

Języki skryptowe

1. Środowiska, shebang, środowiska wirtualne, instalacja pakietów.

Python

Obecnie wiele dystrybucji Linuksa zawiera najnowszego Pythona. Nawet, jak podają na stronie www.python.org, niektóre komputery z Windows.

Można sprawdzić, czy na danym komputerze jest już Python wpisując `py`, a jeśli nie zadziała to `python`.

W przypadku braku należy go pobrać i zainstalować (www.python.org/downloads/).

Dlaczego Python?

Python jest popularnym językiem programowania do pracy w domenach automatyzacji testów, zgrywania stron internetowych i uczenia maszynowego.

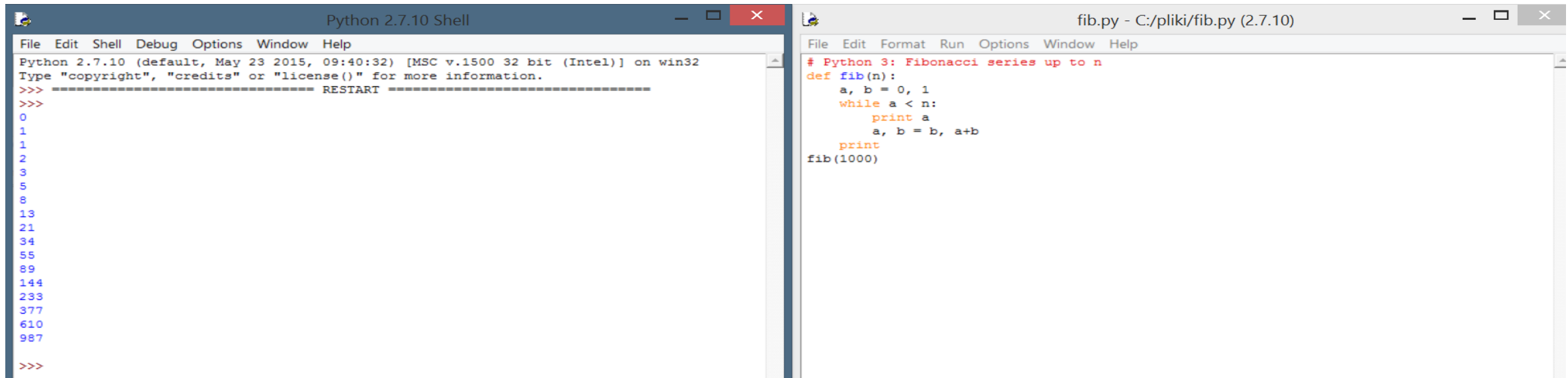
Jest prosty do nauki, wymusza formatowanie kodu a jego chyba największa siła to ogromna liczba przydatnych bibliotek i rozszerzeń.

Dlaczego Python?

Python to język, który może zrozumieć nawet osoba niebędąca programistą. Jest on wysoce **czytelny**, i pozwala skupić się na pisaniu **rozwiązania problemu**.

IDE

IDLE zawiera większość dystrybucji Pythona. Jest to proste środowisko z niewielką liczbą funkcji wspierających programistę.



The image shows two windows from the Python 2.7.10 IDE. The left window, titled 'Python 2.7.10 Shell', displays the output of a script that prints the first 15 Fibonacci numbers. The right window, titled 'fib.py - C:/pliki/fib.py (2.7.10)', shows the source code of the script.

```
Python 2.7.10 Shell
File Edit Shell Debug Options Window Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
>>>
```

```
fib.py - C:/pliki/fib.py (2.7.10)
File Edit Format Run Options Window Help
# Python 3: Fibonacci series up to n
def fib(n):
    a, b = 0, 1
    while a < n:
        print a
        a, b = b, a+b
    print
fib(1000)
```

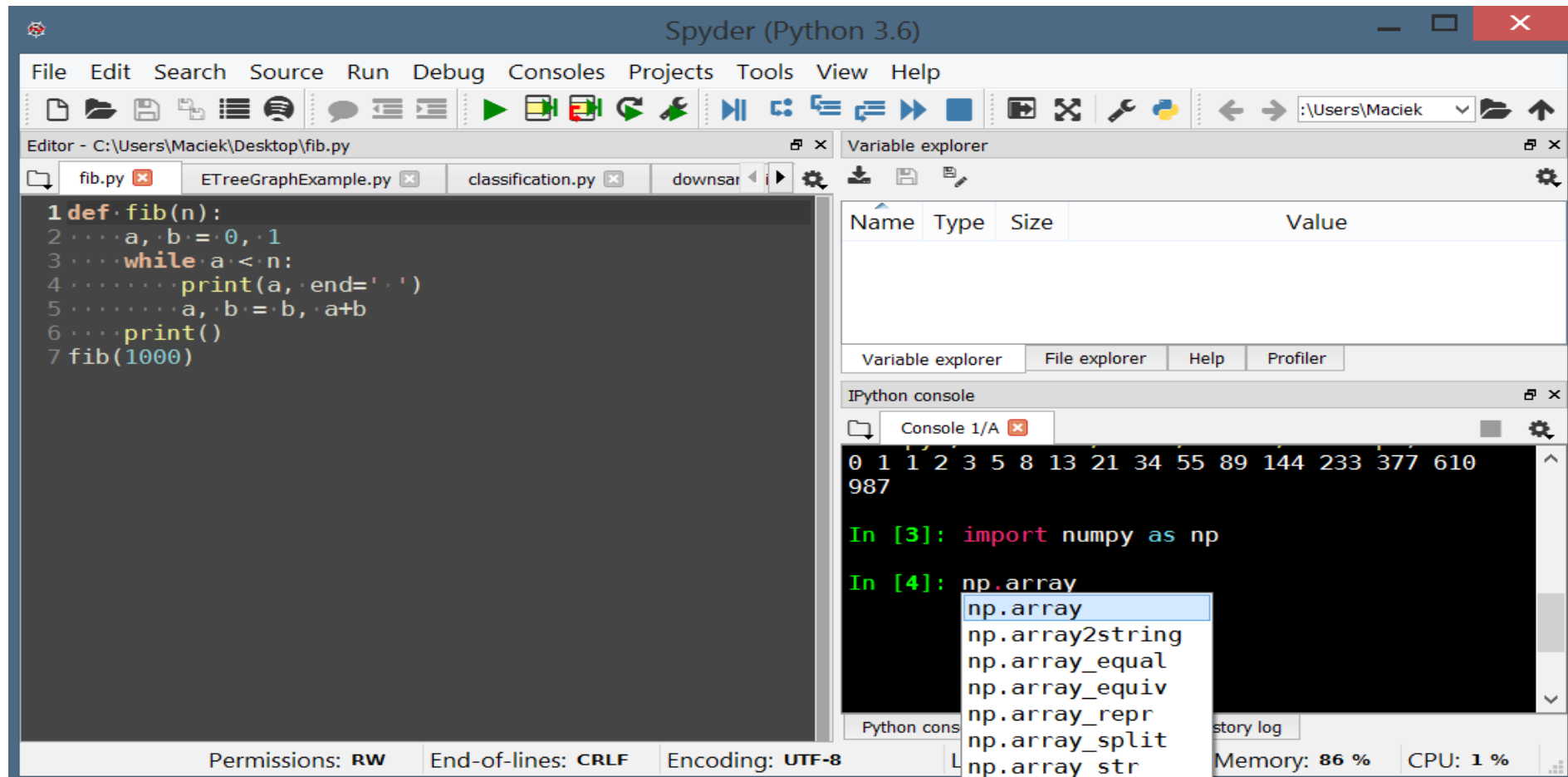
IDE

"**Anaconda**® jest menedżerem pakietów, menedżerem środowiska, dystrybucją języków Python / R oraz kolekcją ponad 7500 pakietów typu open source. Anaconda jest darmowa i łatwa w instalacji oraz oferuje bezpłatne wsparcie społeczności." *

* - źródło <https://docs.anaconda.com/anaconda/> [dostęp X 2020]

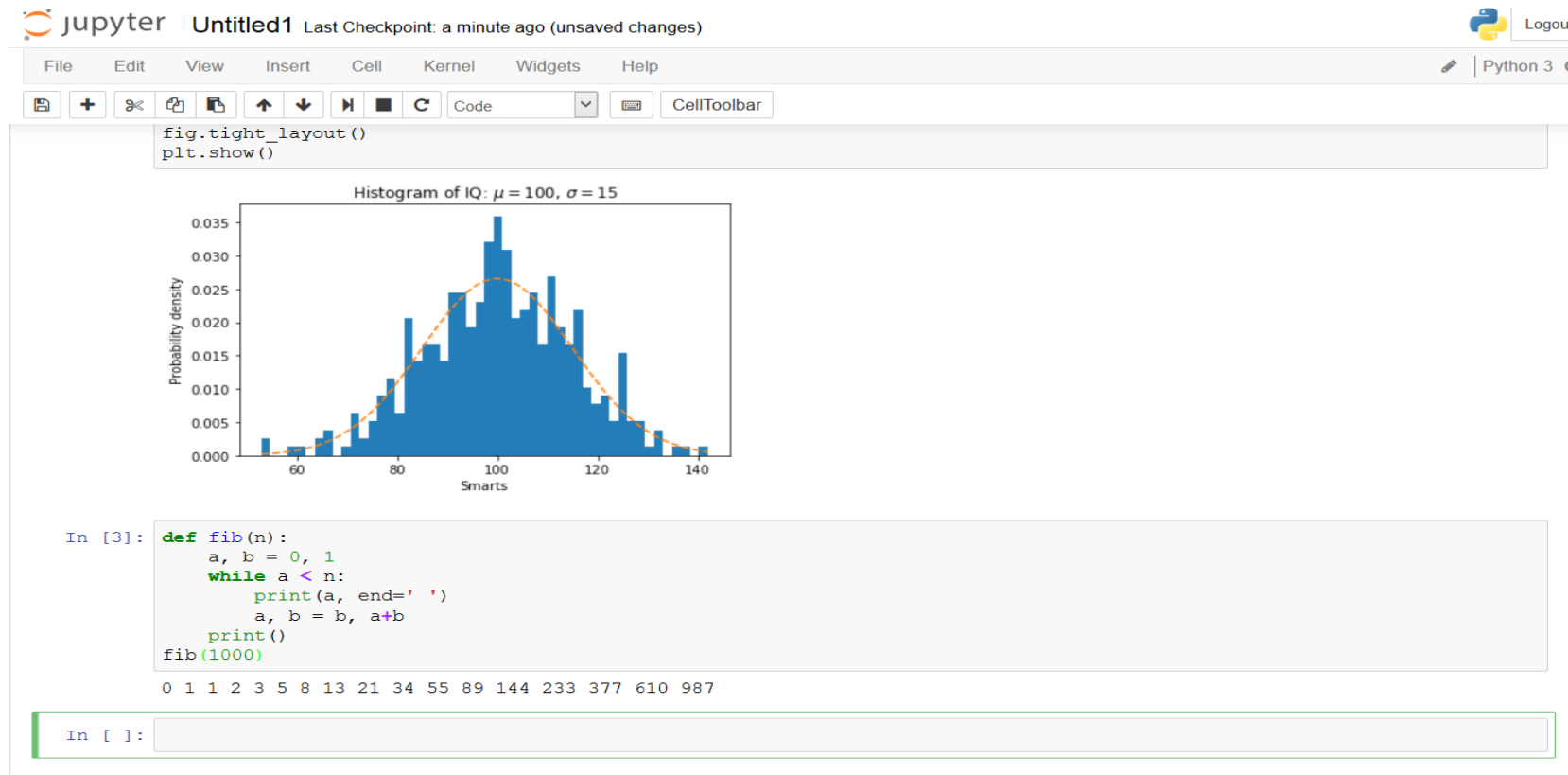
IDE

Spyder - IDE instalowane wraz z Anaconda (ewentualnie możliwość zainstalowania z poziomu Anaconda Navigator).



IDE

Jupyter Notebook – środowisko, które również instalowane jest wraz z Anacondą (ewentualnie możliwość zainstalowania z poziomu Anaconda Navigator). Można je też zainstalować osobno. Stworzone na potrzeby pracy naukowej.



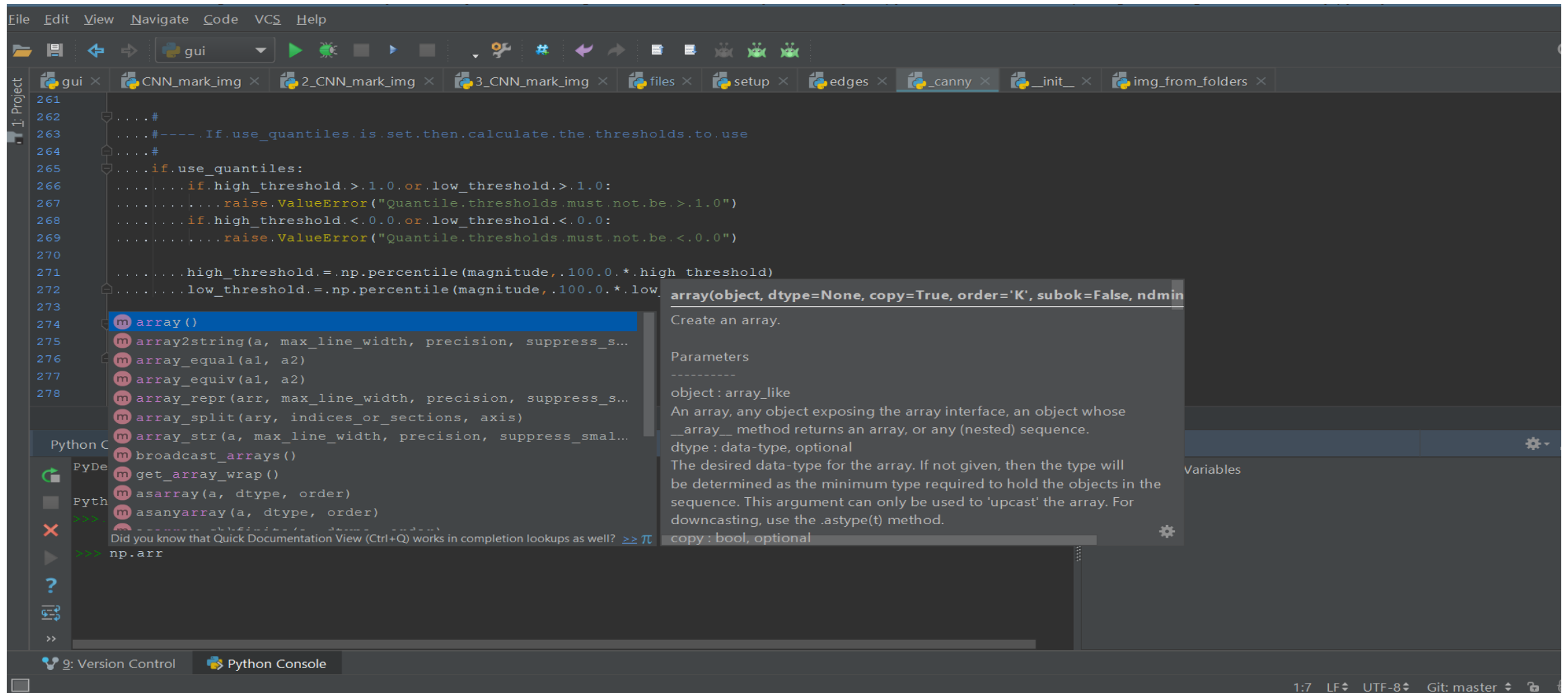
IDE

PyCharm został stworzony przez JetBrains, ludzi stojących za takimi IDE jak IntelliJ IDEA czy WebStorm.

Zawiera mnóstwo przydatnych funkcjonalności od takich jak kolorowanie składni, przez obsługę systemów kontroli wersji, aż po możliwość dostosowania środowiskach do swoich potrzeb.

IDE

PyCharm



IDE

`Eclipse` poszerzony o `PyDev` również może służyć jako IDE do Pythona.

Na podobnej zasadzie można rozszerzyć funkcjonalność `Visual Studio`.

`Visual Studio Code` również obsługuje Pythona (możliwość zainstalowania z poziomu Anaconda Navigator).

Python

Nie ma potrzeby używania separatora, takiego jak średnik, do oznaczenia końca instrukcji.

W Pythonie wcięcie zastępuje nawiasy (patrz C++, Java), aby pogrupować blok instrukcji.

Można użyć tabulatorów lub spacji do wcięcia kodu.

Python

Jednak wyraźnie wskazane jest trzymanie się niektórych reguł (PEP 8), takich jak użycie czterech spacji na poziom wcięcia.

PEP 8

Używa się 4 `spacji` na poziom wcięcia.

`Spacje` są preferowaną metodą wcięcia.

PEP 8

Ogranicza się wszystkie linie do maksymalnie 79 znaków.

PEP 8

Przejsście do nowej linii powinno nastąpić przed operatorem dwuargumentowym.

PEP 8

Puste linie:

- Definicje funkcji i klas najwyższego poziomu oddzielane są dwiema pustymi liniami.
- Definicje metod wewnątrz klasy są otoczone pojedynczą pustą linią.
- Używa się pustych linii w funkcjach, aby wskazać sekcje logiczne.

PEP 8

Import powinien zwykle odbywać się w osobnych wierszach, np.:

TAK:

```
import numpy  
import os
```

NIE:

```
import numpy, os
```

TAK:

```
from matplotlib.ticker import MaxNLocator, FuncFormatter
```

PEP 8

Cudzysłów

W Pythonie pojedynczy cudzysłów i podwójny cudzysłówów mają taką samą funkcjonalność. Nie ma nakazu, które używać. Należy określić swoje preferencje i trzymać się tego.

Białe znaki

Należy unikać nadmiarowych białych znaków.

Zaraz po i tuż przed nawiasem:

TAK:

```
def fun(par_1, {ind: elem})
```

NIE:

```
def fun( par_1, { ind: elem } )
```

Przed przecinkiem, średnikiem lub dwukropkiem:

TAK:

```
def fun(par_1, {ind: elem})
```

NIE:

```
def fun( par_1 , {ind : elem} )
```

PEP 8

Między nazwą funkcji a nawiasem, nazwą listy a nawiasem:

TAK:

```
def fun(par_1)
list_1[ind]
```

NIE:

```
def fun (par_1)
list_1 [ind]
```

W celu wyrównania:

TAK:

```
x = 1
zmienna_y = 3
```

NIE:

```
x =          1
zmienna_y = 3
```

PEP 8

W przypadku operatorów o różnych priorytetach otacza się spacjami ten o niższym:

TAK:

```
result_1 = 3*x + 1  
result_2 = (x+y) * (z+w)
```

NIE:

```
result_1 = 3 * x + 1  
result_2 = (x + y) * (z + w)
```

PEP 8

Nie otacza się spacjami symbolu podstawienia(=) w przypadku nazwanych argumentów(keyword argument):

TAK:

```
def fun(x, y=2):  
    return point(p1=x, p2=y)
```

NIE:

```
def fun(x, y = 2):  
    return point(p1 = x, p2 = y)
```

PEP 8

Komentarze:

Komentarze sprzeczne z kodem są gorsze niż brak komentarzy. Zawsze należy zadbać o aktualność komentarzy w przypadku zmiany kodu.

PEP 8

Komentarze:

Komentarze powinny być **pełnymi zdaniami**. Jeśli komentarz jest frazą lub zdaniem, jego pierwsze słowo powinno być pisane wielkimi literami, chyba że jest to identyfikator rozpoczynający się małą literą.

PEP 8

Komentarze:

Komentarze powinny być w języku angielskim. Chyba, że jest się pewnym, że nigdy nie będą czytane przez kogoś, kto nie mówi w danym języku.

Komentarze:

Komentarze zawarte w tej samej linii, co kod powinny być używane rzadko i rozsądnie. Korzystając z nich należy je oddzielić dwoma spacjami od kodu i po # wstawić jedną spację.

TAK:

```
age = age + 1    # To avoid division by zero.
```

NIE:

```
age = age + 1    # Incrementing age.
```

PEP 8

Konwencje nazewnictwa:

Nazwy klas powinny być pisane w konwencji CamelCase:

```
class Classification:
```

PEP 8

Konwencje nazewnictwa:

Nazwy funkcji powinny być pisane małymi literami, w razie potrzeby słowa są oddzielane znakami podkreślenia.

```
def load_np_data():
```

PEP 8

Konwencje nazewnictwa:

Stałe są pisane wielkimi literami ze znakiem podkreślenia oddzielającym słowa.

IMAGES_NUMBER

Python

W Pythonie **nie deklaruje się zmiennych**. Proste przypisanie wiąże nazwę z obiektem dowolnego typu.

Python jest **językiem interpretowanym**.

W przeciwieństwie do języków kompilowanych programy w języku Python **nie muszą się kompilować**(**compile**) i łączyć(**link**).

Python

Programowanie w języku Python jest **niezależne od platformy**. Oznacza to, że kod powinien działać na wszystkich obsługiwanych platformach.

Python

Programowanie obiektowe (OOP) jest wbudowane w język Python (poza tematyką tych zajęć).

shebang

Jeśli program napisany w Pythonie ma mieć możliwość uruchamiania jak normalny program należy dodać w pierwszej linii tzw. **shebang**. Na początku **#!** a następnie bezwzględną ścieżkę do programu interpretującego.

Unix:

```
#!/usr/bin/env python3
```

Uruchomienie:

```
./nazwa.py
```

Jeśli pojawi się komunikat o braku dostępu/uprawnień:

```
$ chmod a+x nazwa.py
```

shebang

Bez **shebang** można uruchomić program wpisując:

```
$ python3 nazwa.py
```

shebang

W so **Windows** można dodać ścieżkę dostępu do właściwej wersji Pythona. Program będzie wtedy można uruchomić podając jego nazwę w wierszu poleceń lub klikając na niego dwukrotnie. Czasami jednak może się zdarzyć, że uruchamiany program ma swoje **środowisko wirtualne**. W takim wypadku możemy albo dane środowisko wirtualne **aktywować**, albo dodać **odpowiedni shebang**.

```
#! C:\...\PycharmProjects\...\venv\Scripts\python
import tensorflow as tf
print('shebang!')
```

Wirtualne środowisko

Moduł używany do tworzenia środowisk wirtualnych i zarządzania nimi nosi nazwę **venv**.

Wirtualne środowisko

Aby **utworzyć środowisko wirtualne** określa się katalog, w którym ma zostać umieszczone, i uruchamia się moduł venv jako skrypt ze ścieżką do katalogu:

```
python3 -m venv wirtualne
```

Jeśli folder „wirtualne” nie istnieje to zostanie utworzony a wraz z nim katalogi zawierające kopię interpretera języka Python, bibliotekę standardową i różne pliki pomocnicze.

Wirtualne środowisko

Po utworzeniu środowiska wirtualnego można je **aktywować**.

W systemie Windows:

```
wirtualne\Scripts\activate.bat
```

W systemie Unix:

```
source wirtualne/bin/activate
```

Wirtualne środowisko

Po utworzeniu środowiska wirtualnego można je **aktywować**.

W systemie Windows:

wirtualne\Scripts\activate.bat

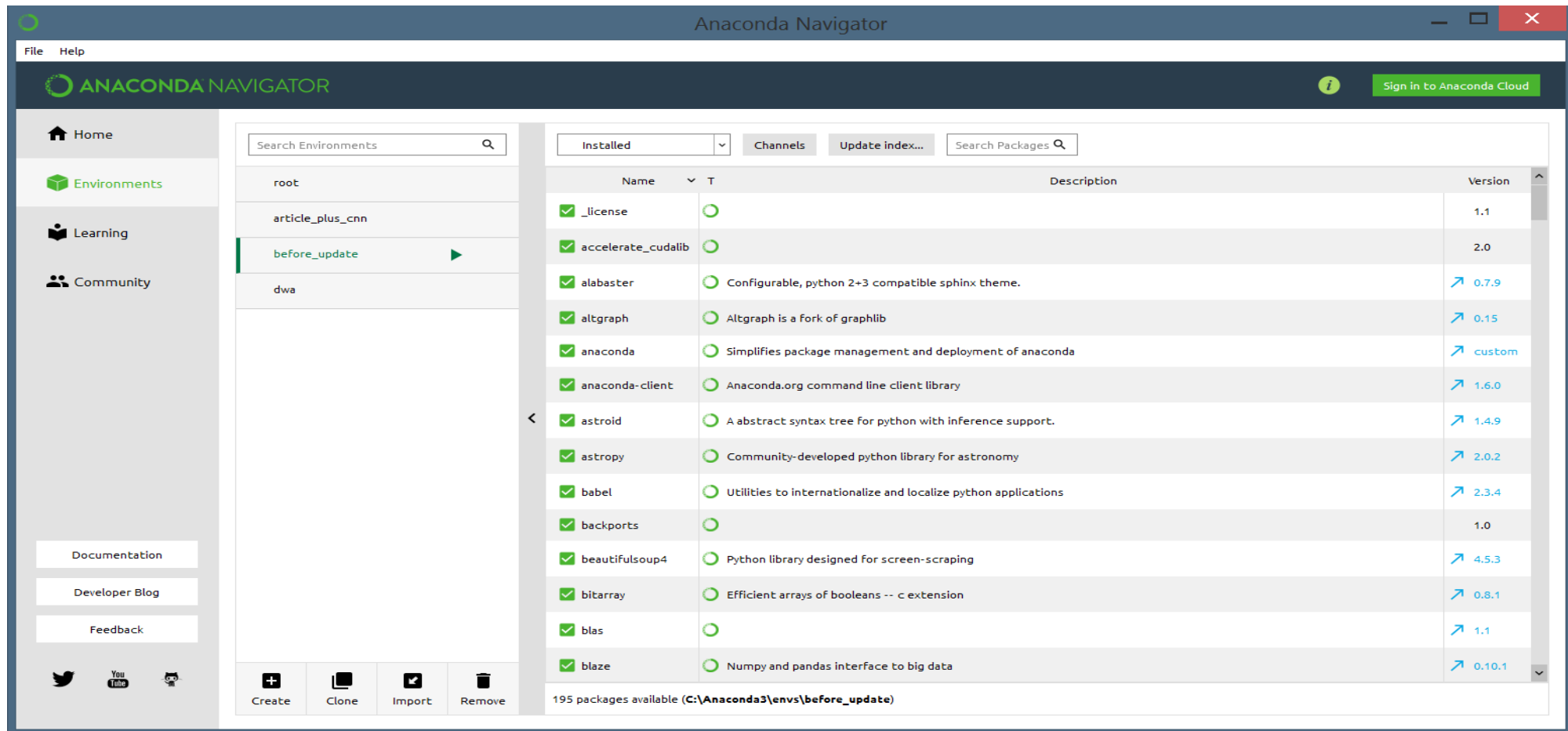
W systemie Unix:

source wirtualne/bin/activate

Po aktywacji przed znakiem zachęty w wierszu poleceń powinna się pojawić informacja w nawiasach. W celu wyłączenia środowiska wirtualnego należy wpisać: **deactivate**.

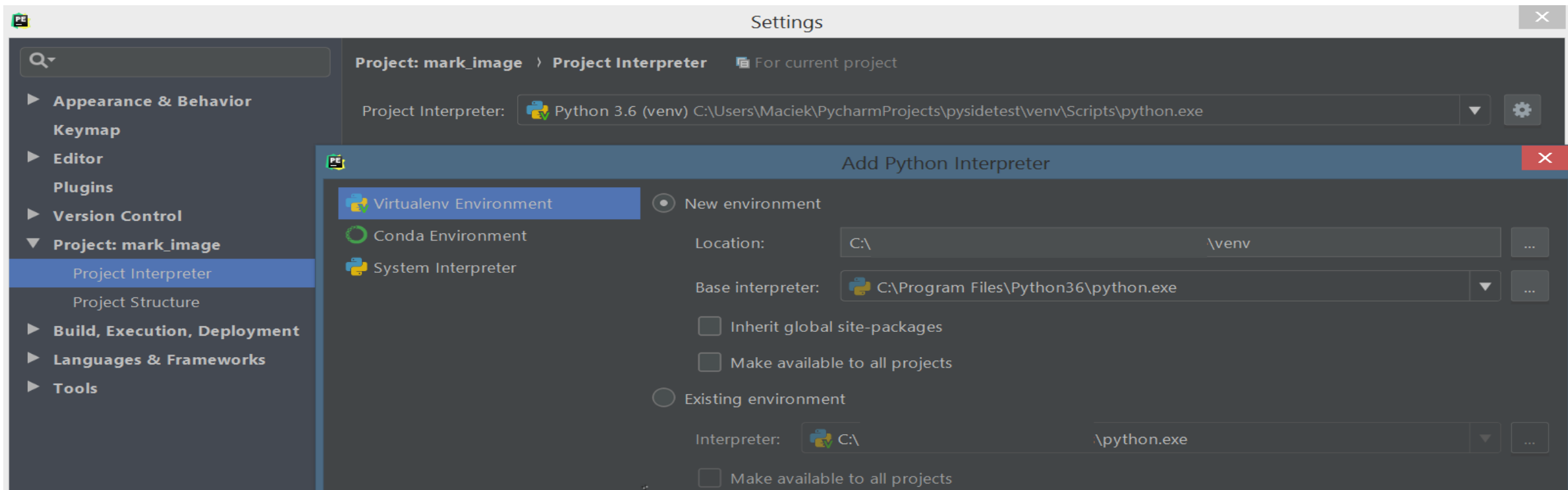
Wirtualne środowisko

Wirtualne środowisko w **Anacondzie** można stworzyć i modyfikować w nawigatorze:



Wirtualne środowisko

Wirtualne środowisko w **PyCharm** można stworzyć i modyfikować przechodząc do ustawień, następnie do interpretera projektu:



PIP

Można instalować, aktualizować i usuwać pakiety za pomocą programu o nazwie **pip**.

Wyszukiwanie:

```
$ pip search nazwa_pakietu
```

Instaluje najnowszą wersję pakietu:

```
$ pip install nazwa_pakietu
```

Instaluje wskazaną wersję pakietu:

```
$ pip install nazwa_pakietu==2.6.0
```

pip install --upgrade zaktualizuje pakiet do najnowszej wersji:

```
$ pip install --upgrade nazwa_pakietu
```

pip show wyświetli informacje o konkretnym pakiecie:

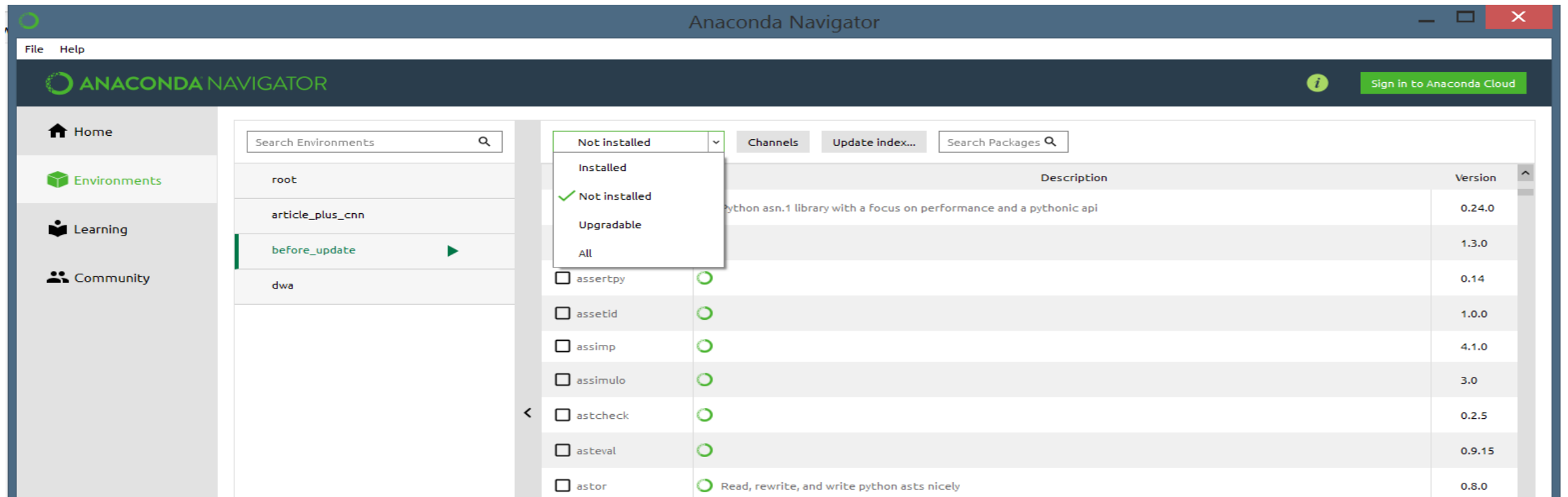
```
$ pip show nazwa_pakietu
```

Wyświetli wszystkie pakiety w środowisku wirtualnym:

```
$ pip freeze
```

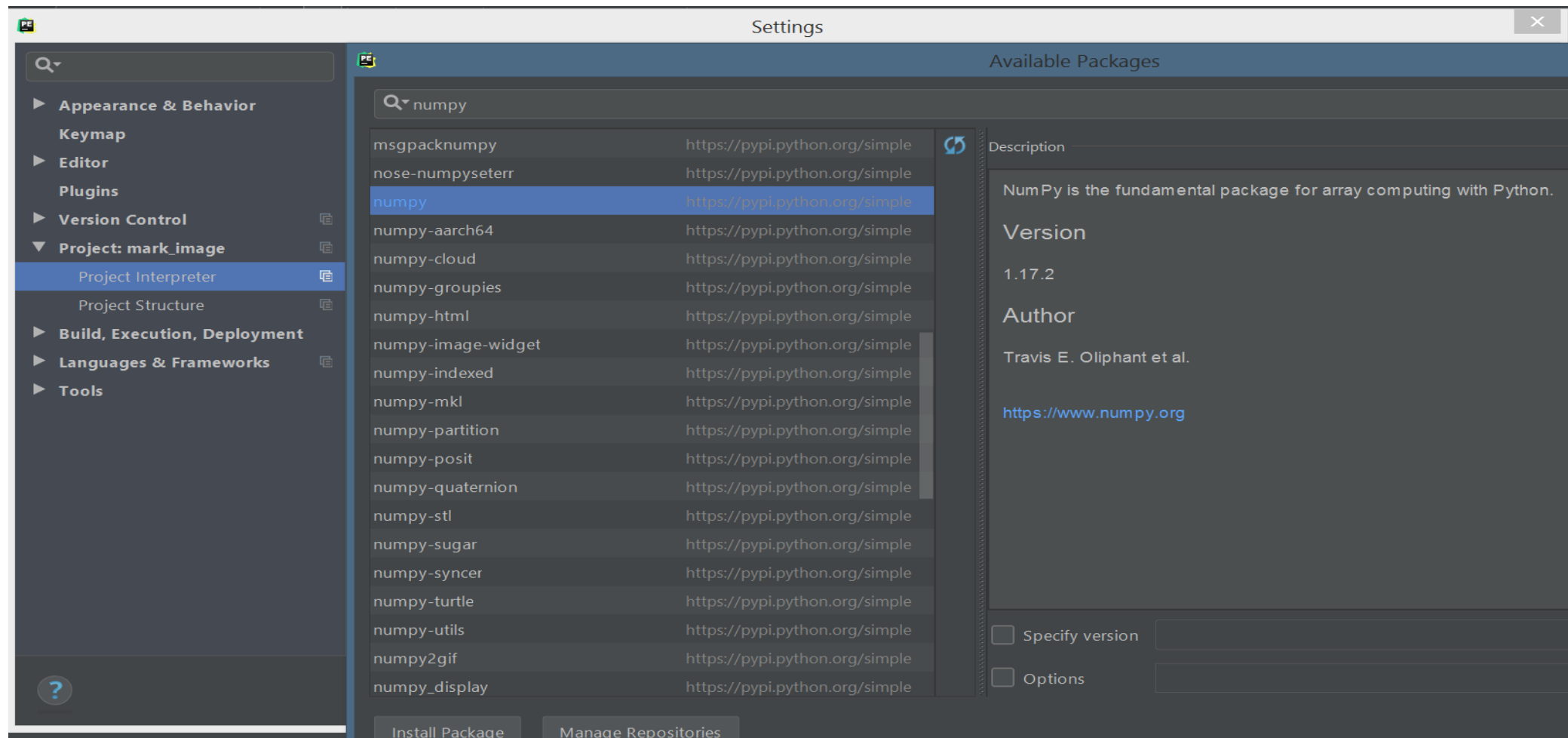
PIP

Pakiety w **Anacondzie** można instalować i modyfikować w nawigatorze:



PIP

Pakiety w **PyCharm** można instalować i modyfikować przechodząc do ustawień, następnie do interpretera projektu:



Zadanie: proszę powiązać wybrane środowisko z systemem kontroli wersji (VCS), na przykład z GIT, założyć darmowe konto, na którymś z portali, takich jak Bitbucket lub Github. Podczas pisania zadań robić migawki (commit) i przesyłać na serwer (push).

Środowiska, shebang, środowiska wirtualne, instalacja pakietów.

KONIEC