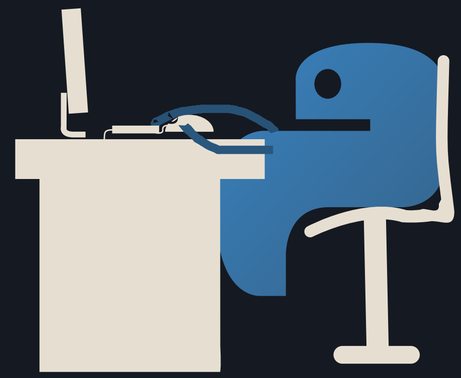
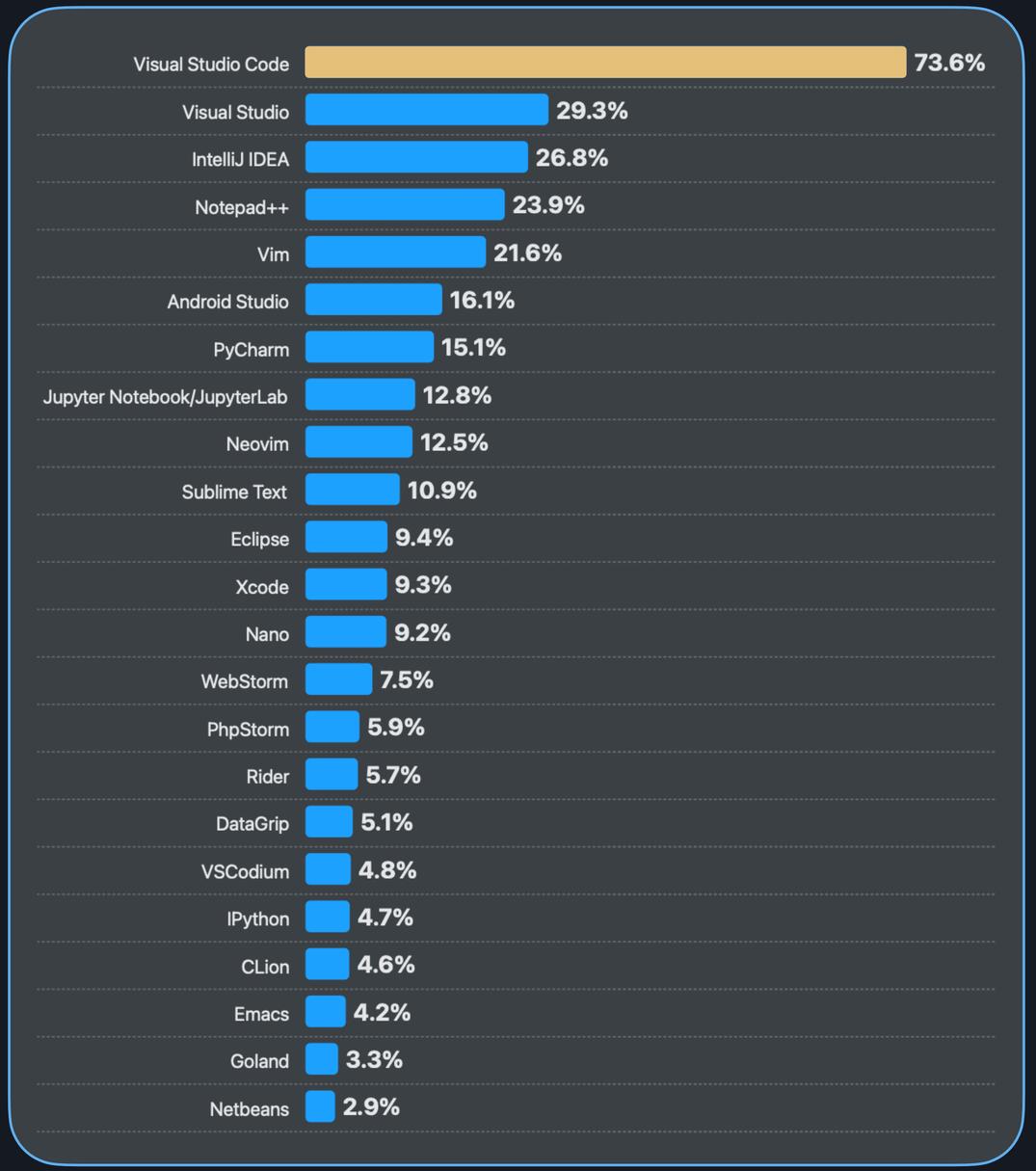
# Python

## Nyelvi alapok

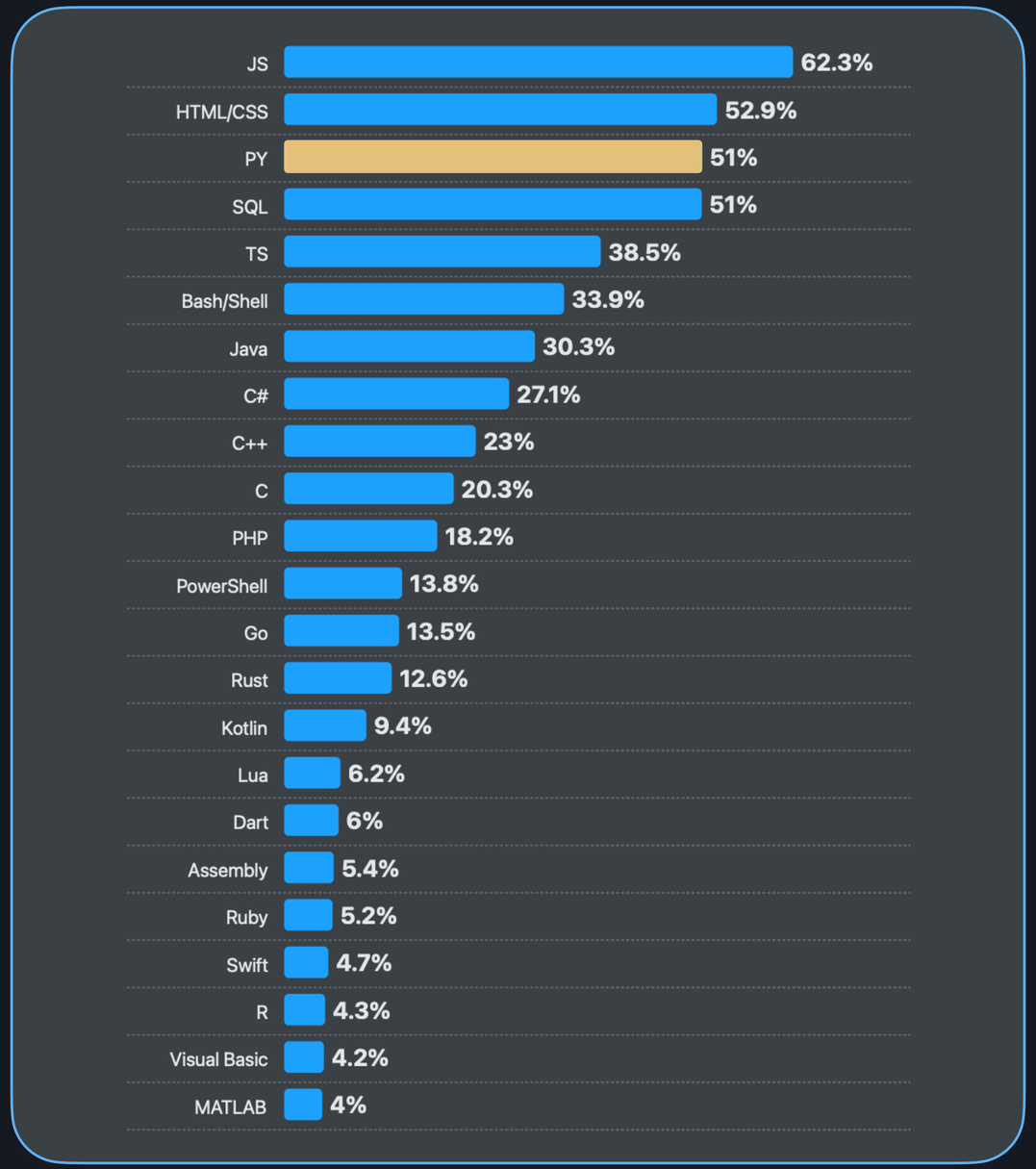


Csuzdi Domonkos, BME KJIT

# Legnépszerűbb szerkesztő




# Legnépszerűbb nyelv




https://survey.stackoverflow.co/2024/

# Loved vs. dreaded


https://survey.stackoverflow.co/2022/

# Bevezetés

## Miért Python?

nyílt forráskódú

magas szintű

ingyenes

objektumorientált

moduláris

gyengén típusos

cross-platform

interpretált[1]

[1] https://github.com/python/cpython

# Bevezetés

## VS Code

fájlok (workspace)

keresés

git

debug

kiegészítők

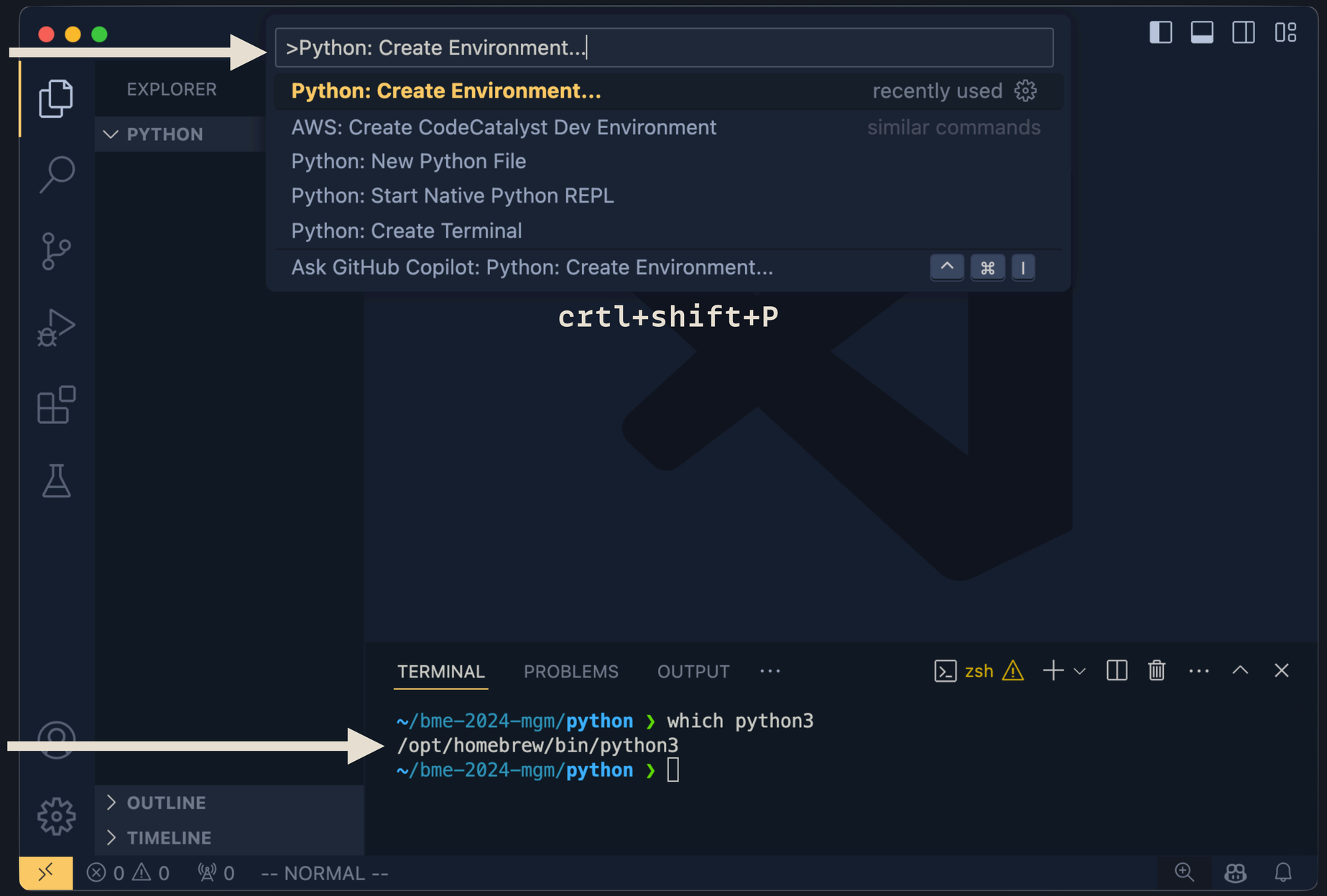

megnyitott

terminal + hibaüzenetek

crtl+J

# Python virtual environment

2. csináljunk egy lokált



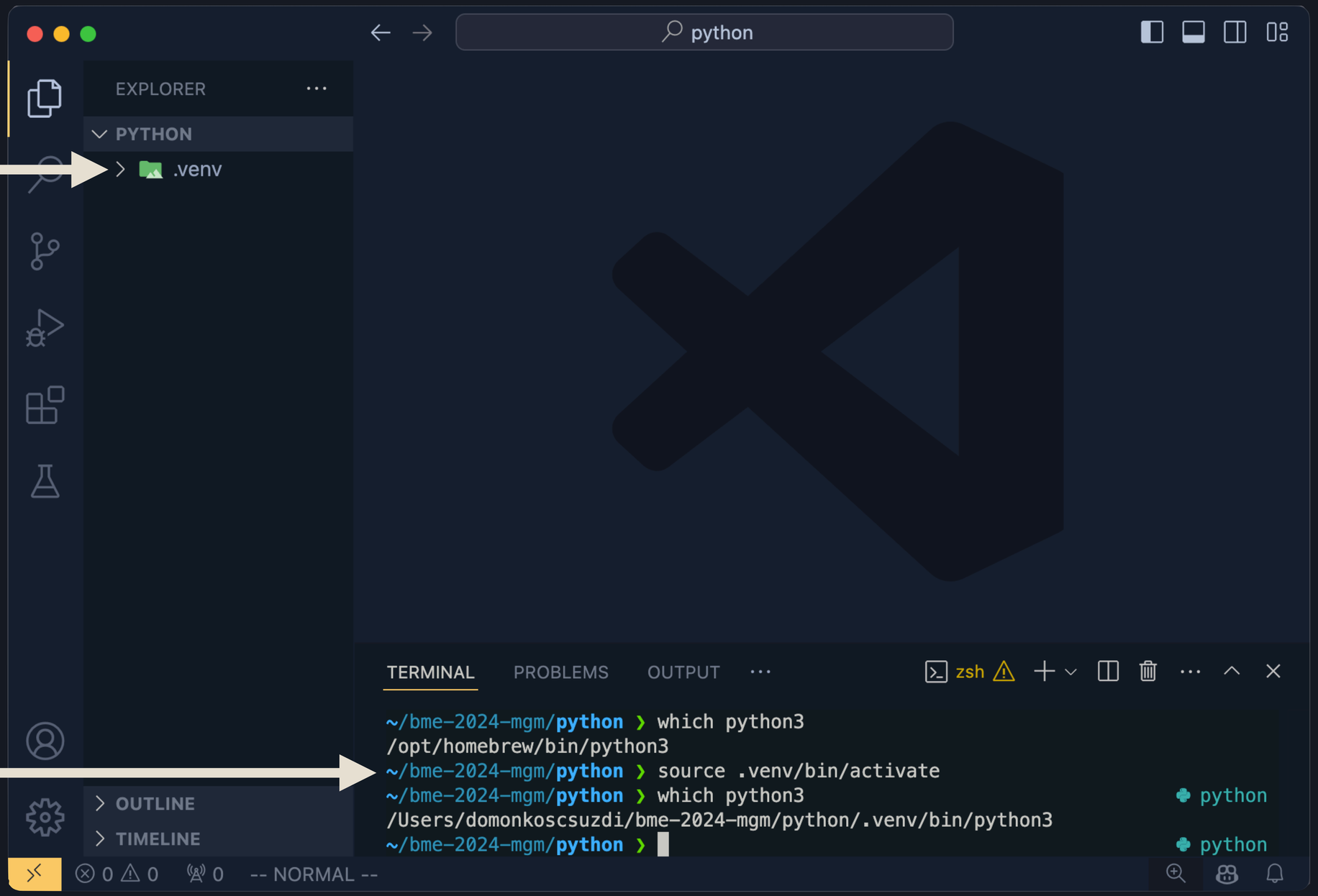

1. ez a globál Python

# Python virtual environment

Igazából csak egy mappa

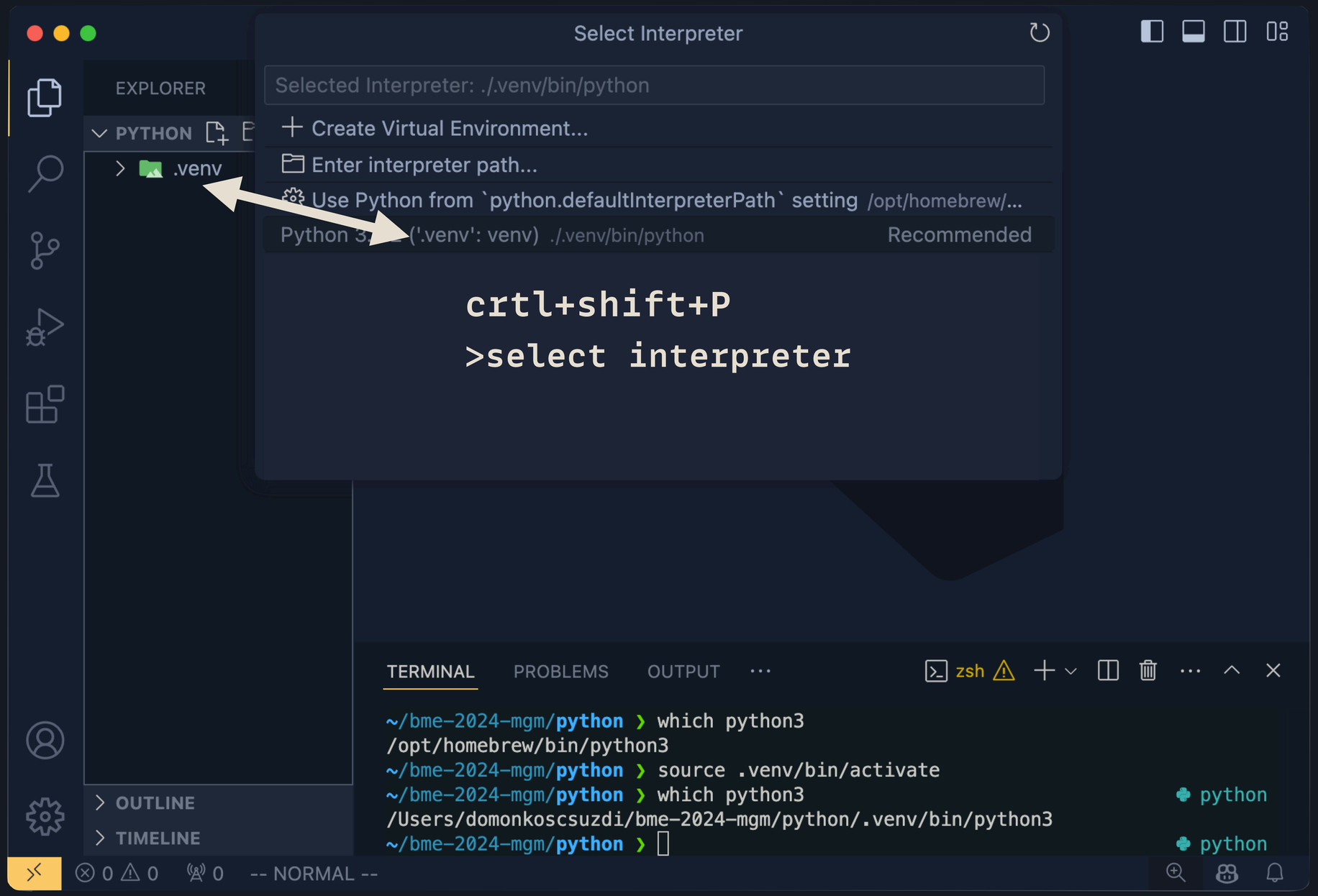
Itt lesznek a feltelepített csomagjaink

használjuk a terminálban



# Python interpreter

Ha az editorból
futtatunk,
akkor itt is
ki kell választani



# Python

## Read Evaluate Print Loop

terminal

```
~/bme-2024-mgm/python ❯ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> r = 3
>>> r
3
>>> 2*r*pi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pi' is not defined
>>> exit()
```

# Python csomagok/modulok

**Adatelemzés**
- numpy
- pandas
- matplotlib
- …

**Gépi tanulás**
- pytorch/tensorflow
- scikit-learn
- scipy
- …

**Webfejlesztés**
- django
- flask
- pyramid
- …

és még sok más...

# Csomagok használata, telepítése

## Beépített csomagok

The Python Standard Library[1]

- os: oprendszer spec.
- sys: parancssor
- time: pillanatnyi idő
- json: JSON fájlok
- math: matek
- antigravity: 
- …

## Külső csomagok

The Python Package Index[2]

(PyPi)

- letöltés és telepítés: `pip`
- példa (terminal): `pip install numpy`

### #TODO

- anaconda

## Saját csomagok

- átláthatóság
- modularitás
- feltölthetjük a PyPi-re[3]
- lokális importok

[1] https://docs.python.org/3/library/index.html [2] https://pypi.org [3] https://packaging.python.org/en/latest/tutorials/packaging-projects/

# Csomag import

**teljes csomagot**

```
import numpy
numpy.__version__
```

**csak egy részét**

```
import matplotlib.pyplot
```

**alias**

```
import numpy as np
np.__version__
```

**egy függvényt/változót**

```
from numpy import pi
print(pi)
```

**kevésbé olvasható a kód**

**minden fv-t,változót**

```
from numpy import *
print(pi)
```

**telerakjuk namespace-t**

# Beépített csomagok

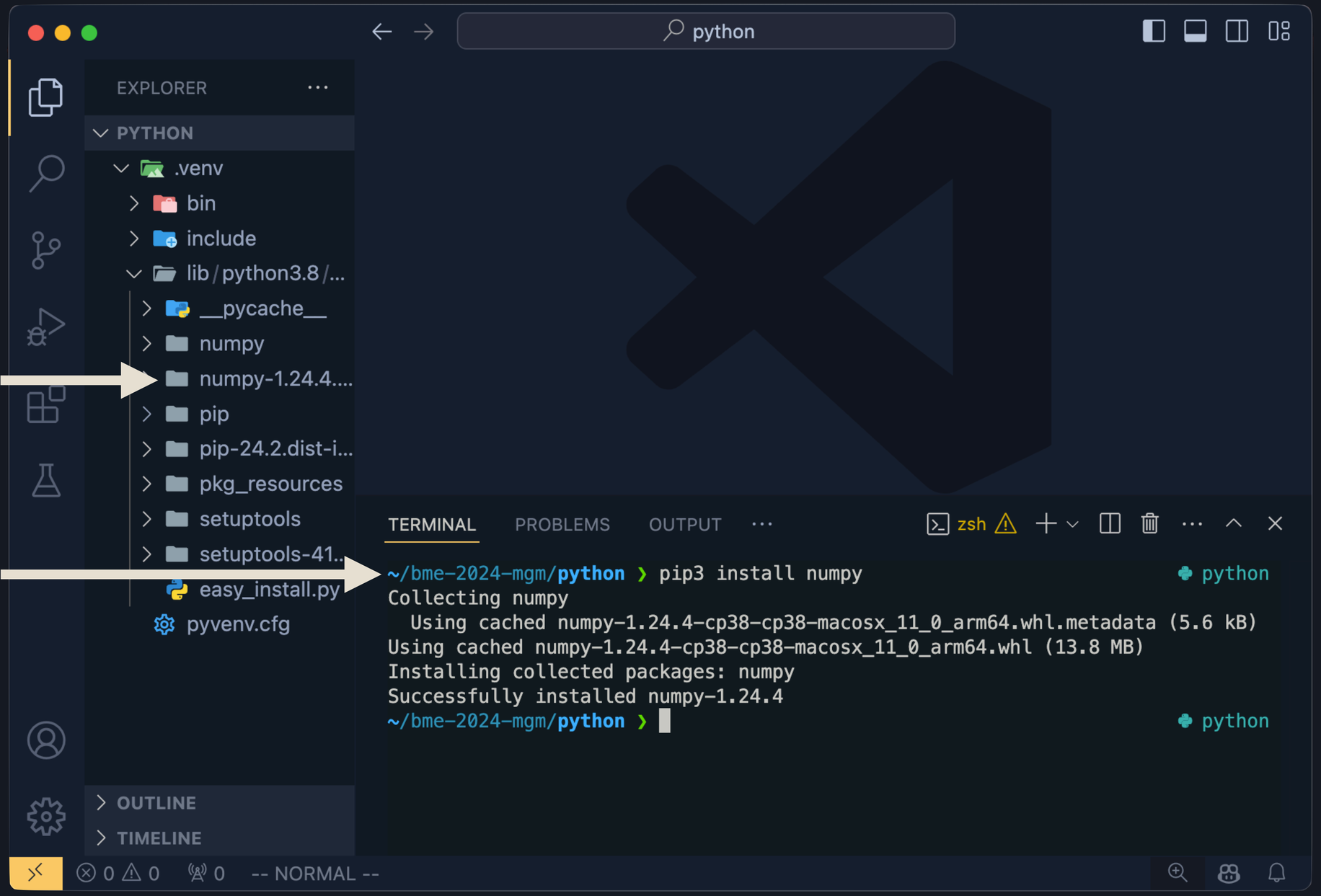
terminal

```
~/bme-2024-mgm/python ❯ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> r = 3
>>> import math
>>> 2*r*math.pi
18.84955592153876
>>> exit()
```

# Külső csomagok

2. itt is van

1. telepítjük



# Külső csomagok

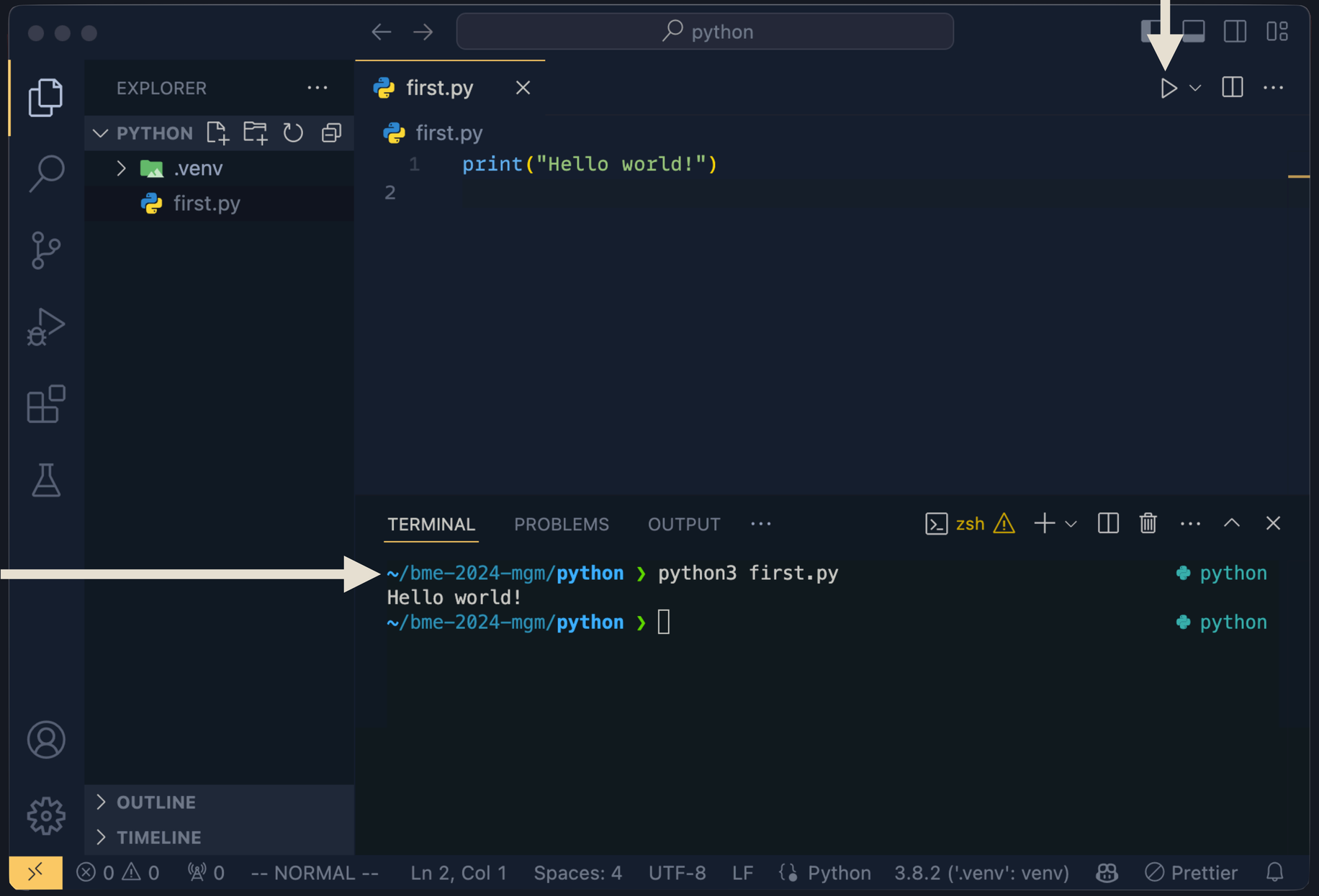
terminal

```
~/bme-2024-mgm/python ❯ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> r = 3
>>> import numpy as np
>>> 2*r*np.pi
18.84955592153876
>>> exit()
```

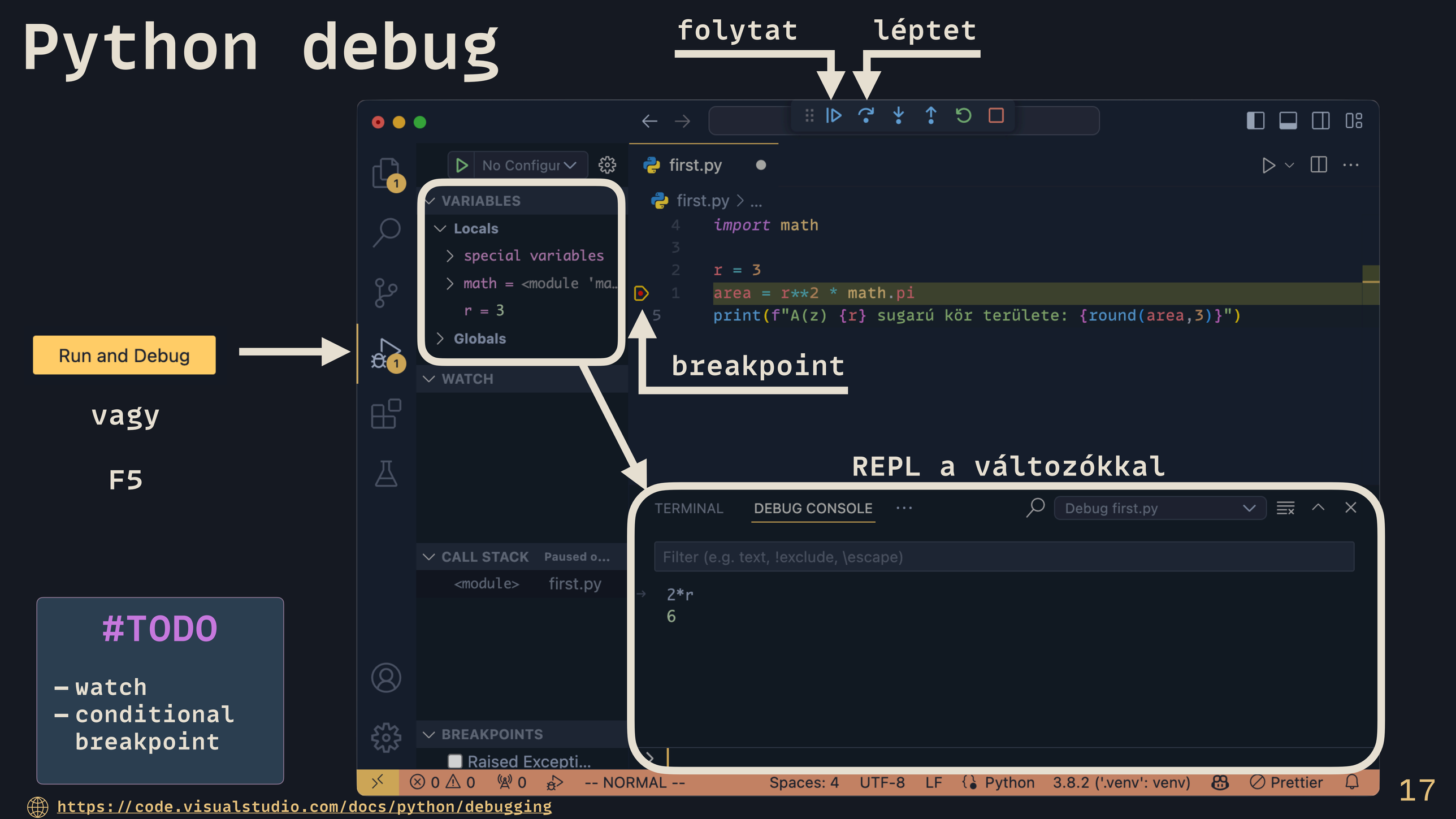
# Python modul (fájl)

2. vagy az editorból

1. futtathatjuk a terminálból



# Python debug




## #TODO

- watch
- conditional breakpoint

https://code.visualstudio.com/docs/python/debugging

# Adatstruktúrák

## list

changeable, ordered, allow duplicates

```python
my_list = [1, 2.0, 1j, "five", [4, 5]]
my_list.append(6)  # [1, 2.0, 1j, "five", [4, 5], 6]
removed = my_list.pop()  # removed: 6, my_list: [1, 2.0, 1j, "five", [4, 5]]
del my_list[0]  # [2.0, 1j, "five", [4, 5]], no return val

start = 1  # inclusive
end = 3  # exclusive
step = 2
new_list = my_list[start:end:step]  # [1j]
reversed_list = my_list[::-1]  # [[4, 5], "five", 1j, 2.0]
```

### #TODO

- sort, sorted
- kapcsolat a stringekkel
- egyéb műveletek

# Adatstruktúrák

## kitérő: numpy array

### Motiváció

```
my_list = [1, 2, 3]
my_list *= 2  # [1, 2, 3, 1, 2, 3]
```

```
import numpy as np

my_array = np.array([1, 2, 3])
# or np.array(my_list)
my_array *= 2  # [2  4  6]
```

### Mátrixok, vektorok, transzponált

```
my_vector = np.array([5, 6])
my_vector.T  # [5 6], not as expected!
my_vector.shape  # (2,)
my_vector.ndim  # 1
```

```
my_vector = np.array([[5, 6]])
my_vector.T  # [[5], [6]]
my_vector.shape  # (1,2)
my_vector.ndim  # 2
my_matrix = np.array([[1, 2], [3, 4]])
my_matrix @ my_vector.T  # [[17], [39]]
```

https://numpy.org/doc/stable/user/numpy-for-matlab-users.html

# Adatstruktúrák

## tuple

**unchangeable, ordered, allow duplicates**

```
my_tuple = (1, 2, 3)
a, b, c = my_tuple  # unpacking, a: 1, b: 2, c: 3
a, _, _ = my_tuple  # do not waste memory
a, *rest = my_tuple  # a: 1, rest: [2, 3]
my_tuple[0]  # 1
my_tuple[1] = 4  # unchangeable! → TypeError

# tuple with one element
new_tuple = (1)  # 1, does not work!
new_tuple = 1,  # (1,)
```

**#TODO**

- függvények hol használják?
- enumerate()

# Adatstruktúrák

## dictionary

changeable, ordered, no duplicates

```python
my_string = "hello"

char_dict = {}
for char in my_string:
    if char in char_dict:
        char_dict[char] += 1
    else:
        char_dict[char] = 1
print(char_dict)  # {'h': 1, 'e': 1, 'l': 2, 'o': 1}

del char_dict["l"]  # {'h': 1, 'e': 1, 'o': 1}
for key,value in char_dict.items():
    print(key, value) # h 1, e 1, o 1
```

**#TODO**

- melyik Py. verzió óta ordered?

# Adatstruktúrák

## set

**unchangeable[1], unordered, no duplicates[1]**

### Alapok

```
my_set = {1, 2, 3}
my_set.add(4)  # {1, 2, 3, 4}
my_set.remove(3)  # {1, 2, 4}
my_set[0]  # unordered → TypeError
```

### Apró trükk

```
# discard duplicates
my_list = [1, 2, 2, 4, 5]
unique_numbers = len(list(set(my_list))) # 4
```

**#TODO**

- set halmazműveletek

### Miért használjuk őket?

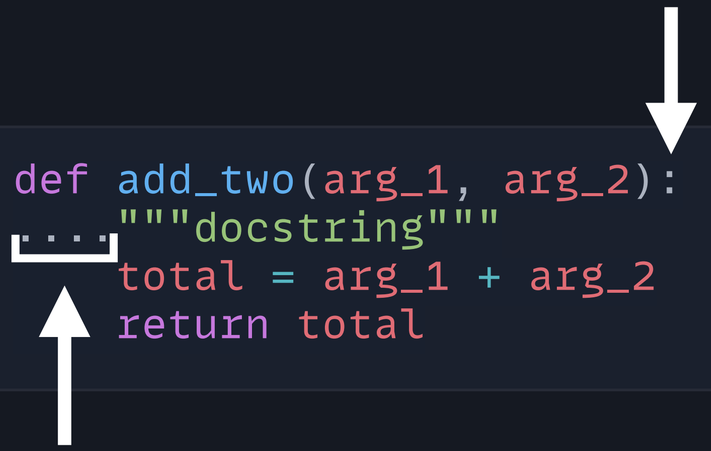
$\mathcal{O}(1)$ vs. $\mathcal{O}(n)$

```
import time

my_list = list(range(1000000))
my_set = set(my_list)

start = time.time()
999999 in my_list
print(f"List lookup: {time.time() - start}")
# 0.007 s

start = time.time()
999999 in my_set
print(f"Set lookup: {time.time() - start}")
# 2.86e-6 s
```

[1] https://www.w3schools.com/python/python_sets.asp

# Függvények

```
def add_two(arg_1, arg_2):
    """docstring"""
    total = arg_1 + arg_2
    return total
```



**4 db space/1 db tab**

```
def add_nums(arg_1: int, *args: int, **kwargs) → int:
    """
    Adds one or more numbers and prints
    any additional keyword arguments.

    Args:
        arg_1 (int): The first number.
        *args (int): Additional numbers to add.
        **kwargs: Additional keyword arguments.

    Returns:
        int: The sum of all provided numbers.
    """
    total = arg_1 + sum(args)
    print(f"Additional keywords: {kwargs}")
    return total


print(add_nums(2, 3, 4, 5, extra1="val1", extra2="val2"))
# Additional keywords: {'extra1': 'val1', 'extra2': 'val2'}
# 14
```

## #TODO

- *args, **kwargs
- type hint
- if __name__ == “__main__”

https://www.geeksforgeeks.org/args-kwargs-python/

# Vezérlési szerkezetek

```python
x = 5
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")
```

```python
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 2:
        continue
    elif num == 3:
        pass
    elif num == 4:
        break
    else:
        print(num)
```

```python
i = 0
while i < 5:
    print(i)
    i += 1
```

## #TODO

- exception-ök (ZeroDivisionError)
- finally
- do-while
- match-case
- walrus operator :=

```python
try:
    x = 1 / 0
except Exception as e:
    print(f"Encountered a problem: {e}")
```

https://docs.python.org/3/tutorial/controlflow.html https://docs.python.org/3/tutorial/errors.html

# The Pythonic Way

## list comprehension, ternary operator

```python
squares = []
for x in range(5):
    squares.append(x**2)
```

```python
squares = [x**2 for x in range(5)]
```

```python
list(range(5))
# [0, 1, 2, 3, 4]
```


```python
even_squares = []
for x in range(5):
    if x % 2 == 0:
        even_squares.append(x**2)
```

```python
even_squares = [x**2 for x in range(5) if x % 2 == 0]
```

### #TODO

- nested list comprehension
- dictionary comprehension

### ternary operator

```python
num = 4
result = "Even" if num % 2 == 0 else "Odd"
print(result)  # Output: "Even"
```

# Objektumorientált programozás 1.

## alapok

### A Pythonban minden egy objektum

**objektum = adattagok + metódusok**

```
a = 3 + 4j # complex number
print(type(a)) # <class 'complex'>
real = a.real # property
imag = a.imag # property
conj = a.conjugate() # method
(komplex konjugált)
```

```
a = 3 + 4j
b = 2 + 5j
c = a.__add__(b)
# c = a + b (egyszerűbben)
```

# Objektumorientált programozás 2.

## osztály létrehozása

```
class Pose2D:
    “""A type to represent 2D pose (x, y, rot)"""

    def __init__(self, x, y, theta):
        self.x = x  # adattagok
        self.y = y
        self.theta = theta

    # metódusok
    def to_degrees(self):
        """Return the rotation angle in degrees."""
        return np.degrees(self.theta)

    def add(self, other):
        x = self.x + other.x
        y = self.y + other.y
        theta = self.theta + other.theta
        return Pose2D(x, y, theta)
```

```
pose_1 = Pose2D(1, 2, np.pi/4)
pose_2 = Pose2D(0.5, 0.5, np.pi/4)
pose_3 = pose_1.add(pose_2)
print(pose_3.x,pose_3.y,pose_3.theta)
```

**#TODO**

- “dunder” metódusok
- privát adattagok

# Objektumorientált programozás 2.

## osztály létrehozása

```python
class Pose2D:
    “""A type to represent 2D pose (x, y, rot)"""

    def __init__(self, x, y, theta):
        self.x = x  # adattagok
        self.y = y
        self.theta = theta

    # metódusok
    def to_degrees(self):
        """Return the rotation angle in degrees."""
        return np.degrees(self.theta)

    def add(self, other):
        x = self.x + other.x
        y = self.y + other.y
        theta = self.theta + other.theta
        return Pose2D(x, y, theta)
```

```python
pose_1 = Pose2D(1, 2, np.pi/4)
pose_2 = Pose2D(0.5, 0.5, np.pi/4)
pose_3 = pose_1.add(pose_2)
print(pose_3.x,pose_3.y,pose_3.theta)
```

# Objektumorientált programozás 2.

## osztály létrehozása

```python
class Pose2D:
    “""A type to represent 2D pose (x, y, rot)"""

    def __init__(self, x, y, theta):
        self.x = x  # adattagok
        self.y = y
        self.theta = theta

    # metódusok
    def to_degrees(self):
        """Return the rotation angle in degrees."""
        return np.degrees(self.theta)

    def add(self, other):
        x = self.x + other.x
        y = self.y + other.y
        theta = self.theta + other.theta
        return Pose2D(x, y, theta)
```

```python
pose_1 = Pose2D(1, 2, np.pi/4)
pose_2 = Pose2D(0.5, 0.5, np.pi/4)
pose_3 = pose_1.add(pose_2)
print(pose_3.x,pose_3.y,pose_3.theta)
```

# Objektumorientált programozás 3.

## öröklődés

```python
class Pose2D:
    def __init__(self, x, y, theta):
        self.x = x
        self.y = y
        self.theta = theta
    def to_degrees(self):
        ...
    def add(self, other):
        ...
```

```python
class Pose2DStamped(Pose2D):
    """2D pose with timestamp"""

    def __init__(self, x, y, theta, timestamp):
        super().__init__(x, y, theta)
        self.timestamp = timestamp

    def add(self, other):
        n_p = super().add(other)
        n_stamp = max(self.timestamp, other.timestamp)
        return Pose2DStamped(n_p.x, n_p.y, n_p.theta, n_stamp)
```

```python
pose_4 = Pose2DStamped(
    1,
    2,
    np.pi / 4,
    12.13)
pose_5 = Pose2DStamped(
    2,
    3,
    np.pi,
    23)
```

# Objektumorientált programozás 3.

## öröklődés: super()

```python
class Pose2D:
    def __init__(self, x, y, theta):
        self.x = x
        self.y = y
        self.theta = theta
    def to_degrees(self):
        ...
    def add(self, other):
        ...
```

```python
class Pose2DStamped(Pose2D):
    """2D pose with timestamp"""

    def __init__(self, x, y, theta, timestamp):
        super().__init__(x, y, theta)
        self.timestamp = timestamp

    def add(self, other):
        n_p = super().add(other)
        n_stamp = max(self.timestamp, other.timestamp)
        return Pose2DStamped(n_p.x, n_p.y, n_p.theta, n_stamp)
```

```python
pose_4 = Pose2DStamped(
    1,
    2,
    np.pi / 4,
    12.13)
pose_5 = Pose2DStamped(
    2,
    3,
    np.pi,
    23)
```

# Objektumorientált programozás 3.

## öröklődés: metódus öröklés

```python
class Pose2D:
    def __init__(self, x, y, theta):
        self.x = x
        self.y = y
        self.theta = theta
    def to_degrees(self):
        ...
    def add(self, other):
        ...
```

```python
class Pose2DStamped(Pose2D):
    """2D pose with timestamp"""

    def __init__(self, x, y, theta, timestamp):
        super().__init__(x, y, theta)
        self.timestamp = timestamp

    def add(self, other):
        n_p = super().add(other)
        n_stamp = max(self.timestamp, other.timestamp)
        return Pose2DStamped(n_p.x, n_p.y, n_p.theta, n_stamp)
```

```python
pose_4 = Pose2DStamped(
    1,
    2,
    np.pi / 4,
    12.13)
pose_5 = Pose2DStamped(
    2,
    3,
    np.pi,
    23)
print(pose_4.to_degrees())
```

# Objektumorientált programozás 3.

## öröklődés: metódus felüldefiniálás

```python
class Pose2D:
    def __init__(self, x, y, theta):
        self.x = x
        self.y = y
        self.theta = theta
    def to_degrees(self):
        ...
    def add(self, other):
        ...
```

```python
class Pose2DStamped(Pose2D):
    """2D pose with timestamp"""

    def __init__(self, x, y, theta, timestamp):
        super().__init__(x, y, theta)
        self.timestamp = timestamp

    def add(self, other):
        n_p = super().add(other)
        n_stamp = max(self.timestamp, other.timestamp)
        return Pose2DStamped(n_p.x, n_p.y, n_p.theta, n_stamp)
```

```python
pose_4 = Pose2DStamped(
    1,
    2,
    np.pi / 4,
    12.13)
pose_5 = Pose2DStamped(
    2,
    3,
    np.pi,
    23)
print(pose_4.to_degrees())
print(pose_4.add(pose_5))
```

meglévő logika

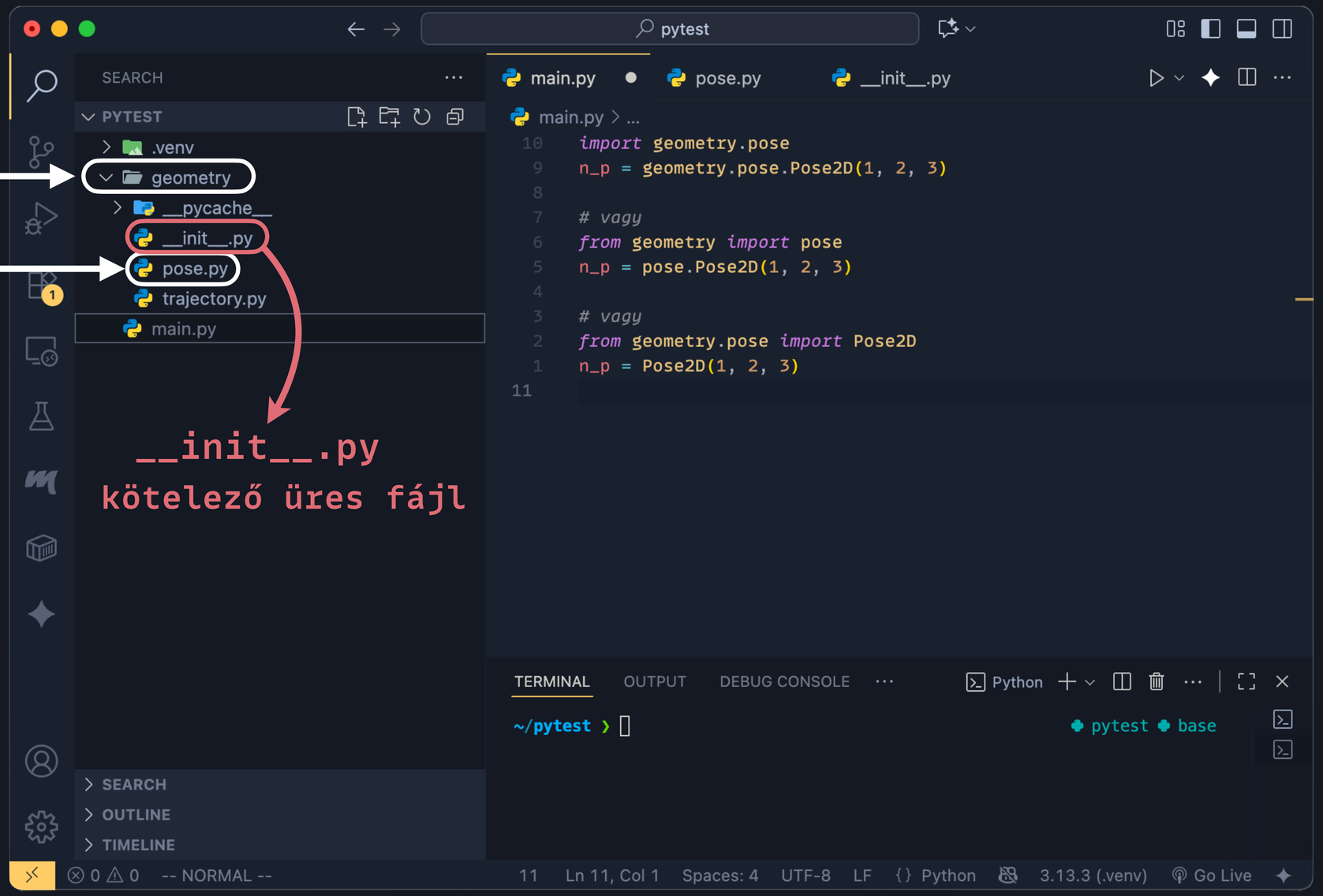
új logika

# Modulok és csomagok (package)

## Osztályok elkülönítése

Csomag (mappa) neve

Modul (fájl) neve



# Modulok és csomagok (package)

## importálás