up to 100 lines

???

?

Scripts with config files

Short scripts
up to 100 lines

Some scripts for automation, but still not very large

?

Scripts with config files

Some scripts for automation, but still not very large

with CLI / without CLI

Configuration file

DYNAMIC PARAMS

With a large number of dynamic parameters in the configuration

STATIC PARAMS

When 90% of the parameters will have a simple value like a string or a number. In general, anything except nested structures such as dictionaries.
When it's expected to update the script from time to time

Scripts with config files

exactpro
EXITUS ACTA PROBAT

DynaConf is good for

- DYNAMIC PARAMS
- for yaml, toml, json, and other configuration file types
- for fast start

Pydantic-settings is good for

- STATIC PARAMS (provides type hints in IDE)
- for .py and .env config files
- takes time to describe configuration
- better for long-supported programs

Scripts with config files

exactpro
EXITUS ACTA PROBAT

```python
from dynaconf import Dynaconf

settings = Dynaconf(
    ...
    envvar_prefix="A_",
    ...
    load_dotenv=True,
    # You cannot read .env files here as in pydantic,
    settings_files=['../config.yaml', 'secrets.yaml']
)
```

config.yaml

```yaml
# Used by DynaConf
ENV: DEV
NAME: global-YAML-file
PORT: 5050
```

```python
from pprint import pprint

from config import settings

if __name__ == '__main__':

    print(f"========= Run with {settings.ENV} configuration =========")

    # [1] Update the config.
    settings.A = 2
    settings.NESTED[12] = {'s': {'12-l2': 'l3-value'}}

    # [2] Getting value
    x = settings.NESTED.get(12)

    # [3] Pretty print of the config.
    print('config file:')
    pprint(settings.to_dict())  # Please note, it prints secrets!
```

DynaConf
DYNAMIC PARAMS

Scripts with config files

exactpro
EXITUS ACTA PROBAT

```python
class Secrets(BaseModel):
    """Just SubModel example."""
    SECRET_KEY: str = "pwd-in-pydantic-cfg"
    PASSWORD: str
    TOKEN: str

class Base(BaseSettings):
    """This class automatically loads and overwrite ENV variables...."""
    ...
    model_config = SettingsConfigDict(
        env_nested_delimiter='.',
        ...
        env_file=('../.env', '../.secrets', '.env.prod', '.secrets.prod'),
        ...
    )

    ENV: str
    PORT: int = 5050
    USERNAME: str = "Slavik"
    NAME: str  # Required parameter without default value.
               ...
    NESTED: Dict[L1, Dict[L2, Dict[L3, Any]]] = {...}

    SECRETS: Secrets
```

```python
from pprint import pprint
from pydantic_config import settings, L2, L3


if __name__ == '__main__':

    print(f"========= Run with {settings.ENV} configuration =========")

    # [1] Update the config.
    settings.NESTED[12] = {'s': {L3('12-l2'): 'l3-value'}}

    # [2] Getting value
    x = settings.NESTED.get(12)

    # [3] Pretty print of the config.
    # Here Pydantic will print UserWarning
    #   `Expected `str` but got `int` - serialized value may not be as expected`
    # It happens because NESTED[ expected type is STRING ]
    print('config file:')
    pprint(settings.model_dump())  # Please note, it prints secrets!
```

Pydantic-settings
STATIC PARAMS

Who’s that program?

Short scripts
up to 100 lines

Some scripts for
automation, but still not
very large

?

Large programs

Short scripts
up to 100 lines

Some scripts for automation, but still not very large

Large programs with many parameters, configuration files, CLI and more

Large programs

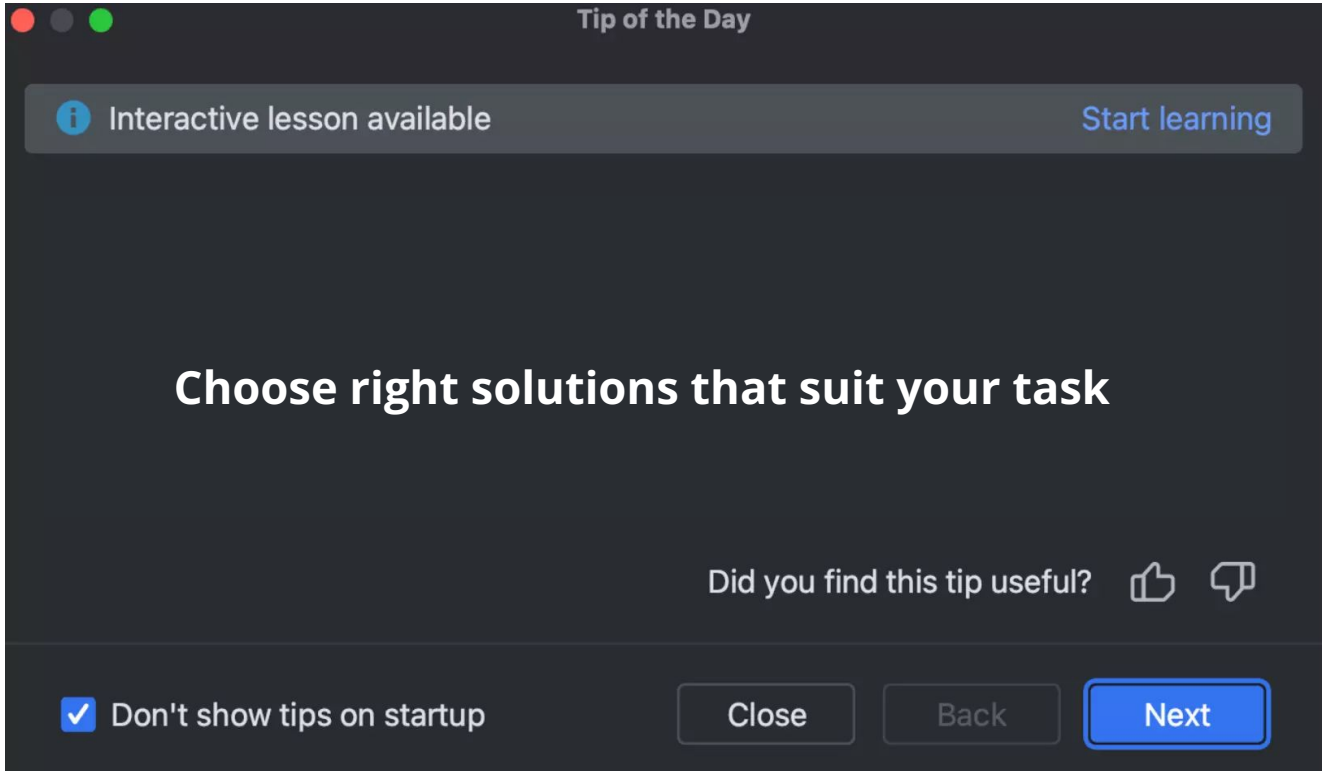
Large programs with many parameters, configuration files, CLI and more

I can only recommend to look at some ready famous frameworks like Hydra

It allows you to

- create a hierarchical configuration by composition and override it through config files and the command line.
- to create CLI
- to run parallel jobs for different configuration
- configure logging
- …

Look more details on the official site


Tip of the Day
Interactive lesson available
Start learning
Choose right solutions that suit your task
Did you find this tip useful?
Don't show tips on startup
Close
Back
Next

Thank you! Questions?

Configuration templates can be found here:

https://github.com/ConnectDIY/python-best-practices/tree/master/config_management

exactpro

EXITUS ACTA PROBAT

Thank You!