ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Εισαγωγή στην Python

Π. Μόσχος

Python

- Η Python είναι μια γλώσσα προγραμματισμού **γενικής χρήσης** που μπορεί να χρησιμοποιηθεί για desktop applications, web applications, κ.α.
- Είναι δημοφιλής ωστόσο σε εφαρμογές data science και artificial intelligence (AI)
- Αναπτύχθηκε από τον Guido Van Rossum το 1989 ενώ εργαζόταν στο Εθνικό Ινστιτούτο Ερευνών της Ολλανδίας
- Η πρώτη έκδοση της Python κυκλοφόρησε το 20 Φεβρουαρίου 1991

Χαρακτηριστικά της Python (1)

- **Δωρεάν** λογισμικό **ανοικτού κώδικα**
- Γλώσσα προγραμματισμού **υψηλού επιπέδου** (διαχείριση μνήμης, ασφάλεια κτλ.). Είναι **Functional** και **Object Oriented**.
- **Διερμηνεύεται** (δεν παράγεται εκτελέσιμο αρχείο) / **Platform independent**.
- Παρέχει **ενσωμάτωση** (χρήση άλλων γλωσσών προγραμματισμού στην Python).
- Διαθέτει εξαιρετικά **πολλές** «**βιβλιοθήκες**».

Χαρακτηριστικά της Python (2)

- **Είναι weakly-typed.** Δεν δηλώνουμε τύπους μεταβλητών
- Εφαρμόζει **Dynamic policy.** Συμπεραίνει τους τύπους δεδομένων @runtime από τις τιμές που εκχωρούμε στις μεταβλητές

Εκδόσεις της Python

- Οι βασικές εκδόσεις της **Python** είναι οι:
- Python 1.0 (1994)
- Python 2.0 (2000)
- **Python 3.0** (2008, τρέχουσα έκδοση)
- Αποτελεί εξαίρεση στο ότι δεν υπάρχει πλήρης προς τα πίσω συμβατότητα. Τα προγράμματα σε Python 2.7 δεν είναι πλήρως συμβατά σε Python 3

Υλοποιήσεις της Python

- Υπάρχουν διάφορες υλοποιήσεις της γλώσσας προγραμματισμού Python. Οι πιο γνωστές είναι:
  - **CPython** (Reference Implementation)
  - Jython
  - IronPython
  - PyPy
  - MicroPython
- Κάθε υλοποίηση έχει τα δικά της δυνατά και αδύνατα σημεία (πλεονεκτήματα και μειονεκτήματα) και συνεπώς μπορεί είναι κατάλληλη για διαφορετικές περιπτώσεις χρήσης

CPython

- Είναι η **υλοποίηση αναφοράς** της Python. Χρησιμοποιείται από το Python Software Foundation
- https://www.python.org/psf-landing/
- Είναι **γραμμένη σε C** και είναι η πιο ευρέως χρησιμοποιούμενη υλοποίηση. Είναι η προεπιλεγμένη υλοποίηση που **συνοδεύει την επίσημη διανομή Python** (www.python.org).\ και περιλαμβάνεται στα Linux-bases συστήματα

Άλλες Υλοποιήσεις της Python

- **Jython.** Πρόκειται για μια υλοποίηση της Python που εκτελείται στην Εικονική Μηχανή Java (JVM) και επιτρέπει στον κώδικα Python να ενσωματωθεί με κώδικα Java
- **IronPython.** Αυτή είναι μια υλοποίηση της Python που εκτελείται στο .NET Framework και επιτρέπει στον κώδικα Python να ενσωματωθεί με κώδικα .NET.
- **PyPy.** Αυτή είναι μια υλοποίηση της Python που χρησιμοποιεί Just-In-Time (JIT) compilation για τη βελτίωση της απόδοσης. Σε κάποιες περιπτώσεις η PyPy μπορεί να είναι πολύ πιο γρήγορη από το CPython.
- **MicroPython.** Είναι μια υλοποίηση της Python που έχει σχεδιαστεί για να εκτελείται σε συσκευές οι οποίες διαθέτουν περιορισμένους υπολογιστικούς πόρους

C, Java, Python

- Παρακάτω παρουσιάζονται τρία παραδείγματα για την εκτύπωση στην κονσόλα του μηνύματος "Hello World" στις γλώσσες προγραμματισμού **C**, **Java** και **Python**.

C

```c
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}

```

Java

```java
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Python

```python
print("Hello World")
```

Python – Standard Library

- Η Standard library της Python είναι μια συλλογή από έτοιμα προγράμματα τα οποία παρέχουν ένα ευρύ φάσμα λειτουργιών στη γλώσσα, όπως:
  - χειρισμός αρχείων
  - προγραμματισμός δικτύου
  - κανονικές εκφράσεις, κ.α.

Εγκαταστάσεις Python

- Η εγκατάσταση της Python (που ήδη έχουμε κάνει) απλά αντιγράφει τα (binary) αρχεία της Python σε ένα folder/directory στον υπολογιστή μας
- Επομένως, **είναι δυνατόν να έχουμε «εγκατεστημένες» στον υπολογιστή μας πολλαπλές εκδόσεις της Python**, αρκεί να έχουμε τοποθετήσει τις διαφορετικές εκδόσεις σε διαφορετικά directories και να καλούμε κάθε φορά το επιθυμητό εκτελέσιμο αρχείο της Python μέσα από το directory / folder

Python – Χαρακτηριστικά (1)

- **Μοντέλο εκτέλεσης κώδικα.** Η Python μεταγλωττίζει τον πηγαίο **κώδικα** σε **bytecode** ο οποίος στη συνέχεια **διερμηνεύεται (εκτελείται)** από τον διερμηνευτή (Python Virtual Machine).
- **Διαχείριση της μνήμης.** Η διαχείριση μνήμης στην Python περιλαμβάνει Stack και Heap. Όλες οι μεταβλητές (όλων των τύπων, primitives και σύνθετων τύπων) είναι references στο Heap. Στο Stack εκτελούνται οι συναρτήσεις / μέθοδοι οργανωμένες σε stack frames
- Η διαχείριση του Heap τυπικά περιλαμβάνει ένα Automatic Garbage Collection

Δυναμικοί Τύποι Δεδομένων

- Όπως αναφέραμε οι δηλώσεις μεταβλητών στην Python δεν περιλαμβάνουν τον τύπο δεδομένων, ενώ ο τύπος δεδομένων αποφασίζεται με βάση την τιμή που έχουμε εκχωρήσει στην μεταβλητή (όπως στην JavaScript)
- Το όνομα της μεταβλητής αποθηκεύεται στο Stack ενώ το περιεχόμενο στο Heap. Επομένως τα **ονόματα όλων των μεταβλητών είναι δείκτες (references) προς τις αντίστοιχες θέσεις στο Heap**

Dynamic Typing

- Το γεγονός ότι γίνεται δυναμική δέσμευση μνήμης (@runtime) χωρίς δήλωση τύπων @compile time, το διαχειρίζεται η Python ΄κουβαλώντας΄ στο Heap εκτός από την τιμή, και τον τύπο της μεταβλητής, ώστε να μπορεί να διενεργεί type checking κατά την αποτίμηση παραστάσεων (expressions)

Python IDEs

- Υπάρχουν διάφορα **IDE** (**I**ntegrated **D**evelopment **E**nvironment) που μπορούν να χρησιμοποιηθούν για τη συγγραφή κώδικα python. Μερικά από τα πιο γνωστά είναι τα παρακάτω:
    - **Visual Studio Code (VSC).** Έχει το πλεονέκτημα ότι εκτελείται τόσο σε Windows όσο και σε macOS και Linux-based Συστήματα (επίσης είναι δωρεάν)
    - **PyCharm** (Jet Brains – Free & Paid Versions)
    - **Spyder** (Περιλαμβάνεται στο **Anaconda** που είναι μία υλοποίηση της Python ειδικά για Data Science μιας και περιλαμβάνει πολλές αντίστοιχες βιβλιοθήκες)

IDE

- Για την εκτέλεση των παραδειγμάτων του παρόντος, μπορείτε να χρησιμοποιήσετε οποιοδήποτε IDE σας εξυπηρετεί
- Στα παραδείγματα του παρόντος θα χρησιμοποιήσουμε VSC (Visual Studio Code) το οποίο μπορεί να εγκατασταθεί σε όλα τα Λειτουργικά Συστήματα

Python Projects

- Κάθε **Python project** τυπικά αντιστοιχεί σε ένα **φάκελο**
- Για να είναι το project μας ανεξάρτητο (isolated) από την κεντρική εγκατάσταση της Python, δημιουργούμε ένα **virtual environment,** που τεχνικά είναι ένας **υποφάκελος** στο root directory **του project**
- **Το Virtual Environment** περιέχει μία συγκεκριμένη version της Python στο συγκεκριμένο project καθώς και τα εγκατεστημένα dependencies

Δημιουργία Virtual Environment (1)

- Προκειμένου να δημιουργήσουμε ένα Virtual Environment στην Python με τη χρήση της default version της Python πρέπει να τρέξουμε την παρακάτω εντολή **μέσα στο φάκελο του project μας**:
- ***python -m venv venv***
- Το **-m venv** σημαίνει να χρησιμοποιηθεί το **m**odule venv ενώ το επόμενο venv είναι το όνομα του virtual environment (το όνομα του υποφακέλου) που μπορεί να είναι οτιδήποτε αλλά κατά σύμβαση (by convention) είναι **venv**

Δημιουργία Virtual Environment (2)

- Η δημιουργία του Virtual Environment τυπικά αντιγράφει το περιβάλλον (αρχεία) του μεταγλωττιστή / διερμηνευτή μαζί με το Python Standard Library μέσα στον υποφάκελο venv
- Στη συνέχεια κάνουμε **activate** το virtual environment ώστε τυχόν Python packages που θα εγκαταστήσουμε, να εγκατασταθούν μόνο στο project μας και όχι στην κεντρική εγκατάσταση, καθώς και για να εκτελέσουμε το project μας

Δημιουργία με συγκεκριμένη Python version

- Αν θέλουμε να δημιουργήσουμε ένα virtual environment με version διαφορετική από τη default μπορούμε να αρχικοποιήσουμε το virtual environment
- <python version> -m venv **venv** (σε Mac/Linux)

  π.χ. python3.10 –m venv venv
- <python path> -m venv **venv** (σε Windows)

  π.χ. C:\Program Files\Python312\python –m venv venv

Activate Virtual Environment (1)

- Όπως αναφέραμε, αν θέλουμε να εγκαταστήσουμε packages σε ένα project, τότε θα θέλαμε να μην επηρεαστεί η κεντρική εγκατάσταση, αλλά μόνο το project μας. Τότε κάνουμε **activate το venv**
- Όταν ενεργοποιούμε (**activate**) ένα virtual environment ουσιαστικά τροποποιείται το Path environmental variable του τρέχοντος Shell ώστε να χρησιμοποιείται, όχι ο interpreter της κεντρικής εγκατάστασης, αλλά το copy του venv.

Activate Virtual Environment (2)

- Αφού ενεργοποιηθεί το venv, στη συνέχεια, οποιαδήποτε εντολή python τρέξουμε θα βασίζεται στην έκδοση της python που έχουμε εγκαταστήσει στο συγκεκριμένο Virtual Environment
- Το ίδιο ισχύει και για τα packages. Θα είναι isolated από την κεντρική ή άλλες εγκαταστάσεις

Activate venv

- Στα **Mac/Linux**:
- **source venv/bin/activate** ή
- **. venv/bin/activate**

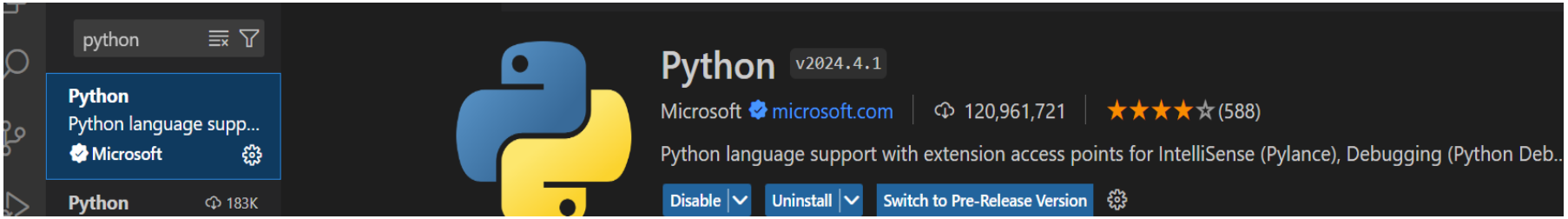
Τεχνικά η εντολή source (και το dot) ενεργοποιούν το venv στο τρέχον Shell

- Στα **Windows**:
- **venv\Scripts\activate** (και εκτελείται το activate.bat μέσα στον φάκελο Scripts του φακέλου venv)

Deactivate

- Το αντίθετο κάνει η deactivate. Επαναφέρει το περιβάλλον στην κεντρική εγκατάσταση
- Linux-based και macOS
  - **source venv/bin/deactivate**
- Windows
  - **venv\Scripts\deactivate**

Python – VSCode



- Μπορούμε να εγκαταστήσουμε το **Python extension** που μας δίνει syntax highlighting, code auto-completion και μαζί εγκαθίσταται και το Pylance που δίνει type checking (τυπικά το Pyright που είναι μέρος του Pylance)
- Το extension δεν μας παρέχει τον interpreter (ο interpreter είναι αυτός που έχουμε εγκαταστήσει)

Python – VSCode

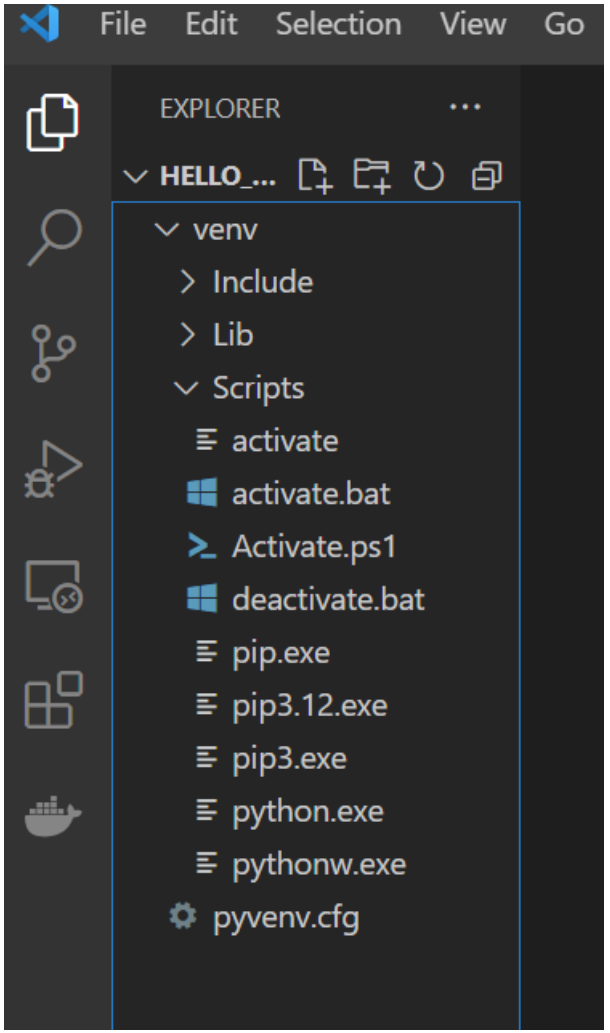
- Ένα project στην Python, όπως και σε άλλες γλώσσες προγραμματισμού, **αντιστοιχεί σε ένα φάκελο**
- Δημιουργούμε ένα φάκελο με όνομα hello_world ο οποίος θα αποτελέσει το project μας

```
a8ana@thanassis-pc MINGW64 ~/OneDrive/CodingFactor
IONAL-MATERIAL/Python/Python-projects/hello_world
```

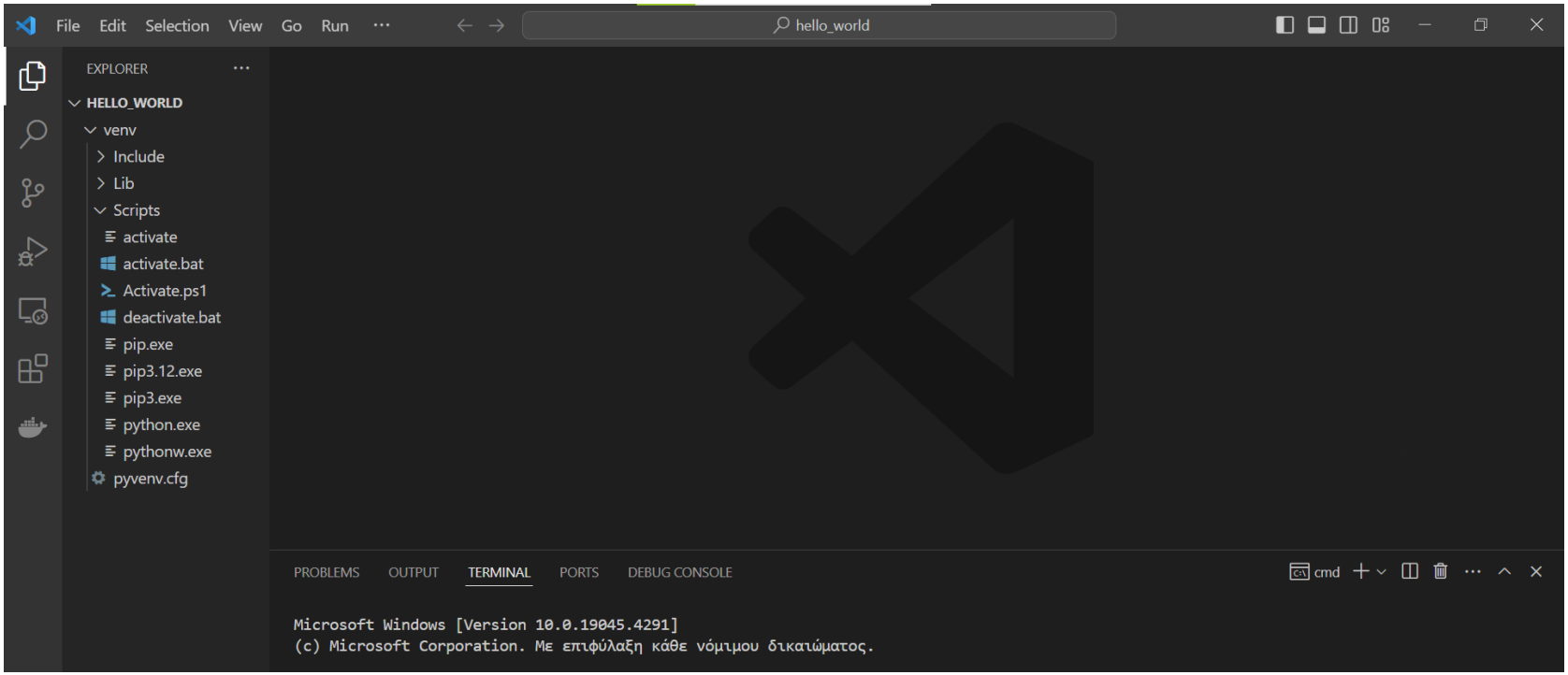
Python – VSCode

Μέσα από το VSCode επιλέγουμε:

1. **File**
2. **Open Folder…**
3. **hello_world** (open)

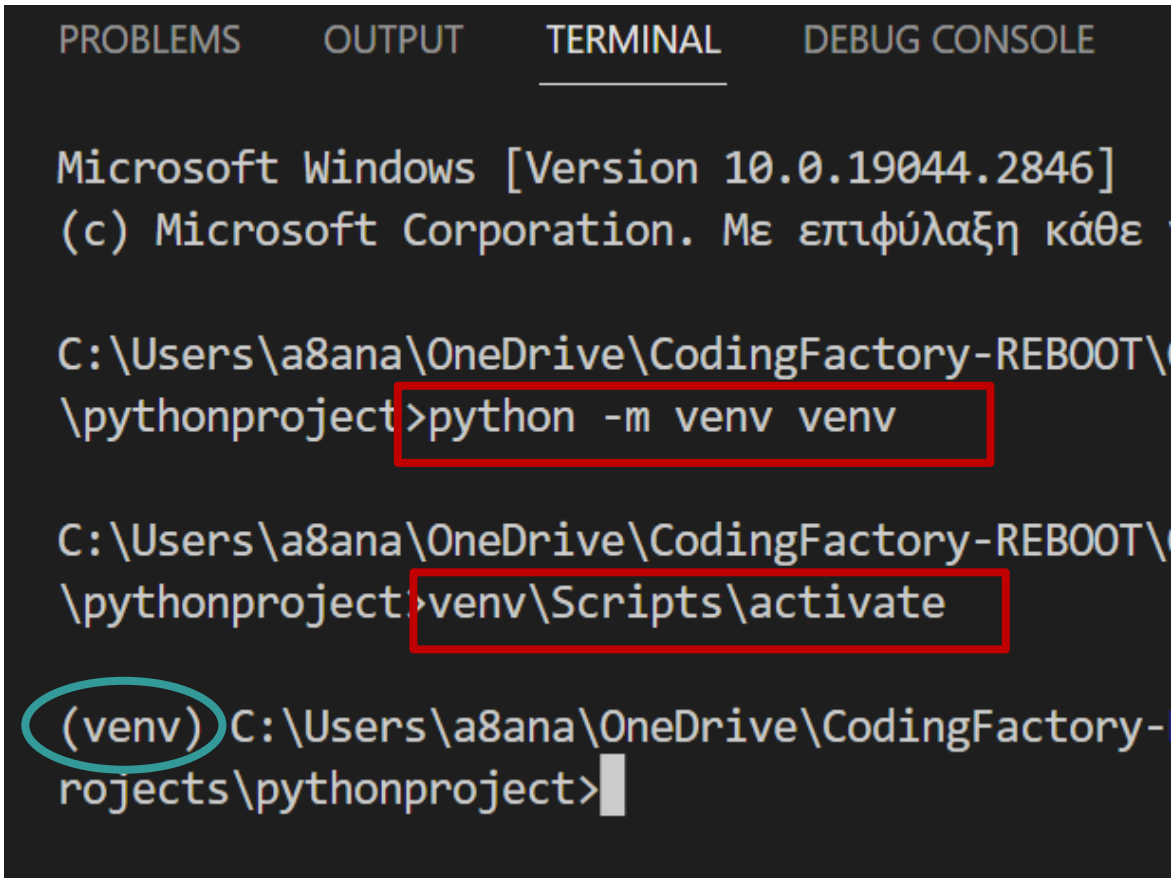


Terminal



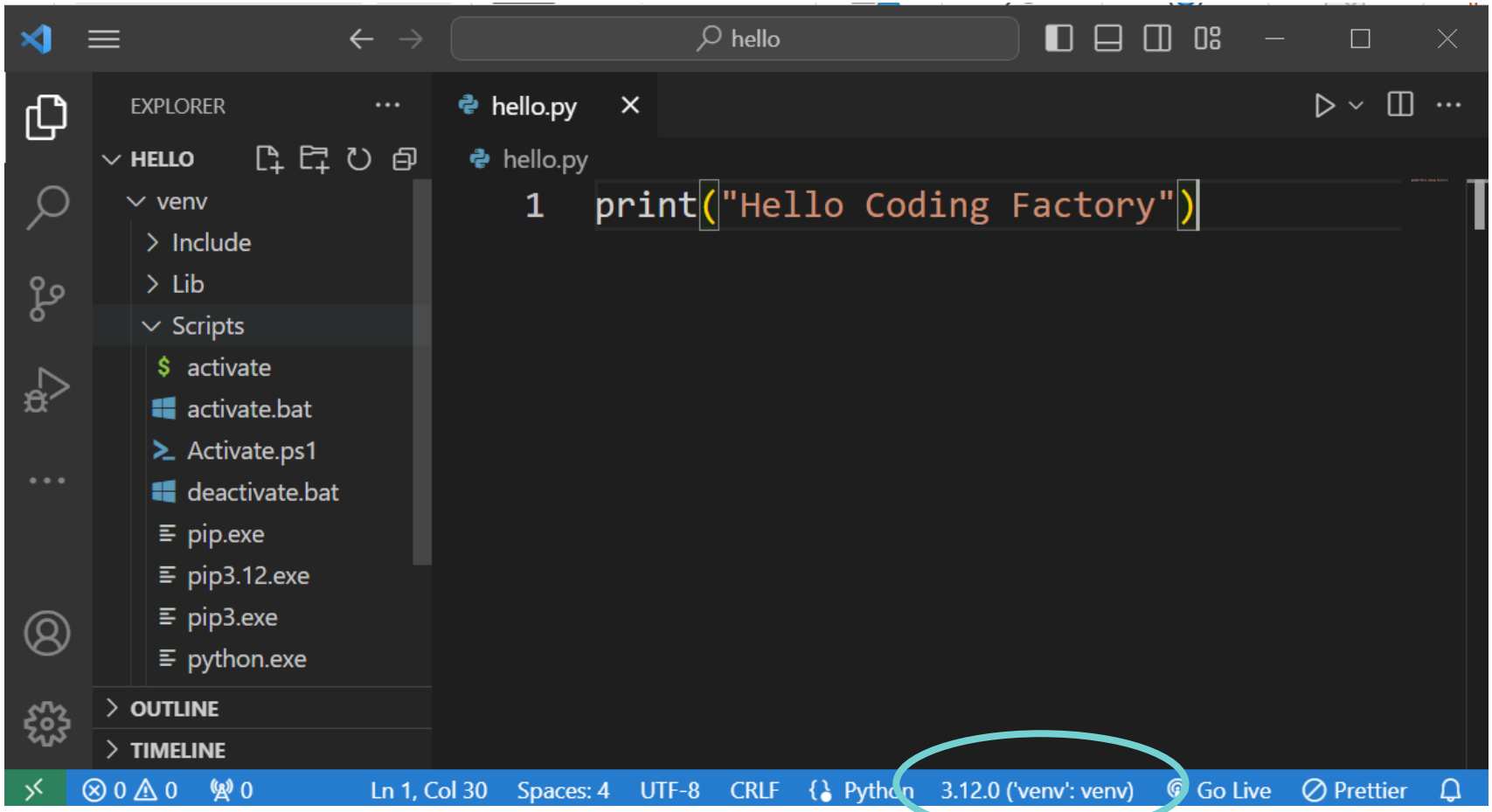
- Ανοίγουμε ένα νέο terminal (Command Prompt, όχι powershell γιατί έχει περιορισμούς ασφάλειας)

venv & activate



- Με **python –m venv venv** δημιουργούμε ένα virtual environment (υποφάκελο venv) και το ενεργοποιούμε

Python – hello.py



Εκτέλεση

- Εκτελούμε με python hello.py

```
C:\Users\a8ana\OneDrive\CodingFactory-REBOOT\cf-projects\EDUCATIONAL-MATER
IAL\Python\Python-projects\hello_world>venv\Scripts\activate

(venv) C:\Users\a8ana\OneDrive\CodingFactory-REBOOT\cf-projects\EDUCATIONA
L-MATERIAL\Python\Python-projects\hello_world>python hello.py
Hello Coding Factory

(venv) C:\Users\a8ana\OneDrive\CodingFactory-REBOOT\cf-projects\EDUCATIONA
L-MATERIAL\Python\Python-projects\hello_world>
```

Εγκατάσταση Dependencies (1)


```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

(venv) C:\Users\a8ana\OneDrive\CodingFactory-REBOOT\cf-projects\EDUCATIONAL-MATE
ject>pip install numpy
Collecting numpy
  Downloading numpy-1.24.2-cp311-cp311-win_amd64.whl (14.8 MB)
     ---------------------------------------- 14.8/14.8 MB 11.9 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.24.2

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) C:\Users\a8ana\OneDrive\CodingFactory-REBOOT\cf-projects\EDUCATIONAL-MATE
ject>
```

- Εγκαθιστούμε πακέτα με *pip install package_name* αφού έχουμε κάνει activate

Εγκατάσταση Dependencies (2)

- Για να εγκαταστήσουμε συγκεκριμένη version ενός dependency, το ορίζουμε στην γραμμή του install με ==<version-number>

```
(venv) C:\Users\a8ana\OneDrive\CodingFactory-
ject>pip install numpy==1.24.2
Requirement already satisfied: numpy==1.24.2
tional-material\python\projects\pythonproject
```

Requirements.txt

- Το αρχείο **requirements.txt** είναι το αρχείο που περιέχει μια λίστα με όλες τις εξαρτήσεις (dependencies) που έχουμε εγκαταστήσει στο project μας
- Για να δημιουργήσουμε το requirements.txt εκτελούμε, μέσα στον root folder, την εντολή:
- **pip freeze > requirements.txt**

```
(venv) C:\Users\a8ana\OneDrive\CodingFa
ject>pip freeze > requirements.txt
```

requirements.txt

```
1   numpy==1.24.2
```

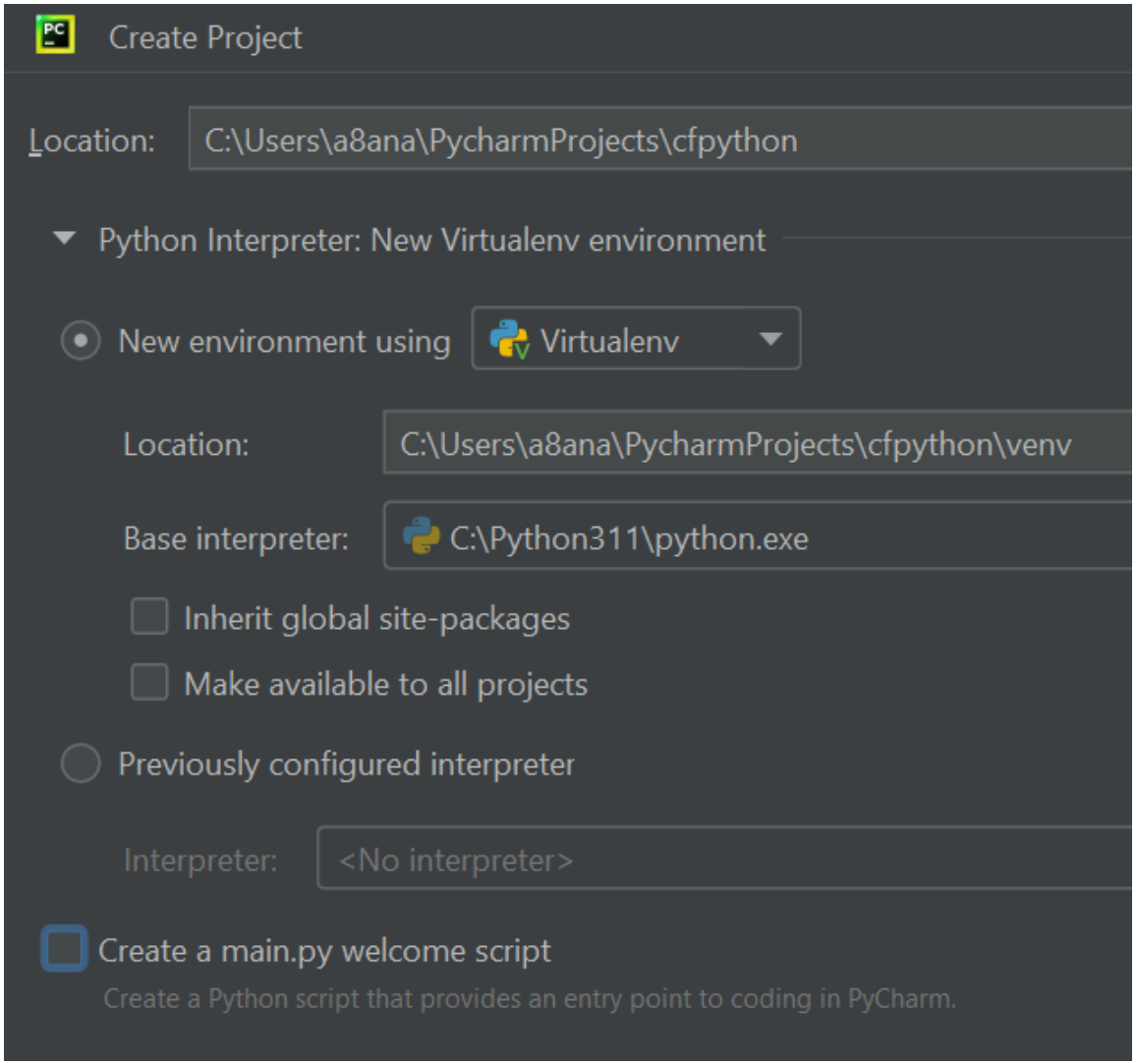
Project με requirements.txt

- Για να κάνουμε migrate ένα project σε ένα άλλο περιβάλλον, δημιουργούμε ένα folder/directory, μετά δημιουργούμε ένα virtual environment και το κάνουμε activate και στη συνέχεια, μέσα στο φάκελο του project εκτελούμε την εντολή
- **pip install –r requirements.txt**

PyCharm

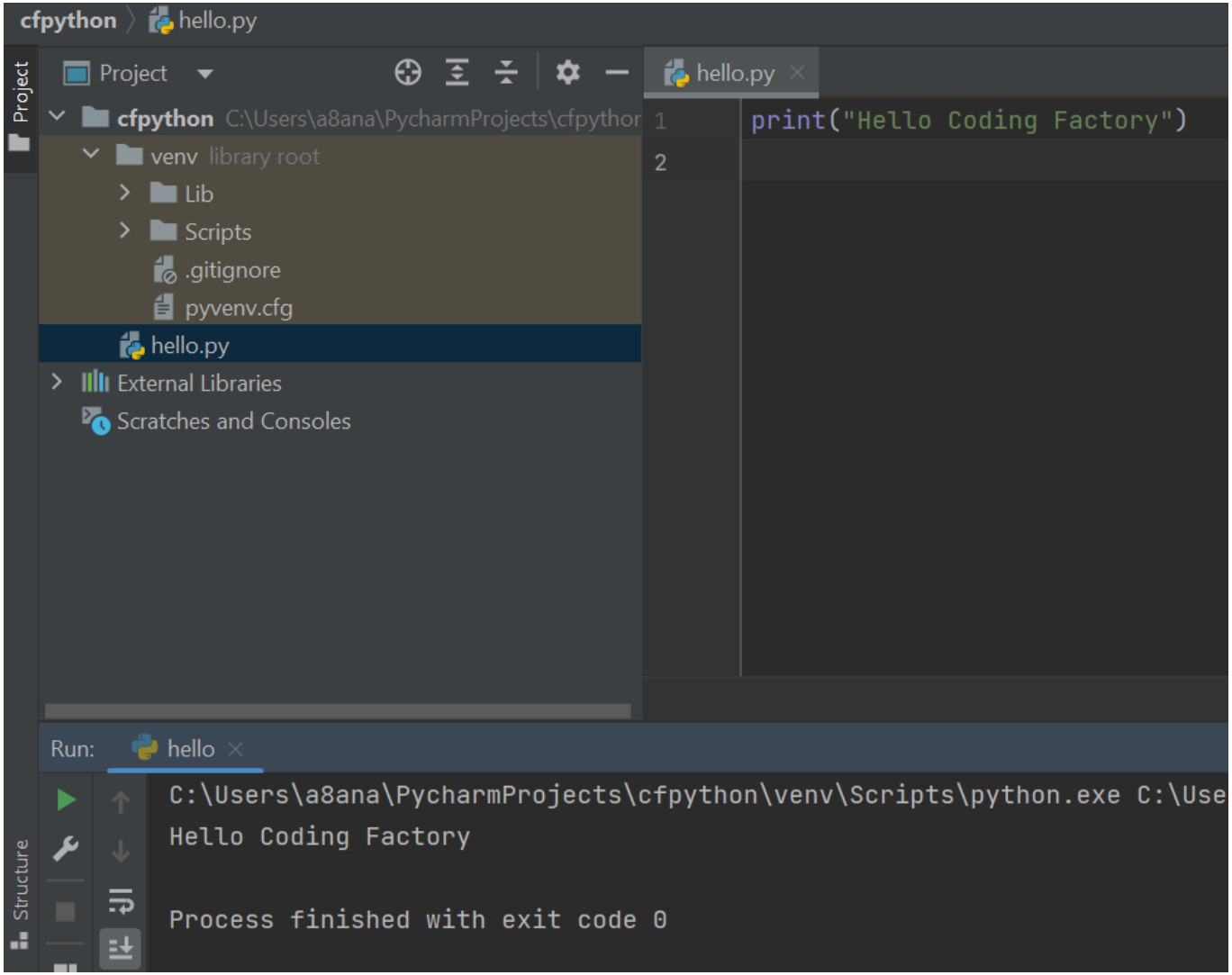

jetbrains.com/pycharm/promo/?source=google&medium=cpc&campaign=14124132468&term=pycharm&content=6038...
JET BRAINS
PC
PyCharm
The Python IDE
for Professional Developers
DOWNLOAD

Δημιουργία νέου Project



- File / New project
- Χρησιμοποιούμε συνήθως το ***Virtualenv*** που είναι ένα εργαλείο δημιουργίας isolated Python environments
- Το *Pipenv* είναι ειδικότερο εργαλείο που περιέχει ένα ειδικό αρχείο pipfile (και pipfile.lock, σε TOML format) για τη διαχείριση των dependencies
- Τα poetry και conda είναι επίσης πιο ειδικοί dependency managers

Hello World στο PyCharm



- Hello Coding Factory στο PyCharm

Python Naming Conventions (1)

- **Μεταβλητές και συναρτήσεις**
- Οι μεταβλητές και οι συναρτήσεις γράφονται με πεζά γράμματα και οι διαφορετικές λέξεις χωρίζονται με κάτω παύλα (_). Χρησιμοποιούνται ονόματα τα οποία να αντιπροσωπεύουν το σκοπό της μεταβλητής και της συνάρτησης αντίστοιχα. π.χ. age, print_hello() κλπ.
- **Κλάσεις**
- Τα ονόματα των κλάσεων γράφονται με PascalCase. π.χ. PersonDetails
- **Σταθερές**
- Οι σταθερές στην Python γράφονται με όλα τα γράμματα κεφαλαία και οι διαφορετικές λέξεις χωρίζονται με κάτω παύλα π.χ. SECONDS_OF_MINUTE

Python Naming Conventions (2)

- **Packages**
- Τα **ονόματα των packages** γράφονται με μικρά γράμματα και συνήθως χρησιμοποιούμε μικρά και περιγραφικά ονόματα κατά περίπτωση. Διαφορετικές λέξεις χωρίζονται με κάτω παύλα (_).
- **Modules**
- Τα ονόματα των modules γράφονται με μικρά γράμματα και οι διαφορετικές λέξεις χωρίζονται με κάτω παύλα (_). Χρησιμοποιούνται μικρά και περιγραφικά ονόματα. π.χ. math, time κ.α.

Python Naming Conventions

- **Ονόματα μεθόδων και instance variables**
- Γράφονται με μικρά γράμματα και οι διαφορετικές λέξεις χωρίζονται με κάτω παύλα (_). Χρησιμοποιούνται ονόματα τα οποία να αντιπροσωπεύουν το σκοπό της ιδιότητας ή της μεθόδου αντίστοιχα. π.χ. age, print_hello(), κλπ.
- **Non-public μεταβλητές ή συναρτήσεις**
- Χρησιμοποιείται ως πρώτο γράμμα η κάτω παύλα (_) για να δηλώσει ότι μια μεταβλητή ή μια μέθοδος δεν είναι public Η διπλή παύλα χρησιμοποιείται σε ειδικές μεθόδους της python.
- π.χ. _age, _balance, _get_id(), __calculate_sth(), __str__, __doc__ κ.α.

Εργασία

- Δημιουργήστε ένα νέο project στο VS Code με όνομα **coding_factory**
- Ανοίξτε ένα νέο terminal (cmd) και (αφού επιβεβαιώσετε ότι είστε μέσα στο φάκελο του project, αλλιώς κάντε cd *folder_name*) δημιουργήστε ένα virtual environment
- Κάντε το activate
- Δημιουργήστε ένα folder chapter1 και γράψτε το hello_world.py
- Πηγαίνετε με cd chapter1 στο chapter1 και εκτελέστε το hello_world.py