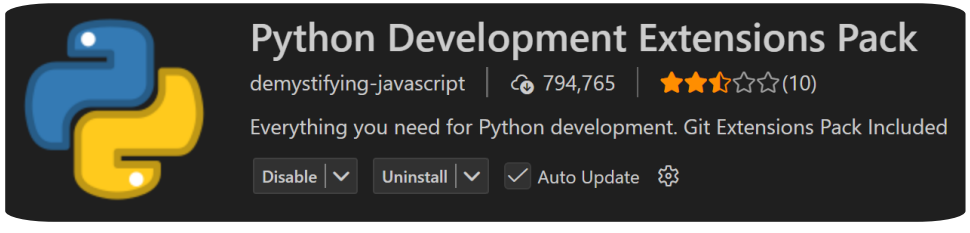
VIRTUÁLNÍ PROSTŘEDÍ V PYTHONU

Andrej Pčelovodov

Arvion

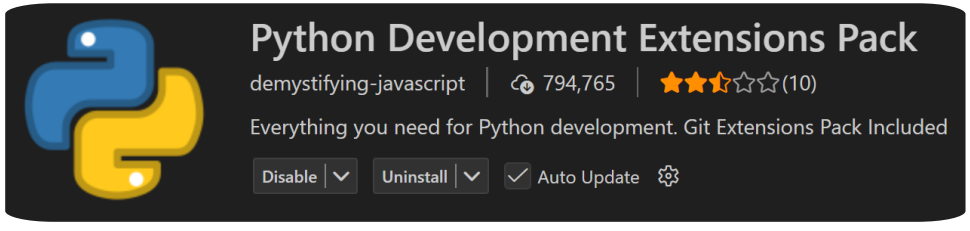
VISUAL STUDIO CODE



Andrej Pčelovodov

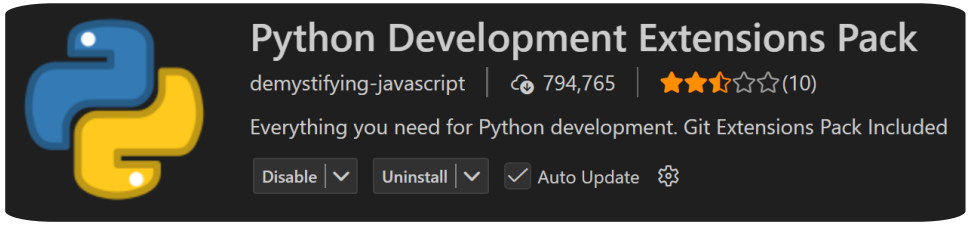
Arvion

VISUAL STUDIO CODE



1. Běžte na
   https://github.com/AndreiBee/Python-Tutorials

VISUAL STUDIO CODE



1. Běžte na
   https://github.com/AndreiBee/Python-Tutorials
2. Nainstalujte si:
   Visual Studio Code
   Python Development Extenstions Pack

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

🐍 PYTHON VENV

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

🐍 PYTHON VENV

- Každé dítě má **svůj vlastní prostor**

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

🐍 PYTHON VENV

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

🐍 PYTHON VENV

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

🐍 PYTHON VENV

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🐍 PYTHON VENV

- Každý projekt má **vlastní prostředí**

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🐍 PYTHON VENV

- Každý projekt má **vlastní prostředí**
- Knihovny se **nemíchají** s jinými projekty

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🐍 PYTHON VENV

- Každý projekt má **vlastní prostředí**
- Knihovny se **nemíchají** s jinými projekty
- Když se rozbije, **smažeš venv**

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🐍 PYTHON VENV

- Každý projekt má **vlastní prostředí**
- Knihovny se **nemíchají** s jinými projekty
- Když se rozbije, **smažeš venv**
- Neovlivní globální Python

🧪 VIRTUALNÍ PROSTŘEDÍ = PÍSKOVIŠTĚ

🏖️ PÍSKOVIŠTĚ

- Každé dítě má **svůj vlastní prostor**
- Hračky se **nemíchají** s ostatními
- Když se rozbije, **prostě ho zrušíš**
- Neovlivní ostatní děti

🐍 PYTHON VENV

- Každý projekt má **vlastní prostředí**
- Knihovny se **nemíchají** s jinými projekty
- Když se rozbije, **smažeš venv**
- Neovlivní globální Python

Andrej Pčelovodov

venv = bezpečné pískoviště pro experimenty i vývoj

PACKAGE MANAGERY

PACKAGE MANAGERY

Andrej Pčelovodov

Arvion

PACKAGE MANAGERY

Andrej Pčelovodov

Arvion

PACKAGE MANAGERY

ANACONDA®

Poetry

PACKAGE MANAGERY

PACKAGE MANAGERY

ANACONDA®

Poetry

Pro výuku a pochopení základů je důležitější **rozumět venv a pip** než používat nástroje, které všechno automatizují a schovávají.

🤵 PACKAGE MANAGERY

Pro výuku a pochopení základů je důležitější **rozumět venv a pip** než používat nástroje, které všechno automatizují a schovávají.

👉 **Nejdřív základy. Až pak nástroje.**

❌ PROČ NEPOUŽÍVÁME ANACONDU

❌ PROČ NEPOUŽÍVÁME ANACONDU

- 📦 **Zbytečně obrovská**
  Anaconda má několik GB a obsahuje stovky balíků, které nikdy nepoužijeme.

❌ PROČ NEPOUŽÍVÁME ANACONDU

- 📦 **Zbytečně obrovská**
  Anaconda má několik GB a obsahuje stovky balíků, které nikdy nepoužijeme.
- 🔀 **Conda ≠ pip**
  Používá vlastní balíčkovací systém → problémy s kompatibilitou a závislostmi.

❌ PROČ NEPOUŽÍVÁME ANACONDU

- 📦 **Zbytečně obrovská**
  Anaconda má několik GB a obsahuje stovky balíků, které nikdy nepoužijeme.
- 🔀 **Conda ≠ pip**
  Používá vlastní balíčkovací systém → problémy s kompatibilitou a závislostmi.
- 🔒 **Uzavírá tě do vlastního ekosystému**
  Mimo Anacondu je pak člověk ztracený (nezná venv, pip, pyproject.toml).

❌ PROČ NEPOUŽÍVÁME ANACONDU

❌ PROČ NEPOUŽÍVÁME ANACONDU

- ⚙️ **Špatná kompatibilita s praxí**
  CI/CD, Docker, produkční servery → všude se počítá s čistým Pythonem.

❌ PROČ NEPOUŽÍVÁME ANACONDU

- ⚙️ **Špatná kompatibilita s praxí**
  CI/CD, Docker, produkční servery → všude se počítá s čistým Pythonem.
- 📚 **Nevhodná pro výuku základů**
  Skrývá, jak Python doopravdy funguje.

❌ PROČ NEPOUŽÍVÁME ANACONDU

- ⚙️ **Špatná kompatibilita s praxí**
  CI/CD, Docker, produkční servery → všude se počítá s čistým Pythonem.
- 📚 **Nevhodná pro výuku základů**
  Skrývá, jak Python doopravdy funguje.
- 🕰️ **Historický důvod, který už neplatí**
  Dnes `pip install numpy` funguje bez problémů.

❌ PROČ NEPOUŽÍVÁME ANACONDU

- ⚙️ **Špatná kompatibilita s praxí**
  CI/CD, Docker, produkční servery → všude se počítá s čistým Pythonem.
- 📚 **Nevhodná pro výuku základů**
  Skrývá, jak Python doopravdy funguje.
- 🕰️ **Historický důvod, který už neplatí**
  Dnes `pip install numpy` funguje bez problémů.

👉 **Anaconda dává smysl hlavně pro data science, ne pro vývojáře.**

VIRTUÁLNÍ PROSTŘEDÍ V PYTHONU

Virtuální prostředí (venv) je izolované prostředí pro Python projekt.

VIRTUÁLNÍ PROSTŘEDÍ V PYTHONU

Virtuální prostředí (venv) je izolované prostředí pro Python projekt.

- Každý projekt má vlastní balíčky

Andrej Pčelovodov

Arvion

VIRTUÁLNÍ PROSTŘEDÍ V PYTHONU

Virtuální prostředí (venv) je izolované prostředí pro Python projekt.

- Každý projekt má vlastní balíčky
- Žádné konflikty verzí

VIRTUÁLNÍ PROSTŘEDÍ V PYTHONU

Virtuální prostředí (venv) je izolované prostředí pro Python projekt.

- Každý projekt má vlastní balíčky
- Žádné konflikty verzí
- Čistý a reprodukovatelný setup

❓ PROČ POUŽÍVAT VENV?

❓ PROČ POUŽÍVAT VENV?

- Různé projekty = různé verze knihoven

❓ PROČ POUŽÍVAT VENV?

- Různé projekty = různé verze knihoven
- Co funguje dnes, musí fungovat i za rok

❓ PROČ POUŽÍVAT VENV?

- Různé projekty = různé verze knihoven
- Co funguje dnes, musí fungovat i za rok
- Bez rozbití globální instalace Pythonu

❓ PROČ POUŽÍVAT VENV?

- Různé projekty = různé verze knihoven
- Co funguje dnes, musí fungovat i za rok
- Bez rozbití globální instalace Pythonu

👉 **Jedno pravidlo:** jeden projekt = jedno virtuální prostředí

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
1 pip install numpy
2 pip install tensorflow
3 pip install flask
```

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
pip install numpy
pip install tensorflow
pip install flask
```

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
pip install numpy
pip install tensorflow
pip install flask
```

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
pip install numpy
pip install tensorflow
pip install flask
```

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
pip install numpy
pip install tensorflow
pip install flask
```

- Všechno se instaluje globálně

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
1 pip install numpy
2 pip install tensorflow
3 pip install flask
```

- Všechno se instaluje globálně
- Konflikty verzí

BEZ VIRTUÁLNÍHO PROSTŘEDÍ

```
pip install numpy
pip install tensorflow
pip install flask
```

- Všechno se instaluje globálně
- Konflikty verzí
- „Na mém počítači to funguje“

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
1 cd D:/Path/To/Project
2 python -m venv .venv # .venv - název složky
```

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
cd D:/Path/To/Project
python -m venv .venv # .venv - název složky
```

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
1 cd D:/Path/To/Project
2 python -m venv .venv # .venv - název složky
```

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
1 cd D:/Path/To/Project
2 python -m venv .venv # .venv - název složky
```

- ".venv" je název složky

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
1 cd D:/Path/To/Project
2 python -m venv .venv # .venv - název složky
```

- ".venv" je název složky
- Vznikne izolované prostředí

🛠️ KROK 1: VYTVOŘENÍ VENV

Otevřte PowerShell ve Windows (Terminal v MacOS/Linux)

```
1 cd D:/Path/To/Project
2 python -m venv .venv # .venv - název složky
```

- ".venv" je název složky
- Vznikne izolované prostředí
- Žádné balíčky kromě základních

KROK 2: AKTIVACE

WINDOWS:

```
1  .venv\Scripts\activate
```

LINUX / MACOS:

```
1  source .venv/bin/activate
```

▶️ KROK 2: AKTIVACE

WINDOWS:

```
1  .venv\Scripts\activate
```

LINUX / MACOS:

```
1  source .venv/bin/activate
```

Po aktivaci vidíš (**.venv**) v terminálu

📦 KROK 3: INSTALACE A OVĚŘENÍ BALÍČKŮ

```
pip install requests
pip install numpy
pip list
```

📦 KROK 3: INSTALACE A OVĚŘENÍ BALÍČKŮ

```
pip install requests
pip install numpy
pip list
```

📦 KROK 3: INSTALACE A OVĚŘENÍ BALÍČKŮ

```
pip install requests
pip install numpy
pip list
```

📦 KROK 3: INSTALACE A OVĚŘENÍ BALÍČKŮ

```
pip install requests
pip install numpy
pip list
```

- Instaluje se jen do .venv

📦 KROK 3: INSTALACE A OVĚŘENÍ BALÍČKŮ

```
pip install requests
pip install numpy
pip list
```

- Instaluje se jen do .venv
- Globální Python zůstává čistý!

JEDNODUCHÝ PŘÍKLAD

Projekt A potřebuje:

```
numpy==1.21
```

Projekt B potřebuje:

```
numpy==2.0
```

🧠 JEDNODUCHÝ PŘÍKLAD

Projekt A potřebuje:

```
numpy==1.21
```

Projekt B potřebuje:

```
numpy==2.0
```

👉 Bez venv **problém**, s venv **žádný stres**

REQUIREMENTS.TXT

```
1 pip freeze > requirements.txt
```

```
1 pip install -r requirements.txt
```

REQUIREMENTS.TXT

```
1 pip freeze > requirements.txt
```

```
1 pip install -r requirements.txt
```

- Přesně stejné prostředí

REQUIREMENTS.TXT

```
1 pip freeze > requirements.txt
```

```
1 pip install -r requirements.txt
```

- Přesně stejné prostředí
- Ideální pro tým i produkci

DEAKTIVACE

```
1 deactivate
```

DEAKTIVACE

```
1 deactivate
```

Vracíš se zpět do globálního Pythonu

✅ SHRNUTÍ

✅ SHRNUTÍ

- Virtuální prostředí je základ

✅ SHRNUTÍ

- Virtuální prostředí je základ
- Používá se vždy, bez výjimek

✅ SHRNUTÍ

- Virtuální prostředí je základ
- Používá se vždy, bez výjimek
- Šetří nervy, čas a rozbité projekty

✅ SHRNUTÍ

- Virtuální prostředí je základ
- Používá se vždy, bez výjimek
- Šetří nervy, čas a rozbité projekty

“Co není izolované,
to se dřív nebo později rozbije. ”