ЗАКАРПАТСЬКИЙ УГОРСЬКИЙ УНІВЕРСИТЕР ІМЕНІ ФЕРЕНЦА РАКОЦІ II
II. RÁKÓCZI FERENC KÁRPÁTALJAI MAGYAR EGYETEM

Кафедра математики та інформатики
Matematika és Informatika Tanszék

МОДУЛІ PYTHON ТА МЕНЕДЖЕР ПАКЕТІВ PIP / PYTHON MODULOK ÉS PIP CSOMAGKEZELŐ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ / MÓDSZERTANI AJÁNLÁS
до самостійної роботи / önalló munkakhoz

СУЧАСНІ МОВИ ПРОГРАМУВАННЯ / KORSZERŰ PROGRAMOZÁSI NYELVEK
(назва навчальної дисципліни / a tantárgy neve)

Перший (бакалавр) / Képzési szint (B Sc)
A Освіта / A Oktatás
A4.09 Середня освіта (Інформатика) / A4.09 Középiskolai oktatás (Informatika)

Берегове / Beregszász
2026

Методичні рекомендації «Модулі Python та менеджер пакетів Pip» до самостійної роботи з дисципліни «Сучасні мови програмування» розроблені на основі Освітньої програми підготовки бакалаврі з галузі знань «*А Освіта*» за напрямом « *А4.09 Середня освіта (Інформатика* » як для студентів денної, так і заочної форми навчання. Метою методичного посібника засвоєння студентами методів використанняє менеджера пакетів Pip для підключення модулів Python. У роботі розглядаються основні можливості практичного використання менеджера Pip, та наводиться довідковий матеріал для усунення помилок, що виникають при його використанні. Наводяться посилання на інформаційні ресурси, які можуть бути використані для поглиблення знань студентів по даному предмету.

Затверджено до використання у навчальному процесі на засіданні кафедри математики та інформатики ЗУУ ім. Ф.Ракоці ІІ (протокол №7 від «16» квітня 2026 року)

Розглянуто та рекомендовано Радою забезпечення якості вищої освіти Закарпатського угорського університету імені Ференца Ракоці ІІ (протокол №5 від «25» травня 2026 року)

Рекомендовано до видання в електронній формі (PDF) рішенням Вченої ради Закарпатського угорського університету імені Ференца Ракоці ІІ (протокол №5 від «27 » травня 2026 року)

Підготовлено до видання в електронній формі (PDF) кафедрою математики та інформатики спільно з Видавничим відділом Закарпатського угорського університету ім. Ференца Ракоці ІІ

Розробник методичних вказівок:

Йожеф ГОЛОВАЧ – професор кафедри математики та інформатики Закарпатського угорського університету імені Ференца Ракоці ІІ, доктор технічних наук.

Рецензенти:

Олександр МІЦА – професор кафедри інформаційних управляючих систем та технологій УжНУ, доктор техніічих наук, професор (м. Ужгород)

Олександр Тилищак – професор кафедри математики та інформатики Закарпатського угорського університету імені Ференца Ракоці ІІ, доктор фізико-математичних наук

Відповідальні за випуск:

Мирослав СТОЙКА – завідувач кафедри математики та інформатики Закарпатського угорського університету імені Ференца Ракоці ІІ, доцент, кандидат фізико-математичних наук

Олександр ДОБОШ – начальник Видавничого відділу ЗУУ ім. Ф.Ракоці ІІ

За зміст методичних вказівок відповідальність несе розробник.

Видавництво: Закарпатський угорський університет імені Ференца Ракоці ІІ (адреса: пл. Кошута 6, м. Берегове, 90202)

A „Modern programozási nyelvek” tárgy önálló munkához szükséges „Python modulok és Pip csomagkezelő” című módszertani ajánlásokat az „A4 középfokú oktatás” specializáció „A4.09 Középfokú oktatás (Informatika)” képzési program alapján dolgozták ki, mind a nappali, mind a levelező tagozatos hallgatók számára.
A módszertani útmutató célja, hogy segítse a hallgatókat a Pip csomagkezelő Python modulok összekapcsolására való használatának módszereinek elsajátításában. Az utmutató a Pip kezelő gyakorlati alkalmazásának főbb lehetőségeit vizsgálja, és referenciaanyagokat biztosít a használata során előforduló hibák kiküszöbölésére. Linkeket tartalmaz az információs forrásokhoz, amelyek segítségével elmélyíthetők a hallgatók ismeretei ebben a témában.

Az oktatási folyamatban történő felhasználását jóváhagyta a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Matematika és Informatika Tanszéke (2026.04.16 , 7 számú jegyzőkönyv).

Megjelentetésre javasolta a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Minőségbiztosítási Tanácsa (2026.05ю25. 5. számú jegyzőkönyv).

Elektronikus formában (PDF fájlformátumban) történő kiadásra javasolta a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Tudományos Tanácsa (2026.27.05. 5.számú jegyzőkönyv).

Kiadásra előkészítette a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Matematika és Informatika Tanszéke, valamint Kiadói Részlege.

A módszertani útmutató kidolgozója:

HOLOVÁCS József – professzor, műszaki tudományok doktora, a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Matematika és Informatika Tanszékének professzora

Szakmai lektorok:

MITSA Oleksandr – Az UNE Információs vezérlési Rendszerek és Technológiák Tanszék professzora, műszaki tudományok doktora

Олександр Тилищак

TILISTYÁK Oleksandr – a fizikai és matematikai tudományok dorkora, a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Matematika és Informatika Tanszékének professzora

A kiadásért felelnek:

SZTOJKA Miroszláv – PhD, a fizikai és matematikai tudományok kandidátusa, a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Matematika és Informatika Tanszékének tanszékvezetője és docense

DOBOS Sándor – a II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem Kiadói Részlegének vezetője

A segédlet tartalmáért kizárólag a módszertani útmutató kidolgozója felel.

Kiadó: II. Rákóczi Ferenc Kárpátaljai Magyar Egyetem (cím: 90202, Beregszász, Kossuth tér 6. E-mail: foiskola@kmf.uz.ua)

ЗМІСТ/ TARTALOM

Довідники

Популярні модулі Python / Népszerű Python modulok

pip install не працює: причини і способи виправлення/ A pip telepítése nem működik: okok és javítási módok

Використання комбінації клавіш Win+R./ A Win+R billentyűkombináció használata

Пропоновані джерела/ Javasolt források

Вступ/ Bevezetés

Мова програмування Python є однією з найпопулярніших сучасних мов програмування завдяки простому синтаксису, широким можливостям та великій кількості готових бібліотек. Python активно використовується у веброзробці, аналізі даних, штучному інтелекті, автоматизації, математичних обчисленнях, створенні графічних інтерфейсів та STEM-освіті.

Однією з головних переваг Python є модульність. Програми можуть складатися з окремих модулів і пакетів, що дозволяє повторно використовувати код, спрощує розробку та підвищує зрозумілість програм. Окрім стандартної бібліотеки, Python має величезну кількість додаткових пакетів, створених спільнотою розробників.

Для встановлення та керування пакетами використовується менеджер пакетів pip. За його допомогою можна швидко встановлювати, оновлювати та видаляти програмні модулі. Використання pip значно спрощує роботу програміста та дозволяє легко підключати сучасні бібліотеки для розв'язування практичних задач.

У даній методичній розробці розглядаються основні поняття модулів і пакетів у Python, принципи їх використання, а також робота з менеджером пакетів pip. Наведено приклади встановлення популярних бібліотек та практичні приклади їх застосування.

A Python programozási nyelv napjaink egyik legnépszerűbb programozási nyelve egyszerű szintaxisa, széleskörű lehetőségei és a rendelkezésre álló könyvtárak nagy száma miatt. A Python nyelvet széles körben alkalmazzák webfejlesztésben, adatelemzésben, mesterséges intelligenciában, automatizálásban, matematikai számításokban, grafikus felületek készítésében és STEM-oktatásban.

A Python egyik legfontosabb előnye a modularitás. A programok különálló modulokból és csomagokból épülhetnek fel, ami lehetővé teszi a kód újrafelhasználását, egyszerűsíti a fejlesztést és növeli a programok áttekinthetőségét. A szabványos könyvtár mellett a Python hatalmas mennyiségű, a fejlesztői közösség által készített külső csomagot is kínál.

A csomagok telepítésére és kezelésére a pip csomagkezelő szolgál. Segítségével gyorsan telepíthetők, frissíthetők és eltávolíthatók a különböző programkönyvtárak. A pip használata jelentősen megkönnyíti a programozók munkáját, és lehetővé teszi modern könyvtárak egyszerű integrálását különféle gyakorlati feladatok megoldásához.

Jelen módszertani segédanyag bemutatja a Python moduljainak és csomagjainak alapfogalmait, használatuk elveit, valamint a pip csomagkezelő alkalmazását. A jegyzet példákat tartalmaz népszerű könyvtárak telepítésére és gyakorlati használatára is.

На зміст

1. Загальна характеристика модулів/ A modulok általános jellemzői

Модулі Python — це файли з розширенням .py, що містять код (функції, класи, змінні), який можна використовувати повторно. Внутрішні (стандартні) модулі постачаються разом з Python (наприклад, math, os, sys), а зовнішні розробляються спільнотою та встановлюються окремо через pip (наприклад, requests, pandas).

A Python modulok .py kiterjesztésű fájlok, amelyek újrafelhasználható kódot (függvényeket, osztályokat, változókat) tartalmaznak. A belső (standard) modulok a Python részét képezik (pl. math, os, sys), míg a külső modulokat a közösség fejleszti, és külön telepíti pip-en keresztül (pl. requests, pandas).

1.1. Внутрішні (стандартні) модулі Python/ Belső (standard) Python modulok

Це вбудована бібліотека, яка готова до використання одразу після встановлення Python. Вони не вимагають встановлення тільки імпорту.

Приклади: os (робота з ОС), sys (параметри системи), math (математичні функції), datetime (робота з датами), random (генерація випадкових чисел), json (парсинг даних), re (регулярні вирази).

Використання:

import math

print(math.sqrt(16)) # Результат: 4.0

Ezek beépített könyvtárak, amelyek a Python telepítése után azonnal használatra készek. Nem igényelnek telepítést, csak importálást.

Példák: os (OS), sys (rendszerparaméterek), math (matematikai függvények), datetime (dátumkezelés), random (véletlenszám-generálás), json (adatelemzés), re (reguláris kifejezések).

Használat:

import math

print(math.sqrt(16)) # Eredmény: 4.0

На зміст

1.2. Зовнішні модулі Python/ Külső Python modulok

Це сторонні бібліотеки, створені іншими розробниками розширення функціональності мови. Вони зберігаються у репозиторії PyPI (Python Package Index) та встановлюються за допомогою менеджера пакетів pip.

Приклади: requests (запити до HTTP), pandas (аналіз даних), numpy (наукові обчислення), flask/django (веб-розробка).

Використання:

Установка: pip install requests

Імпорт: import requests

Ezek harmadik féltől származó könyvtárak, amelyeket más fejlesztők hoztak létre a nyelv funkcionalitásának kibővítésére. A PyPI (Python Package Index) adattárban tárolódnak, és a pip csomagkezelővel telepíthetők.

Példák: kérések (HTTP kérések), pandas (adatelemzés), numpy (tudományos számítástechnika), flask / django (webfejlesztés).

Használat:

Telepítés: pip install requests

Importálás: import requests

На зміст

1.3. Основні відмінності

Установка: Внутрішні модулі - "з коробки", зовнішні - через pip.

Джерело: Внутрішні підтримуються командою розробників Python, зовнішні - спільнотою.

Призначення: Внутрішні покривають базові завдання, зовнішні - специфічні (Data Science, Інтернет, тестування).

Főbb különbségek

Telepítés: A belső modulok alapértelmezés szerint telepítve vannak, a külsők pipen keresztül.

Forrás: A belső modulokat a Python fejlesztőcsapat, a külsőket pedig a közösség tartja karban.

Cél: A belső modulok alapvető feladatokat, a külsők pedig specifikus feladatokat (adattudomány, web, tesztelés) fednek le.

На зміст

2. Використання pip (pip3) /A pip (pip3) használata

2.1. Що таке pip/ Mi az a pip

pip — це стандартний менеджер пакетів Python.
Його використовують для:

- встановлення додаткових бібліотек;
- оновлення бібліотек;
- видалення бібліотек;
- перегляду списку встановлених пакетів;
- збереження списку залежностей у файл;
- автоматичного встановлення пакетів із файлу.

Простіше кажучи, pip дозволяє розширювати можливості Python без ручного копіювання файлів бібліотек.

Наприклад, якщо хочемо працювати з:

- математикою — numpy, scipy, sympy,
- графікою — matplotlib,
- таблицями даних — pandas,
- веброзробкою — flask, django,

то найчастіше ці пакети встановлюються саме через pip.

A pip a Python szabványos csomagkezelője.
Arra használjuk, hogy:

- új könyvtárakat telepítsünk,
- meglévő csomagokat frissítsünk,
- csomagokat eltávolítsunk,
- megtekintsük a telepített csomagok listáját,
- a függőségeket fájlba mentsük,
- egy fájlból automatikusan telepítsük a szükséges csomagokat.

Egyszerűbben: a pip segítségével bővíthetjük a Python lehetőségeit anélkül, hogy kézzel másolnánk be a könyvtárakat.

Például ha az alábbi területeken dolgozunk:

- matematika — numpy, scipy, sympy,
- grafika — matplotlib,
- adatelemzés — pandas,
- webfejlesztés — flask, django,

akkor ezeket általában pip segítségével telepítjük.

На зміст

2.2. pip і pip3: у чому різниця/ pip és pip3: mi a különbség

Історично склалося так, що на деяких комп'ютерах:

- pip може бути пов'язаний із Python 2,
- pip3 – із Python 3.

Оскільки зараз у навчанні та практиці майже завжди використовується **Python 3**, часто застосовують команду:

pip3 install package_name

або

python3 -m pip install package_name

У Windows часто працює і так:

py -m pip install package_name

Найбезпечніший варіант — запускати pip через сам інтерпретатор Python:

python -m pip install package_name

або

python3 -m pip install package_name

Це важливо, бо на комп'ютері може бути кілька версій Python, і потрібно бути впевненим, що пакет ставиться саме до потрібної версії.

Történetileg előfordulhat, hogy egyes gépeken:

- a pip a Python 2-höz kapcsolódik,
- a pip3 pedig a Python 3-hoz.

Mivel ma a gyakorlatban szinte mindig **Python 3**-at használunk, gyakran ezt a parancsot látjuk:

pip3 install csomag_neve

vagy

python3 -m pip install csomag_neve

Windows alatt gyakran ez is működik:

py -m pip install csomag_neve

A legbiztonságosabb módszer, ha a pip-et közvetlenül a Python interpreterrel indítjuk:

python -m pip install csomag_neve

vagy

python3 -m pip install csomag_neve

Ez azért fontos, mert a gépen több Python-verzió is lehet, és biztosnak kell lennünk abban, hogy a csomag a megfelelő verzióhoz települ.

На зміст

2.3. Як перевірити, чи встановлено Python і pip / Hogyan ellenőrizzük, hogy telepítve van-e a Python és a pip

У терміналі або командному рядку можна виконати:

python --version

або

python3 --version

Щоб перевірити pip:

pip --version

або

pip3 --version

Ще надійніше:

python -m pip --version

Приклад результату:

pip 24.0 from .../site-packages/pip (python 3.11)

Це означає, що pip встановлений і працює разом із Python 3.11.

Terminálban vagy parancssorban a következőkkel ellenőrizhetjük a Python verzióját:

python --version

vagy

python3 --version

A pip ellenőrzéséhez:

pip --version

vagy

pip3 --version

Még biztosabb módszer:

python -m pip --version

Példa kimenet:

pip 24.0 from .../site-packages/pip (python 3.11)

Ez azt jelenti, hogy a pip telepítve van, és a Python 3.11-hez kapcsolódik.

На зміст

2.4. Встановлення пакета / Csomag telepítése

Загальна форма:

pip install package_name

або

python -m pip install package_name

Приклади:

pip install numpy

pip install matplotlib

pip install pandas

Якщо використовується Python 3:

pip3 install numpy

або

python3 -m pip install numpy

Після встановлення пакет зазвичай можна імпортувати в програмі:

import numpy

Általános alak:

pip install csomag_neve

vagy

python -m pip install csomag_neve

Példák:

pip install numpy

pip install matplotlib

pip install pandas

Ha kifejezetten Python 3-at használunk:

pip3 install numpy

vagy

python3 -m pip install numpy

Telepítés után a csomagot általában így importálhatjuk:

import numpy

Встановлення конкретної версії/ Meghatározott verzió telepítése

Іноді потрібно встановити не найновішу, а конкретну версію пакета:

pip install numpy==1.26.4

Тут == означає точний збіг версії.

Можна також задавати обмеження:

pip install "numpy>=1.24,<2.0"

Це означає: встановити версію не нижче 1.24, але меншу за 2.0.

Такі записи корисні в навчальних проєктах, де код перевірений лише на певних версіях бібліотек.

Néha nem a legújabb, hanem egy konkrét verzióra van szükség:

pip install numpy==1.26.4

Itt a == pontos verzióegyezést jelent.

Feltételeket is megadhatunk:

pip install "numpy>=1.24,<2.0"

Ez azt jelenti: olyan verzió települjön, amely legalább 1.24, de kisebb mint 2.0.

Ez különösen hasznos oktatási projektekben, amikor a kódot csak bizonyos könyvtárverziókkal ellenőrizték.

На зміст

2.5. Оновлення пакету / Csomag frissítése

Для оновлення використовують ключ --upgrade:

pip install --upgrade numpy

Скорочений варіант:

pip install -U numpy

Щоб оновити сам pip:

python -m pip install --upgrade pip

Це часто корисно, якщо старий pip погано працює з новими пакетами.

Frissítéshez a --upgrade kapcsolót használjuk:

pip install --upgrade numpy

Rövid változat:

pip install -U numpy

Magának a pip-nek a frissítése:

python -m pip install --upgrade pip

Ez gyakran hasznos, ha egy régebbi pip már nem működik jól az újabb csomagokkal.

На зміст

2.6. Видалення пакету / Csomag eltávolítása

Для видалення:

pip uninstall numpy

Після цього система може попросити підтвердження.

Щоб уникнути запиту:

pip uninstall -y numpy

Eltávolításhoz:

pip uninstall numpy

A rendszer általában megerősítést kér.

Megerősítés nélküli változat:

pip uninstall -y numpy

На зміст

2.7. Перегляд встановлених пакетів / A telepített csomagok listázása

Щоб побачити всі встановлені пакети:

pip list

Щоб отримати детальнішу інформацію:

pip show numpy

Приклад:

pip show matplotlib

Це може показати:

- назву пакета,

- версію,
- місце встановлення,
- залежності.

A telepített csomagok listája:

pip list

Részletesebb információ egy adott csomagról:

pip show numpy

Például:

pip show matplotlib

Ez általában megmutatja:

- a csomag nevét,
- verzióját,
- telepítési helyét,
- függőségeit.

На зміст

2.8. Збереження у файл requirements.txt/ Mentés a requirements.txt fájlba

У проєктах часто потрібно зберегти список пакетів у файл:

pip freeze > requirements.txt

У результаті створюється файл, наприклад:

numpy==1.26.4

matplotlib==3.8.2

pandas==2.2.0

Це потрібно для того, щоб інша людина пізніше могли відтворити таке саме програмне середовище.

Projektekben gyakran szükség van a csomaglista fájlba mentésére:

pip freeze > requirements.txt

Ennek eredményeként létrejön egy fájl, például:

numpy==1.26.4

matplotlib==3.8.2

pandas==2.2.0

Ez azért fontos, hogy mások később ugyanazt a környezetet tudja újra létrehozni.

На зміст

2.9. Встановлення пакетів із requirements.txt/ 10. Csomagok telepítése requirements.txt fájlból

Якщо є файл requirements.txt, усі потрібні пакети можна встановити однією командою:

pip install -r requirements.txt

Це дуже зручно в командній роботі, на лабораторних заняттях і при перенесенні проєкту на інший комп'ютер.

Ha már van requirements.txt fájl, akkor az összes szükséges csomag egyetlen paranccsal telepíthető:

pip install -r requirements.txt

Ez nagyon kényelmes csapatmunkában, laborfoglalkozásokon és akkor is, amikor a projektet másik gépre visszük át.

На зміст

2.10. Віртуальне середовище/ Virtuális környezet

Дуже важливо розуміти, що пакети краще встановлювати не глобально, а у **віртуальному середовищі**.

Віртуальне середовище — це окрема папка проєкту зі своїм Python і власними пакетами.
Це дозволяє:

- не змішувати бібліотеки різних проєктів;

- уникати конфліктів версій;
- легко переносити проєкт.

Створення віртуального середовища:

python -m venv venv

Активація у Windows:

venv\Scripts\activate

Після активації можна встановлювати пакети:

pip install numpy matplotlib

Тоді вони потраплять саме в це середовище.

Вихід із середовища:

deactivate

Nagyon fontos megérteni, hogy a csomagokat jobb nem globálisan, hanem **virtuális környezetben** telepíteni.

A virtuális környezet egy külön projektmappa saját Python-környezettel és saját csomagokkal.
Előnyei:

- a különböző projektek könyvtárai nem keverednek,
- elkerülhetők a verzióütközések,
- a projekt könnyebben hordozható.

Virtuális környezet létrehozása:

python -m venv venv

Aktiválás Windows alatt:

venv\Scripts\activate

Aktiválás után telepíthetünk csomagokat:

pip install numpy matplotlib

Ezek ekkor a virtuális környezetbe kerülnek.

Kilépés:

deactivate

На зміст

2.11. Типові корисні команди / Gyakori hasznos parancsok

python -m pip --version

python -m pip install numpy

python -m pip install --upgrade pip

python -m pip list

python -m pip show pandas

python -m pip uninstall matplotlib

python -m pip freeze > requirements.txt

python -m pip install -r requirements.txt

2.12. Типові помилки та їх пояснення/ Tipikus hibák és magyarázatuk

1. pip не розпізнається як команда

Приклад:

'pip' is not recognized as an internal or external command

Причина:
pip не доданий у PATH або Python встановлений некоректно.

Що робити:

- спробувати:

python -m pip --version

- перевірити, чи встановлено Python;
- при перевстановленні Python увімкнути опцію **Add Python to PATH**.

2. Пакет встановився, але import не працює

Причина:
пакет встановлено в одну версію Python, а програма запускається іншою.

Що робити:

python -m pip install package_name

і запускати програму тим самим python.

3. Немає прав доступу

Приклад:

Permission denied

Причина:
система не дозволяє глобальне встановлення.

Що робити:

- використати віртуальне середовище;
- або встановити пакет лише для поточного користувача:

pip install --user package_name

4. Застарілий pip

Деякі пакети можуть не встановлюватися через стару версію pip.

Рішення:

python -m pip install --upgrade pip

5. Проблеми з інтернетом або дзеркалом пакетів

pip завантажує пакети з інтернету. Якщо з'єднання погане, встановлення може перерватися.

---

1. A pip parancs nem ismert

Példa:

'pip' is not recognized as an internal or external command

Ok:
a pip nincs benne a PATH változóban, vagy a Python telepítése hibás.

Megoldás:

- próbáljuk meg:

python -m pip --version

- ellenőrizzük, hogy a Python telepítve van-e;
- újratelepítéskor jelöljük be az **Add Python to PATH** opciót.

2. A csomag települt, de az import nem működik

Ok:
a csomag másik Python-verzióhoz települt, mint amellyel a program fut.

Megoldás:

python -m pip install csomag_neve

és ugyanazzal a python paranccsal futtassuk a programot.

3. Jogosultsági hiba

Példa:

Permission denied

Ok:
a rendszer nem engedi a globális telepítést.

Megoldás:

- használjunk virtuális környezetet;
- vagy telepítsünk csak a jelenlegi felhasználónak:

pip install --user csomag_neve

4. Elavult pip

Egyes csomagok nem települnek megfelelően régi pip verzióval.

Megoldás:

python -m pip install --upgrade pip

5. Internetkapcsolati probléma

A pip az internetről tölti le a csomagokat, ezért gyenge kapcsolat esetén a telepítés megszakadhat.

На зміст

3. Створення власних модулів у Python/ Saját modulok készítése Pythonban

Модуль у Python — це звичайний файл з розширенням .py, у якому зберігаються функції, змінні або класи.
Якщо програма стає більшою, зручно винести частину коду в окремий файл і підключати його через import.

Наприклад, можна створити файл mymath.py, у якому будуть власні математичні функції, а потім використовувати їх у файлі main.py.

A Python modul egy szokásos .py kiterjesztésű fájl, amely függvényeket, változókat vagy osztályokat tartalmaz.
Ha a program nagyobb lesz, célszerű a kód egy részét külön fájlba helyezni, majd import segítségével felhasználni.

Például létrehozhatunk egy mymath.py fájlt saját matematikai függvényekkel, majd ezeket a main.py fájlban használhatjuk.

Важливе зауваження / Fontos megjegyzés

Щоб імпорт працював, файли main.py і mymath.py повинні бути або в одній папці, або модуль має бути в доступному шляху пошуку Python.

Ahhoz, hogy az import működjön, a main.py és mymath.py fájloknak ugyanabban a mappában kell lenniük, vagy a modulnak a Python keresési útvonalán kell szerepelnie.

Структура файлів / Fájlszerkezet

project/

│

├── mymath.py

└── main.py

Приклади/Példák

Приклад 1 / 1. példa

Файл mymath.py

```
def square(x):
    return x * x
def cube(x):
    return x * x * x
def average(a, b):
    return (a + b) / 2
```

Файл main.py

```
import mymath
print("square(4) =", mymath.square(4))
print("cube(3) =", mymath.cube(3))
print("average(10, 20) =", mymath.average(10, 20))
```

Результат / Eredmény

```
square(4) = 16
cube(3) = 27
average(10, 20) = 15.0
```

Приклад 2: імпорт окремих функцій /2. példa: külön függvények importálása

```
from mymath import square, average
print(square(5))
```

print(average(8, 12))

Тут ми імпортуємо не весь модуль, а лише окремі функції.

Itt nem a teljes modult importáljuk, hanem csak néhány függvényt.

Приклад 3: псевдонім модуля / 3. példa: modul álnévvel

Псевдонім (as mm) робить запис коротшим. /Az álnév (as mm) rövidebbé teszi az írást.

import mymath as mm

print(mm.square(6))

print(mm.cube(2))

Приклад 4: модуль для роботи з рядками / 4. példa: modul szövegekhez

Файл mystrings.py

```
def shout(text):
    return text.upper()
def first_letter(text):
    return text[0]
def word_count(text):
    return len(text.split())
```

Файл main.py

```
import mystrings

s = "python is interesting"

print(mystrings.shout(s))

print(mystrings.first_letter(s))

print(mystrings.word_count(s))
```

Висновок / Összegzés

Робота з власними модулями — це перший крок до правильної організації більших програм.
Студент повинен уміти не лише використовувати готові бібліотеки через pip, а й створювати власні модулі.

A saját modulok használata az első lépés a nagyobb programok helyes szervezése felé.
A hallgatónak nemcsak kész könyvtárakat kell tudnia használni pip segítségével, hanem saját modulokat is létre kell tudnia hozni.

На зміст

4. Завдання по використанню Pip

4.1. Короткі навчальні приклади/ Rövid oktatási példák

Приклад 1. Встановити бібліотеку numpy

pip install numpy

Потім перевірити в Python:

import numpy as np

print(np.array([1, 2, 3]))

Приклад 2. Оновити matplotlib

pip install --upgrade matplotlib

Приклад 3. Зберегти список пакетів

pip freeze > requirements.txt

Приклад 4. Встановити пакети з файлу

pip install -r requirements.txt

---

1. példa: a numpy telepítése

pip install numpy

Ezután ellenőrzés Pythonban:

import numpy as np

print(np.array([1, 2, 3]))

2. példa: a matplotlib frissítése

pip install --upgrade matplotlib

3. példa: a csomaglista mentése

pip freeze > requirements.txt

4. példa: csomagok telepítése fájlból

pip install -r requirements.txt

На зміст

4.2. Збірник завдань з Python по використанню PIP

Зміст комплексу завдань

- ЛР 1. Основи pip: перевірка Python і встановлення пакетів
- ЛР 2. Віртуальне середовище venv
- ЛР 3. Базові обчислення з numpy
- ЛР 4. Побудова графіків у matplotlib
- ЛР 5. Основи pandas
- ЛР 6. Міні-проєкт: аналіз даних і візуалізація
- ЛР 7. Створення власних модулів

Лабораторна робота №1 / 1. labor

Основи pip: перевірка Python і встановлення пакетів / A pip alapjai: Python ellenőrzése és csomagtelepítés

Мета / Cél

Навчитися навчитися перевіряти наявність Python і pip, встановлювати пакети та переглядати список бібліотек.

A feladat célja: ellenőrizni a Python és a pip jelenlétét, csomagot telepíteni és a könyvtárak listáját megjeleníteni.

Інструкція / Utasítás

Крок 1. Перевірка версій / Verzióellenőrzés

```
python --version
python -m pip –version
```

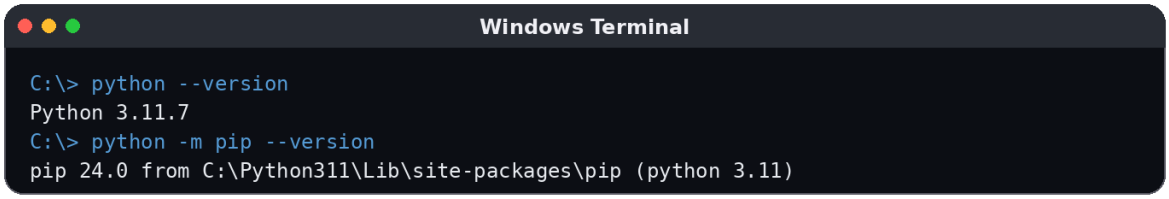

Рис. 1.1. Ілюстрація до кроку 1 / Ábra 1.1. A(z) 1. lépés illusztrációja

Крок 2. Встановлення numpy / A numpy telepítése

python -m pip install numpy

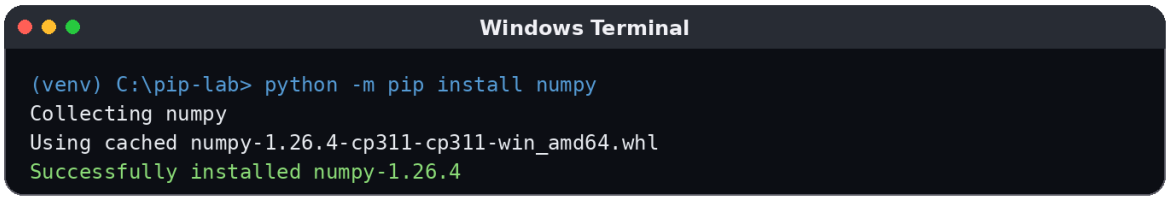

Рис. 1.2. Ілюстрація до кроку 2 / Ábra 1.2. A(z) 2. lépés illusztrációja

Крок 3. Список пакетів / Csomaglista

python -m pip list

Що здати / Beadandó

Скріншоти команд і короткий висновок / Képernyőképek és rövid összegzés.

Лабораторна робота №2 / 2. labor

Віртуальне середовище venv / Virtuális környezet venv

Мета / Cél

Навчитися створити ізольоване середовище для окремого проєкту.

A feladat célja: elkülönített környezet létrehozása egy projekthez.

Інструкція / Utasítás

Крок 1. Створення та активація / Létrehozás és aktiválás

```
python -m venv venv
venv\Scripts\activate
```

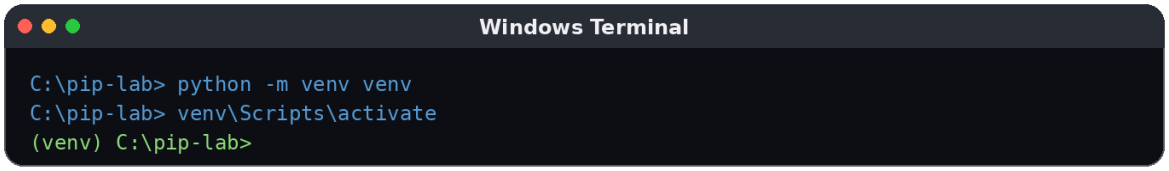

Рис. 2.1. Ілюстрація до кроку 1 / Ábra 2.1. A(z) 1. lépés illusztrációja

Крок 2. Встановлення numpy у venv / Numpy telepítése a venv-ben

```
python -m pip install numpy
python -m pip list
```

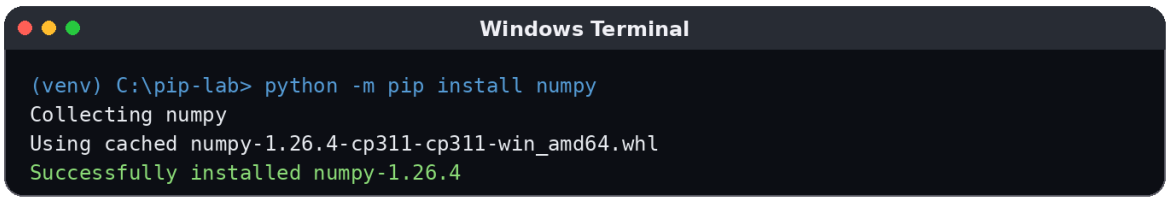

Рис. 2.2. Ілюстрація до кроку 2 / Ábra 2.2. A(z) 2. lépés illusztrációja

Що здати / Beadandó

Скріншот появи префікса (venv) і список пакетів / Képernyőkép a (venv) előtagról és a csomaglistáról.

Лабораторна робота №3 / 3. labor

Базові обчислення з numpy / Alapműveletek numpy-val

Мета / Cél

Навчитися працювати з масивами, сумою, середнім та простими обчисленнями.

A feladat célja: tömbök, összeg, átlag és egyszerű számítások kezelése.

Інструкція / Utasítás

Крок 1. Код / Kód

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
print("Array:", a)
print("Sum:", np.sum(a))
print("Mean:", np.mean(a))
```

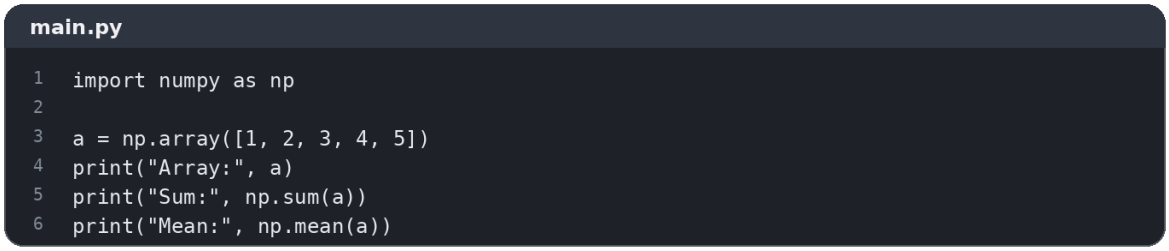

Рис. 3.1. Ілюстрація до кроку 1 / Ábra 3.1. A(z) 1. lépés illusztrációja

Крок 2. Результат / Kimenet

Рис. 3.2. Ілюстрація до кроку 2 / Ábra 3.2. A(z) 2. lépés illusztrációja

Що здати / Beadandó

Файл .py і скріншот результату / .py fájl és képernyőkép a kimenetről.

Лабораторна робота №4 / 4. labor

Побудова графіків у matplotlib

Grafikonok rajzolása matplotlib-pel

Мета / Cél

Навчитися навчитися будувати графік функції та підписувати осі.

A feladat célja: függvénygrafikon rajzolása és tengelyfeliratok használata.

Інструкція / Utasítás

Крок 1. Встановлення / Telepítés

```
python -m pip install matplotlib
```

Крок 2. Код / Kód

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

plt.plot(x, y, marker='o')
plt.title("y = x^2")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Крок 3. Приклад графіка / Példa grafikon

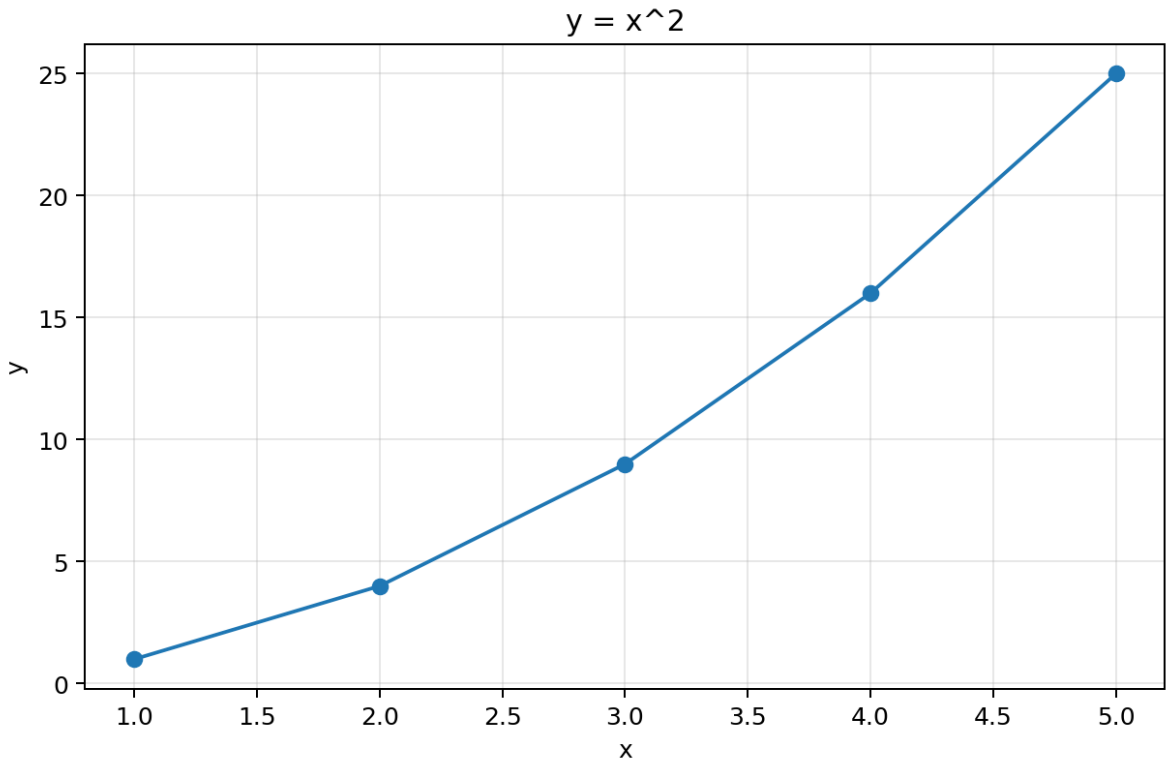

Рис. 4.3. Ілюстрація до кроку 3 / Ábra 4.3. A(z) 3. lépés illusztrációja

Що здати / Beadandó

Код і скріншот графіка / Kód és a grafikon képernyőképe.

Лабораторна робота №5 / 5. labor

Основи pandas

A pandas alapjai

Мета / Cél

Навчитися створювати таблицю даних DataFrame і переглядати її.

A feladat célja: DataFrame létrehozása és megjelenítése.

Інструкція / Utasítás

Крок 1. Код / Kód

```
import pandas as pd
data = {"A": [1, 2, 3], "B": [4, 5, 6]}
df = pd.DataFrame(data)
print(df)
```

Крок 2. Встановлення / Telepítés

```
python -m pip install pandas
```

Що здати / Beadandó

Файл .py і коротке пояснення стовпців / .py fájl és rövid magyarázat az oszlopokról.

Лабораторна робота №6 / 6. labor

Міні-проєкт: аналіз даних і візуалізація

Mini projekt: adatelemzés és vizualizáció

Мета / Cél

Навчитися об’єднати numpy, pandas і matplotlib в одному міні-проєкті.

A feladat célja: numpy, pandas és matplotlib együttes használata mini projektben.

Інструкція / Utasítás

Крок 1. Приклад задачі / Feladatötlet

Створити таблицю з 5–10 значень, обчислити середнє і побудувати графік. / Hozzon létre 5–10 adatból álló táblát, számítsa ki az átlagot és készítsen grafikont.

Що здати / Beadandó

Короткий звіт із висновком / Rövid jelentés következtetéssel.

Методичні поради / Módszertani megjegyzések

Для всіх лабораторних рекомендовано використовувати python -m pip замість просто pip.

A laborokhoz célszerű a python -m pip formát használni a sima pip helyett.

Лабораторна робота №7 / 7. labor

Тема / Téma: Створення власних модулів у Python / Saját modulok készítése Pythonban

Мета / Cél

Навчитися створювати власні модулі, імпортувати функції з інших файлів і організовувати програму з кількох частин.

A hallgató tanulja meg saját modulok létrehozását, más fájlokból való importálást, valamint a program részekre bontását.

Завдання 1 / 1. feladat

Створіть модуль mycalc.py з функціями:

- add(a, b)
- subtract(a, b)
- multiply(a, b)

Створіть файл main.py і перевірте роботу цих функцій.

Hozzon létre egy mycalc.py modult az alábbi függvényekkel:

- add(a, b)
- subtract(a, b)
- multiply(a, b)

Készítsen main.py fájlt, és ellenőrizze a működésüket.

Завдання 2 / 2. feladat

Створіть модуль geometry.py з функціями:

- площа квадрата;
- площа прямокутника;
- периметр прямокутника.

Készítsen geometry.py modult az alábbi függvényekkel:

- négyzet területe;
- téglalap területe;
- téglalap kerülete.

Завдання 3 / 3. feladat

Створіть модуль mytext.py, який містить функції:

- кількість символів у рядку;
- кількість слів;
- перетворення рядка у верхній регістр.

Készítsen mytext.py modult, amely tartalmazza:

- a karakterek számát;
- a szavak számát;
- a szöveg nagybetűssé alakítását.

Контрольні питання / Ellenőrző kérdések

1. Що таке модуль у Python?

   Mi a modul Pythonban?

2. Для чого використовують import?

   Mire használjuk az import utasítást?

3. У чому різниця між

   import mymath

   і

   from mymath import square?

4. Чому зручно ділити програму на модулі?

   Miért hasznos a programot modulokra bontani?

На зміст

ДОВІДНИКИ/ KÉZIKÖNYVEK

Практичні поради/ Gyakorlati tanácsok

1. Для навчальних проєктів краще використовувати **віртуальне середовище**.
2. Краще писати:

python -m pip ...

ніж просто pip ..., бо це точніше.

3. Перед встановленням великої кількості пакетів корисно оновити pip
4. Якщо код не працює після встановлення пакета, перевірте, чи той самий Python використовується для запуску програми.
5. У командних проєктах обов'язково використовуйте requirements.txt.

---

1. Oktatási projektekhez érdemes **virtuális környezetet** használni.
2. Jobb igy írni:

python -m pip ...

mint egyszerűen csak pip ..., mert ez pontosabb.

3. Sok csomag telepítése előtt érdemes frissíteni a pip-et.

4. Ha a telepítés után a kód mégsem működik, ellenőrizzük, hogy ugyanazt a Python-verziót használjuk-e futtatásra.

5.. Csoportos projektekben mindig használjunk requirements.txt fájlt.

Підсумок / Összegzés

pip — один із найважливіших інструментів у Python.
Без нього сучасна робота з бібліотеками практично неможлива. Студент повинен уміти:

- перевірити наявність pip,

- встановити пакет,
- оновити пакет,
- видалити пакет,
- переглянути список пакетів,
- працювати з requirements.txt,
- бажано використовувати віртуальне середовище.

A pip a Python egyik legfontosabb eszköze.
Nélküle a modern Python-fejlesztés könyvtárakkal gyakorlatilag elképzelhetetlen. A hallgatónak tudnia kell:

- ellenőrizni a pip meglétét,
- csomagot telepíteni,
- csomagot frissíteni,
- csomagot eltávolítani,
- a telepített csomagok listáját megtekinteni,
- requirements.txt fájllal dolgozni,
- és lehetőleg virtuális környezetet használni.

На зміст

Міні-довідник / Mini útmutató

python -m pip --version

python -m pip install numpy

python -m pip install --upgrade pip

python -m pip list

python -m pip show numpy

python -m pip uninstall numpy

python -m pip freeze > requirements.txt

python -m pip install -r requirements.txt

python -m venv venv

venv\Scripts\activate

deactivate

На зміст

Популярні модулі Python/ Népszerű Python modulok

Основні бібліотеки для навчання та практики

Модуль	**Для чого (UA)**	**Mire használjuk (HU)**
Math	базові математичні функції (sin, cos, sqrt)	alap matematikai függvények (sin, cos, sqrt)
Random	генерація випадкових чисел	véletlen számok generálása
Datetime	робота з датами і часом	dátum és idő kezelése
Os	робота з файлами та ОС	fájlok és operációs rendszer kezelése
Sys	параметри системи, аргументи запуску	rendszerparaméterek, programindítás
Time	затримки, вимірювання часу	időkezelés, késleltetés
Json	робота з JSON-даними	JSON adatok kezelése

Модуль	Для чого (UA)	Mire használjuk (HU)
Re	регулярні вирази (пошук у тексті)	reguláris kifejezések
collections	розширені структури даних	speciális adatszerkezetek
Itertools	комбінаторика, ітерації	kombinatorika, iterációk

Наукові та навчальні бібліотеки

Модуль	Для чого (UA)	Mire használjuk (HU)
Numpy	масиви, чисельні обчислення	tömbök, numerikus számítások
Scipy	наукові обчислення	tudományos számítások
matplotlib	побудова графіків	grafikonok készítése
Pandas	таблиці, аналіз даних	táblázatok, adatelemzés
Sympy	символьна математика	szimbolikus matematika
statsmodels	статистичний аналіз	statisztikai elemzés

Робота з файлами та мережею

Модуль	Для чого (UA)	Mire használjuk (HU)
Requests	HTTP-запити (API, сайти)	HTTP kérések
Urllib	робота з URL	URL kezelés
Csv	читання/запис CSV-файлів	CSV fájlok kezelése
Pickle	серіалізація об'єктів	objektumok mentése
sqlite3	робота з базами даних SQLite	SQLite adatbázis

Візуалізація та GUI

Модуль	Для чого (UA)	Mire használjuk (HU)
tkinter	графічний інтерфейс (GUI)	grafikus felület
seaborn	статистичні графіки	statisztikai grafikonok
plotly	інтерактивні графіки	interaktív grafikonok

ШІ та сучасні бібліотеки

Модуль	Для чого (UA)	Mire használjuk (HU)
Sklearn	машинне навчання	gépi tanulás
tensorflow	нейронні мережі	neurális hálók
Torch	deep learning (PyTorch)	deep learning

Вбудовані vs зовнішні модулі / Beépített vs külső modulok

Модуль	Тип (UA)	Típus (HU)
Math	Вбудований	beépített
Random	Вбудований	beépített
Json	Вбудований	beépített
Numpy	зовнішній (pip)	külső (pip)
Pandas	зовнішній (pip)	külső (pip)
Matplotlib	зовнішній (pip)	külső (pip)
Requests	зовнішній (pip)	külső (pip)

Завдання до таблиць / Feladatok a táblázatokhoz

1. Визначте, які модулі потребують pip install. / Határozza meg, mely modulok igényelnek pip install-t.

2. Напишіть програму з використанням math і random. / Írjon programot math és random használatával.

3. Встановіть numpy і обчисліть суму масиву. / Telepítse a numpy-t és számolja ki egy tömb összegét.

4. Побудуйте графік з matplotlib. / Készítsen grafikont matplotlib segítségével.

На зміст

pip install не працює: причини і способи виправлення

(по матеріалам сайту https://ikt.in.ua/pip-install-ne-pratsyuje-10-prychyn-i-sposoby-vypravlennya/)

Коли **pip install не працює**, це часто виглядає як одна й та сама проблема, хоча реальна причина може бути зовсім різною. Іноді не вистачає прав доступу, іноді система бачить не той Python, а інколи стара версія pip або конфлікт у віртуальному оточенні. Нижче розглянемо 10 найпоширеніших причин і кроки, які допомагають у більшості випадків.

1. pip не встановлений або не знайдений

Найперше варто перевірити, чи взагалі доступний pip для тієї версії Python, яку ви використовуєте. У багатьох системах команда pip може бути не прив'язана до потрібного інтерпретатора.

Що зробити:

- Перевірте версію Python:

python --version

або

python3 --version.

- Перевірте pip через Python:

python -m pip --version

або

python3 -m pip --version.

- Якщо команда не знаходиться, спробуйте встановити pip разом із Python або через модуль **ensurepip**.

Найбезпечніший підхід — запускати pip саме як модуль:

python -m pip install package_name.

Так ви точно використовуєте правильне оточення.

2. Використовується не той інтерпретатор Python

На комп'ютері може бути кілька версій Python, і тоді pip встановлює пакети не туди, куди ви очікуєте. Це часта причина, коли пакет ніби встановився, але під час запуску програма його не бачить.

Як перевірити:

- Дізнайтесь шлях до Python:

which python

або

where python

у Windows.

- Перевірте, який pip прив'язаний до цього Python:

python -m pip --version.

- Запускайте встановлення через конкретний інтерпретатор, наприклад

python3.11 -m pip install package_name.

Якщо працюєте з проектом, краще одразу створювати окреме віртуальне середовище, щоб не плутати системний Python із проектним.

3. Не активоване віртуальне середовище

У Python-проєктах дуже часто використовують **venv.** Якщо середовище створене, але не активоване, команда

pip install

може змінювати зовсім іншу інсталяцію пакетів.

Що перевірити:

- Чи бачите ви в терміналі назву середовища в дужках, наприклад (venv).
- Чи активоване середовище перед встановленням пакетів.
- Чи не запустили ви pip у глобальному середовищі замість проектного.

Якщо середовище неактивне, активуйте його відповідною командою і повторіть встановлення. Це часто вирішує проблему одразу.

4. Проблеми з правами доступу

На Windows можуть виникати ситуації, коли термінал запущений без потрібних прав.

Як діяти:

- Уникайте глобального встановлення, якщо працюєте над проєктом.
- Використовуйте віртуальне середовище замість sudo pip install.
- За потреби запускайте термінал із підвищеними правами, але лише коли це справді потрібно.

Найкраща практика — встановлювати пакети локально в межах проєкту. Так простіше уникати конфліктів і випадкових поломок системного Python.

5. Застарілий pip

Стара версія pip може некоректно працювати з новими пакетами, залежностями або форматами збірки. Іноді оновлення самої утиліти вирішує проблему швидше, ніж будь-які інші дії.

Спробуйте:

- *python -m pip install --upgrade pip*
- Після оновлення повторіть встановлення потрібного пакета.

Якщо оновлення не проходить, перевірте, чи немає обмежень у мережі або конфлікту з поточним середовищем. Але в більшості випадків актуальний pip працює помітно стабільніше.

6. Проблеми з інтернетом або проксі

pip завантажує пакети з репозиторіїв, тому будь-які проблеми з мережею швидко стають причиною помилки. Особливо це помітно в корпоративних мережах, за VPN або через проксі.

Ознаки:

- Завантаження зависає.
- З’являються помилки тайм-ауту.
- pip не може дістатися до джерела пакетів.

Перевірте стабільність інтернету, спробуйте іншу мережу або уточніть налаштування проксі. Іноді допомагає повторити команду пізніше, якщо проблема тимчасова.

7. Невірна назва пакета

Трапляється, що сам пакет вказано неправильно: з помилкою в назві, зайвим символом або плутаниною між назвою проєкту і назвою модуля. У такому разі pip чесно повідомляє, що пакет не знайдено.

Що перевірити:

- Написання назви пакета в команді.
- Чи немає зайвих пробілів, лапок або нестандартних символів.
- Чи не шукаєте ви модуль за назвою імпорту замість назви пакета для встановлення.

Наприклад, назва для pip install і назва для import у коді не завжди збігаються. Це поширена дрібниця, яка забирає багато часу.

8. Конфлікт залежностей

Інколи pip не може встановити пакет через несумісні версії залежностей. Це особливо часто трапляється у великому проєкті, де багато бібліотек із різними вимогами.

Ознаки конфлікту:

- pip повідомляє про несумісні версії.
- Після встановлення один пакет ламає інший.
- Проблема виникає лише в конкретному проєкті.

Що допомагає: оновити або знизити версію проблемного пакета, переглянути список залежностей і встановлювати їх у чистому віртуальному середовищі. Якщо проєкт старий, іноді простіше відтворити робочий набір пакетів заново.

9. Пошкоджений кеш pip

pip зберігає тимчасові файли та кеш, і іноді саме вони спричиняють дивні помилки під час встановлення. Це не найчастіша причина, але її варто мати на увазі, якщо все інше вже перевірили.

Спробуйте:

- Очистити кеш:

  python -m pip cache purge.

- Повторити встановлення без кешу:

  python -m pip install --no-cache-dir package_name.

Якщо після цього пакет встановлюється нормально, значить проблема була саме у проміжних файлах. У таких випадках очищення кешу часто швидко повертає стабільність.

10. Проблеми з версією Python або сумісністю пакета

Не кожен пакет підтримує будь-яку версію Python. Іноді бібліотека розрахована лише на новіші інтерпретатори, а інколи, навпаки, ще не адаптована до найсвіжіших змін.

Що робити:

- Перевірити версію Python у проєкті.
- Спробувати іншу версію пакета.
- За потреби використати сумісне середовище з іншим релізом Python.

Якщо пакет відмовляється ставитися без зрозумілої причини, саме несумісність версій часто стає прихованим винуватцем. Тому корисно звіряти вимоги до Python ще до інсталяції.

Зведений список кроків аналізу, чому pip install не працює

Якщо немає часу довго діагностувати проблему, пройдіться коротким списком:

- Перевірте версію Python і pip.
- Запускайте встановлення через python -m pip.
- Активуйте віртуальне середовище.
- Оновіть pip.
- Перевірте мережу та проксі.
- Уточніть назву пакета.
- Спробуйте встановити без кешу.
- Перевірте конфлікти залежностей.
- Звірте сумісність версій Python і пакета.
- За потреби створіть нове чисте середовище.

Висновок

Коли **pip install не працює**, не варто одразу шукати складну причину. У більшості випадків проблема зводиться до одного з десяти сценаріїв: неправильне оточення, не той Python, відсутні права, стара версія pip або

конфлікт залежностей. Найкраща стратегія — перевіряти все по черзі, починаючи з найпростішого.

Якщо ви працюєте в Python регулярно, звичка використовувати віртуальні середовища, запускати pip через модуль і стежити за версіями пакунків заощадить багато часу. Це робить роботу з проєктами передбачуваною і значно зменшує шанс натрапити на типову помилку встановлення.

A pip telepítése nem működik: okok és javítási módok (a https://ikt.in.ua/pip-install-ne-pratsyuje-10-prychyn-i-sposoby-vypravlennya/ webhely anyagai alapján)

Amikor a pip telepítése nem működik, gyakran úgy tűnik, mintha ugyanaz a probléma lenne, de a valódi ok teljesen más lehet. Néha a nem megfelelő jogosultságok miatt van, néha a rendszer rossz Pythont lát, néha a pip egy régi verzióját, vagy ütközés van a virtuális környezetben. Az alábbiakban a 10 leggyakoribb okot és a legtöbb esetben segítő lépéseket vizsgáljuk meg.

1. A pip nincs telepítve vagy nem található

Először is ellenőrizni kell, hogy a pip elérhető-e a használt Python verzióhoz. Sok rendszeren előfordulhat, hogy a pip parancs nincs a megfelelő értelmezőhöz kötve.

Mit kell tenni:

• Ellenőrizze a Python verzióját:

python --version

vagy

python3 --version.

• Ellenőrizze a pip-et Pythonon keresztül:

python -m pip --version

vagy

python3 -m pip --version.

• Ha a parancs nem található, próbálja meg telepíteni a pip-et a Pythonnal együtt, vagy az ensurepip modulon keresztül.

A legbiztonságosabb megközelítés a pip modulként futtatása:

python -m pip install csomag_neve.

Így biztos lehet benne, hogy a megfelelő környezetet használja.

2. Rossz Python értelmezőt használ

Előfordulhat, hogy a Python több verziója is van a számítógépén, és a pip a várt helyre telepíti a csomagokat. Ez egy gyakori ok, amikor a csomag telepítve van, de a program indításakor nem látja.

Az ellenőrzés módja:

• Keresse meg a Python elérési útját:

which python

vagy

where python

Windows rendszeren.

• Ellenőrizze, hogy melyik pip van ehhez a Pythonhoz társítva:

python -m pip --version.

• Futtassa a telepítést egy adott értelmezőn keresztül, például

python3.11 -m pip install csomag_neve.

Ha egy projekten dolgozik, jobb, ha azonnal létrehoz egy külön virtuális környezetet, hogy ne keverje össze a rendszer Pythont a projekt Pythonjával.

3. Inaktív virtuális környezet

A Python projektek gyakran használják a venv parancsot. Ha a környezet létrejön, de nincs aktiválva, a

pip install

parancs teljesen más csomagtelepítést módosíthat.

Amit ellenőrizni kell:

• Látható-e a környezet neve zárójelben a terminálban, például (venv)?

• Aktiválva van-e a környezet a csomagok telepítése előtt?

• A pip parancsot a globális környezetben futtatta a projektkörnyezet helyett?

Ha a környezet inaktív, aktiválja a megfelelő paranccsal, és próbálja újra a telepítést. Ez gyakran azonnal megoldja a problémát.

4. Engedélyezési problémák

Windows rendszeren előfordulhatnak olyan helyzetek, amikor a terminál a szükséges engedélyek nélkül indul el.

Amit tenni kell:

• Kerülje a globális telepítéseket, ha egy projekten dolgozik.

• Használjon virtuális környezetet a sudo pip install helyett.

• Futtassa a terminált emelt jogosultságokkal, ha szükséges, de csak akkor, ha valóban szükséges.

A legjobb gyakorlat a csomagok helyi telepítése a projekten belül. Így könnyebb elkerülni a konfliktusokat és a véletlenszerű Python rendszerösszeomlásokat.

5. Elavult pip

Előfordulhat, hogy a pip egy régi verziója nem működik megfelelően új csomagokkal, függőségekkel vagy build formátumokkal. Előfordul, hogy maga a segédprogram frissítése gyorsabban megoldja a problémát, mint bármely más művelet.

Próbálja ki:

• python -m pip install --upgrade pip

• A frissítés után telepítse újra a kívánt csomagot.

Ha a frissítés sikertelen, ellenőrizze a hálózati korlátozásokat vagy az aktuális környezettel való ütközéseket. A legtöbb esetben azonban a jelenlegi pip észrevehetően stabilabban működik.

6. Internet- vagy proxyproblémák

A pip a csomagokat a tárolókból tölti le, így a hálózati problémák gyorsan a hiba okává válnak. Ez különösen vállalati hálózatokban, VPN mögött vagy proxyn keresztül észrevehető.

Tünetek:

• A letöltés lefagy.

• Időtúllépési hibák jelennek meg.

• A pip nem tudja elérni a csomagok forrását.

Ellenőrizze az internet stabilitását, próbáljon ki egy másik hálózatot, vagy adja meg a proxybeállításokat. Néha segít, ha később megismétli a parancsot, ha a probléma átmeneti.

7. Helytelen csomagnév

Előfordulhat, hogy maga a csomag van helytelenül megadva: hiba van a névben, egy felesleges karakter, vagy összekeveredik a projekt neve és a modul neve. Ebben az esetben a pip őszintén jelenti, hogy a csomag nem található.

Amit ellenőrizni kell:

• A csomag nevének helyesírása a parancsban.

• Nincsenek felesleges szóközök, idézőjelek vagy nem szabványos karakterek.

• Az import név alapján keres modult a telepítendő csomag neve helyett?

Például a pip install és az import név a kódban nem mindig egyezik. Ez egy gyakori apróság, ami sok időt vesz igénybe.

8. Függőségi ütközés

Előfordul, hogy a pip nem tud telepíteni egy csomagot a függőségek inkompatibilis verziói miatt. Ez különösen gyakori egy nagy projektben, ahol sok, eltérő követelményekkel rendelkező könyvtár van.

Az ütközés jelei:

• a pip inkompatibilis verziókat jelez.

• A telepítés után az egyik csomag összeomlik a másikkal.

• A probléma csak egy adott projektben jelentkezik.

Ami segít: Frissítse vagy állítsa alacsonyabb verzióra a problémás csomagot, tekintse át a függőségeket, és telepítse őket egy tiszta virtuális környezetbe. Ha a projekt régi, néha könnyebb újra létrehozni a csomagok működő készletét.

9. Sérült pip gyorsítótár

A pip ideiglenes fájlokat és gyorsítótárat tárol, és néha ezek okoznak furcsa hibákat a telepítés során. Ez nem a leggyakoribb ok, de érdemes szem előtt tartani, ha mindent ellenőriztünk.

• Törölje a gyorsítótárat:

python -m pip cache purge.

• Ismételje meg a telepítést a gyorsítótár nélkül:

python -m pip install --no-cache-dir csomag_neve.

Ha a csomag ezután normálisan települ, akkor a probléma a köztes fájlokban volt. Ilyen esetekben a gyorsítótár törlése gyakran gyorsan visszaállítja a stabilitást.

10. Problémák a Python verziójával vagy a csomag kompatibilitásával

Nem minden csomag támogatja a Python bármely verzióját. Néha a könyvtár csak újabb interpreterekhez készült, néha pedig éppen ellenkezőleg, még nem alkalmazkodik a legújabb változásokhoz.

Teendők:

• Ellenőrizze a Python verzióját a projektben.

• Próbáljon ki a csomag egy másik verzióját.

• Szükség esetén használjon kompatibilis környezetet egy másik Python kiadással.

Ha egy csomag látható ok nélkül nem települ, a verzió-inkompatibilitás gyakran a rejtett ok. Ezért hasznos a Python követelményeinek ellenőrzése a telepítés előtt.

A pip telepítés sikertelenségének elemzéséhez szükséges lépések összefoglalása

Ha sokáig nincs ideje diagnosztizálni a problémát, íme egy gyors lista:

• Ellenőrizze a Python és a pip verzióit.

• Futtassa a telepítést a python -m pip paranccsal.

• Aktiválja a virtuális környezetét.

• Frissítse a pip-et.

• Ellenőrizze a hálózatot és a proxyt.

• Adja meg a csomag nevét.

• Próbálja meg a gyorsítótár nélküli telepítést.

• Ellenőrizze a függőségi ütközéseket.

• Ellenőrizze a Python és a csomagverziók kompatibilitását.

• Szükség esetén hozzon létre egy új, tiszta környezetet.

Következtetés

Ha a pip telepítése nem működik, ne keressen azonnal összetett okot. A legtöbb esetben a probléma tíz forgatókönyv egyikére vezethető vissza: helytelen környezet, helytelen Python, hiányzó jogosultságok, régi pip verzió vagy függőségi ütközés. A legjobb stratégia az, ha mindent egymás után ellenőriz, a legegyszerűbbel kezdve.

Ha rendszeresen dolgozik Pythonban, a virtuális környezetek használatának szokása, a pip modulon keresztüli futtatása és a csomagverziók figyelése sok időt takaríthat meg. Ezáltal a projektekkel való munka kiszámíthatóbbá válik, és jelentősen csökken a tipikus telepítési hibák előfordulásának esélye.

На зміст

Використання комбінації клавіш Win+R

Для швидкого запуску команд зручно використовувати комбінацію клавіш Win+R. Windows відкриває діалогове вікно «Виконати» (Run), яке призначене для швидкого запуску програм, системних утиліт, відкриття папок та документів. Це прискорює доступ до прихованих налаштувань та інструментів адміністрування, минаючи багаторівневі меню.

Як використовувати:

1. Натисніть Win (клавіша із логотипом Windows) + R.

2. У полі, що з'явилося, введіть команду.

3. Натисніть клавішу Enter або кнопку OK.

Корисні команди для Win+R:

• cmd — запуск командного рядка.

• PowerShell — запуск Windows PowerShell.

• control — Відкриття класичної панелі керування.

• msconfig – конфігурація системи (автозавантаження, служби).

• regedit – редактор реєстру.

• appwiz.cpl — встановлення та видалення програм.

• devmgmt.msc — Менеджер пристроїв.

• services.msc — керування службами.

• taskmgr — Менеджер завдань.

• shutdown /r /t 0 — миттєве перезавантаження комп'ютера.

Інколи побачити результат виконання команди cmd неможливо, тому що сама команда виконується за долю секунди, і вікно (консоль) відразу завершує роботу. Щоб побачити результат, скористайтеся одним із цих способів:

- запустіть через командний рядок: Відкрийте меню «Пуск», введіть cmd, натисніть Enter. У вікні, що з'явиться, введіть вашу команду вручну - ***вікно залишиться відкритим***, і ви побачите весь текст відповіді.
- Додайте команду утримання (cmd /k): Натисніть Win + R, зітріть старий текст і введіть команду у форматі:
- cmd /k [Ваша_Команда]. Ключ /k змушує вікно залишатися відкритим після виконання.

Результат можна зберегти у файлі: якщо хочете отримати великий список (наприклад, запущених служб), виконайте команду, перенаправивши виведення в текстовий документ, так:

[Ваша_Команда] > result.txt.

Після завершення просто відкрийте створений файл result.txt.

A Win+R billentyűkombináció használata.

A parancsok gyors elindításához kényelmes a Win+R billentyűkombináció használata. A Windows megnyitja a Futtatás párbeszédpanelt, amely a programok, rendszer segédprogramok, mappák és dokumentumok gyors elindítására szolgál. Ez felgyorsítja a rejtett beállításokhoz és adminisztrációs eszközökhöz való hozzáférést, megkerülve a többszintű menüket.

Használat:

1. Nyomja meg a Win (Windows logó billentyű) + R billentyűkombinációt.

2. A megjelenő mezőbe írja be a parancsot.

3. Nyomja meg az Enter billentyűt vagy az OK gombot.

Hasznos parancsok a Win+R billentyűkombinációhoz:

• cmd — parancssor indítása.

• PowerShell — Windows PowerShell indítása.

• control — klasszikus vezérlőpult megnyitása.

• msconfig — rendszerkonfiguráció (indítás, szolgáltatások).

• regedit — beállításjegyzék-szerkesztő.

• appwiz.cpl — programok telepítése és eltávolítása.

• devmgmt.msc — Eszközkezelő.

• services.msc — szolgáltatások kezelése.

• taskmgr — Feladatkezelő.

• shutdown /r /t 0 — azonnali számítógép újraindítás.

Néha lehetetlen meglátni a cmd parancs eredményét, mert maga a parancs egy pillanat alatt végrehajtódik, és az ablak (konzol) azonnal bezárul. Az eredmény megtekintéséhez használja az alábbi módszerek egyikét:

• **parancssorból futtatva**: Nyissa meg a Start menüt, írja be a cmd parancsot, majd nyomja meg az Enter billentyűt. A megjelenő ablakban írja be manuálisan a parancsot - az ablak nyitva marad, és a válasz teljes szövegét látni fogja.

• **Tartóparancs hozzáadása (cmd /k).** Nyomja meg a **Win + R** billentyűkombinációt, törölje a régi szöveget, és írja be a parancsot a következő formátumban:

cmd /k [Saját_Parancs].

A /k billentyű kikényszeríti, hogy az ablak a végrehajtás után nyitva maradjon.

Az eredmény egy fájlba menthető: ha egy nagy listát szeretne kapni (például a futó szolgáltatásokról), futtassa a parancsot a kimenet szöveges dokumentumba való átirányításával, például így:

[Saját_Parancs] > eredmény.txt.

Utána egyszerűen nyissa meg a létrehozott *eredmény.txt* fájlt.

На зміст

Пропоновані джерела/ Javasolt források

1. **Python Official Documentation**

   Офіційна документація мови Python.

2. **Python Packaging User Guide**

   Документація щодо пакетів, модулів та менеджера пакетів pip.

3. **pip Documentation**

   Офіційна документація менеджера пакетів pip.

4. **PyPI – Python Package Index**

   Центральний репозиторій пакетів Python.

5. **Real Python**

   Освітній ресурс із прикладами та поясненнями для Python.

6. **W3Schools Python Tutorial**

   Базові навчальні матеріали з Python.

7. **GeeksforGeeks Python Programming Languag**

   Практичні приклади та пояснення з Python і pip.

Модулі Python та менеджер пакетів Pip/Python modulok és Pip csomagkezelő Методичні рекомендації до самостійної роботи з дисципліни «Сучасні мови програмування» для здобувачів першого (бакалаврського) рівня рівня вищої освіти денної та заочної форм навчання, освітня програма: А Освіта, А4.09 Середня освіта (Інформатика). Розробник: Йожеф Головач. – Берегове: ЗУУ ім. Ф.Ракоці ІІ, 2026, 59 с. (українською та угорською мовами).