CEGŁADANYCH
DANE W CHMURZE

# Apache Spark na Windowsie czy to możliwe?

23.09.2021 Krzysztof Nojman


Witam, do tej pory pisałem o Databricks jako o super narzędziu do Big Data. Jest on niewątpliwie bardzo użyteczny, ale do tego potrzeba przeglądarki i dostępu do chmury publicznej, Azure, AWS lub GCP. A co jeśli chcesz zacząć przygodę ze Apache Spark na Windowsie bez wydawania kasy na chmurę. Mam dla Ciebie dobre wieści jest to możliwe, żeby używać Big Data na kilku rdzeniach. **Spark on Windows** ?? tak to możliwe. Dzięki genialnemu rozwiązaniu, jaki ma Spark da się go używać w opcji 'Local Mode' właśnie po to by można było go uruchomić lokalnie. W takim modelu działa Databricks Community.

Sparka można używać w konsoli co może się wydawać ułatwieniem, lecz ja bym proponował od razu zacząć w porządnym narzędziu. IntelliJ lub PyCharm. Dzięki temu

# Wymagania

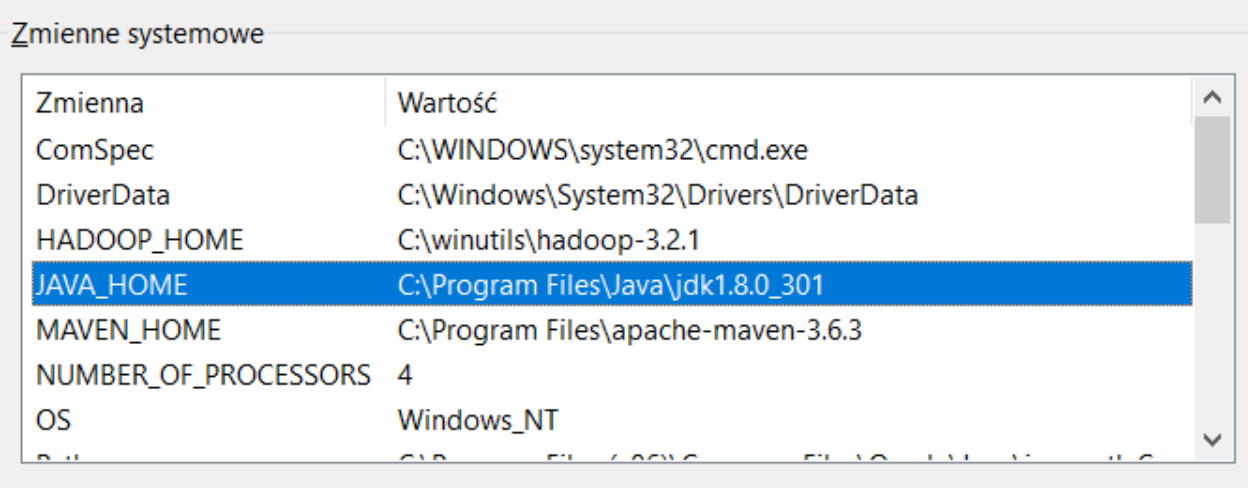
Oto lista narzędzi, którą trzeba zainstalować **Java**, **Python**, **Spark**, **winutils,** w zależności, w jakim języku chcesz pracować, dla Scali proponuję **IntelliJ** dla Pythona **PyCharm**. Z dodatkowych narzędzi do budowania mam zainstalowane sbt i maven to oczywiście jest świat Intellij. Sbt współpracuje lepiej ze scalą, więc na początek polecam prace z sbt. Jeśli chodzi o maven, to pracując w Azure ma on natywne wsparcie w pipelinach devopsowych (Artifacts), więc w Azure będzie lepszym wyborem.

## Java

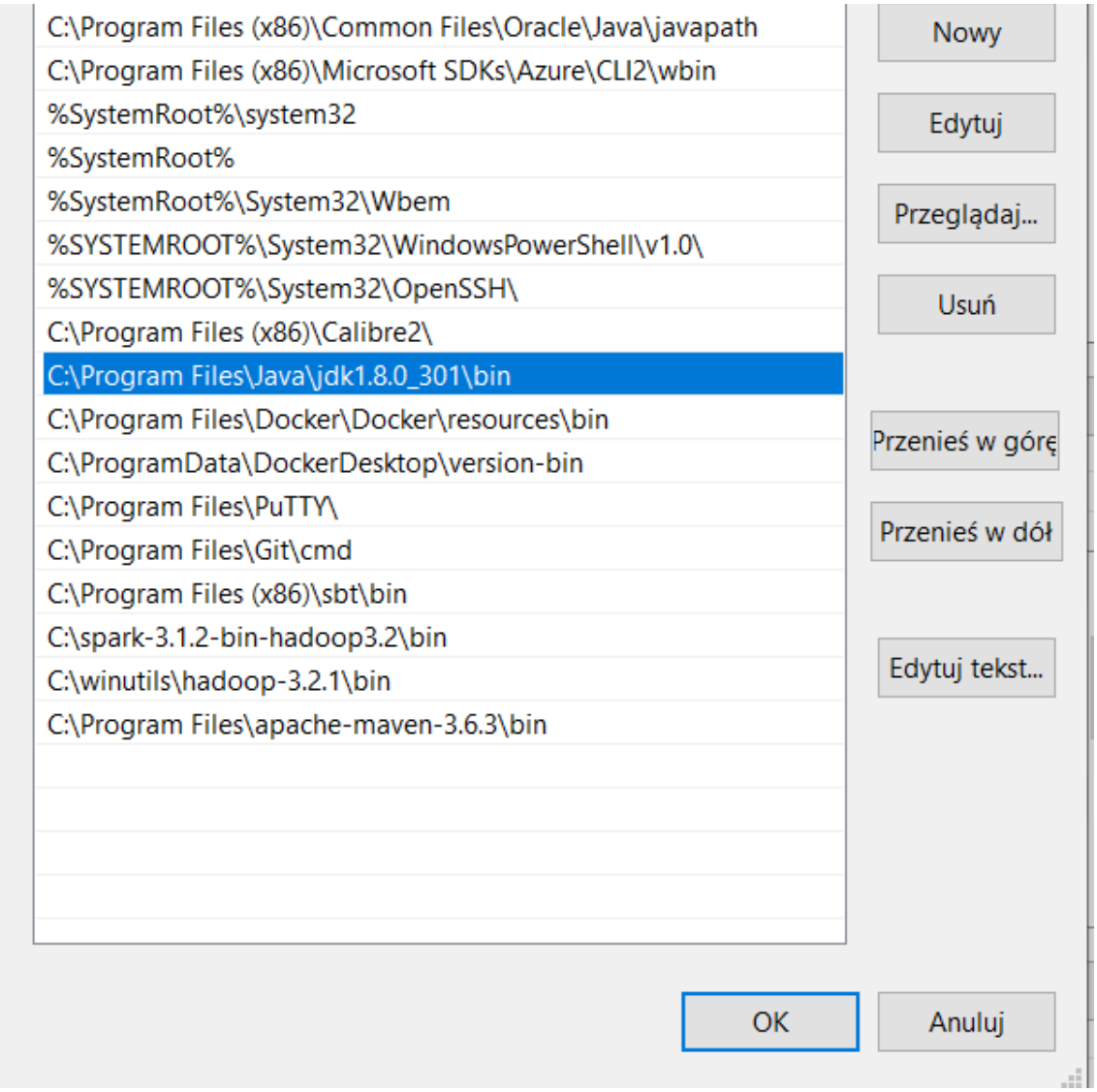
Jest bardzo małe prawdopodobieństwo że nie masz Java, jeśli tak jest to instrukcje znajdziesz tutaj. Wymagana wersja 1.8 lub 11. Po zainstalowaniu trzeba przejść kolejne kroki.

1. Upewnij się, że Java jest dodana do Zmiennych środowiskowych (‘Environment Variables’).

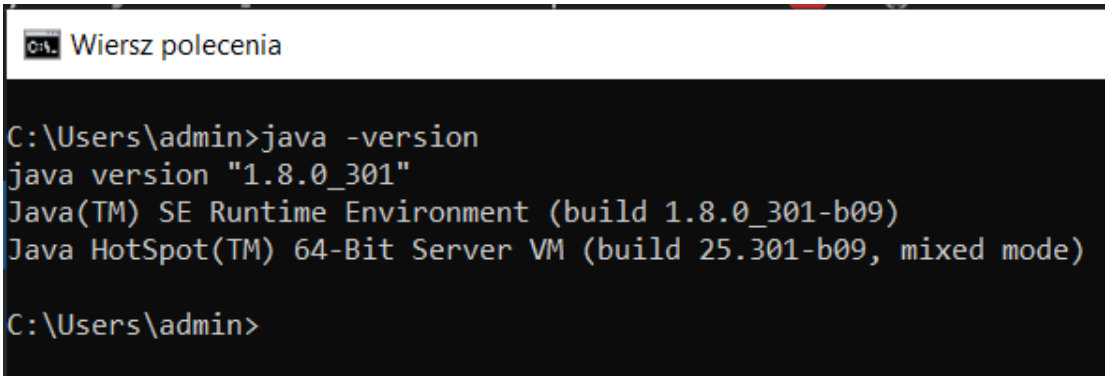
- Dodaj JAVA_HOME do Zmienne Systemowe




2. Dodaj ścieżkę folderu bin do **Path**

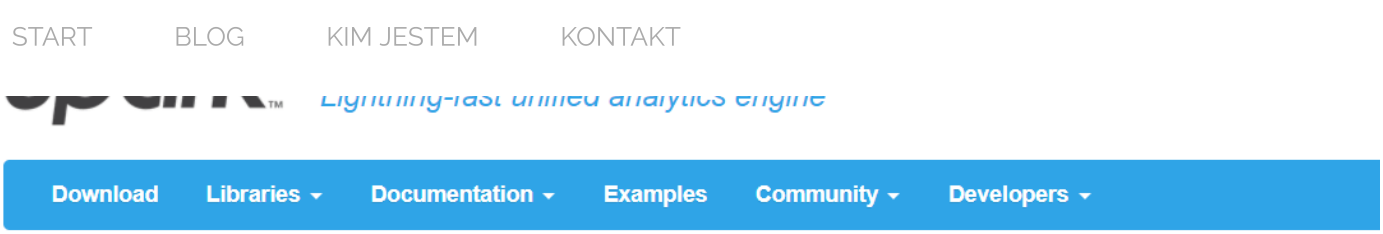



3. Sprawdź w konsoli czy Java jest zainstalowana. Konsole możesz uruchomić wpisując w wyszukiwarkę Windows '**cmd**' potem wpisz 'java -version'. Jeśli wszystko jest dobrze będzie to wyglądało tak:



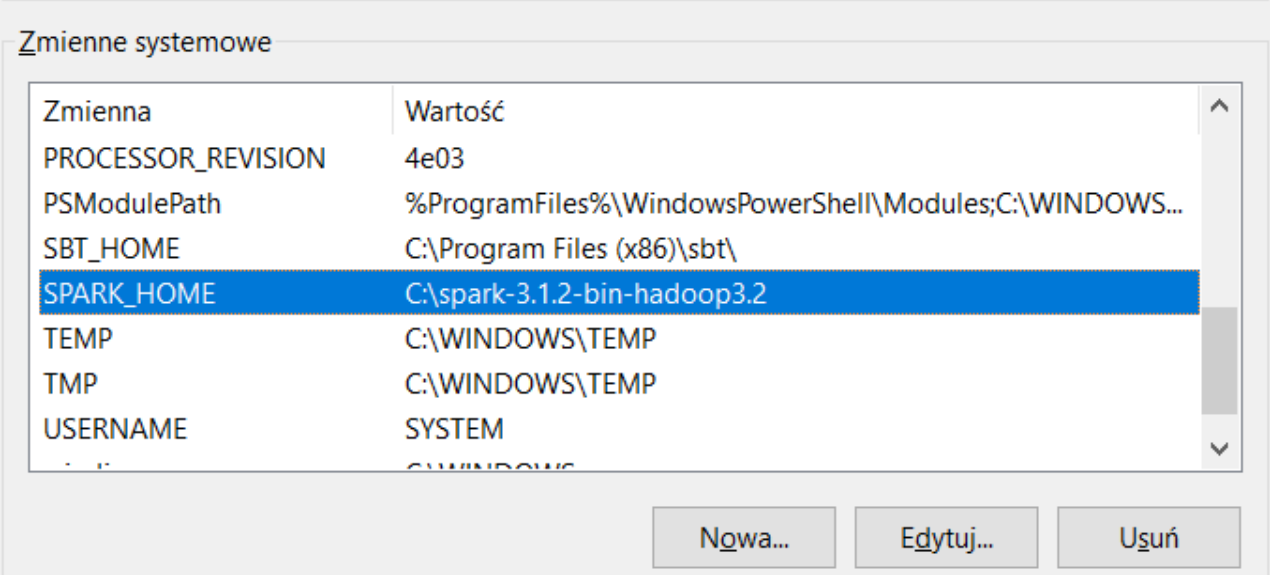

## Apache Spark

1. Pobierz najnowszego Sparka ze strony https://spark.apache.org/downloads.html

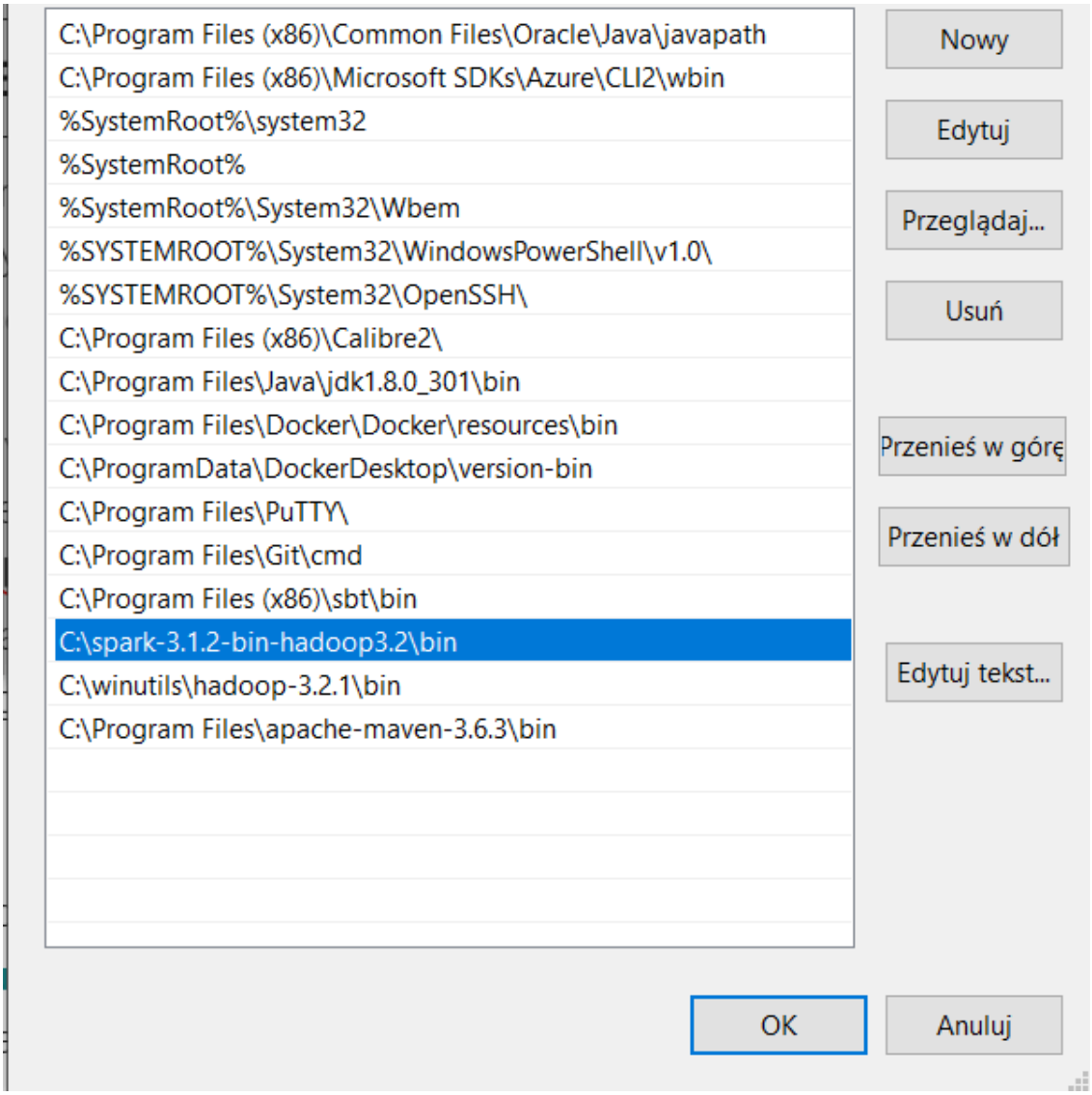



*Spark*

1. Rozpakuj pobrany plik, w moim przypadku jest to **spark-3.1.2-bin-hadoop3.2.tgz**
2. Umieść wszystkie foldery w ścieżce na dysku, ja trzymam to na C:\spark-3.1.2-bin-hadoop3.2, ale możesz to umieścić w innym miejscu.
3. Dodaj Zmienną środowiskową (‘Environment Variables’).
    - Dodaj SPARK_HOME do Zmienne Systemowe



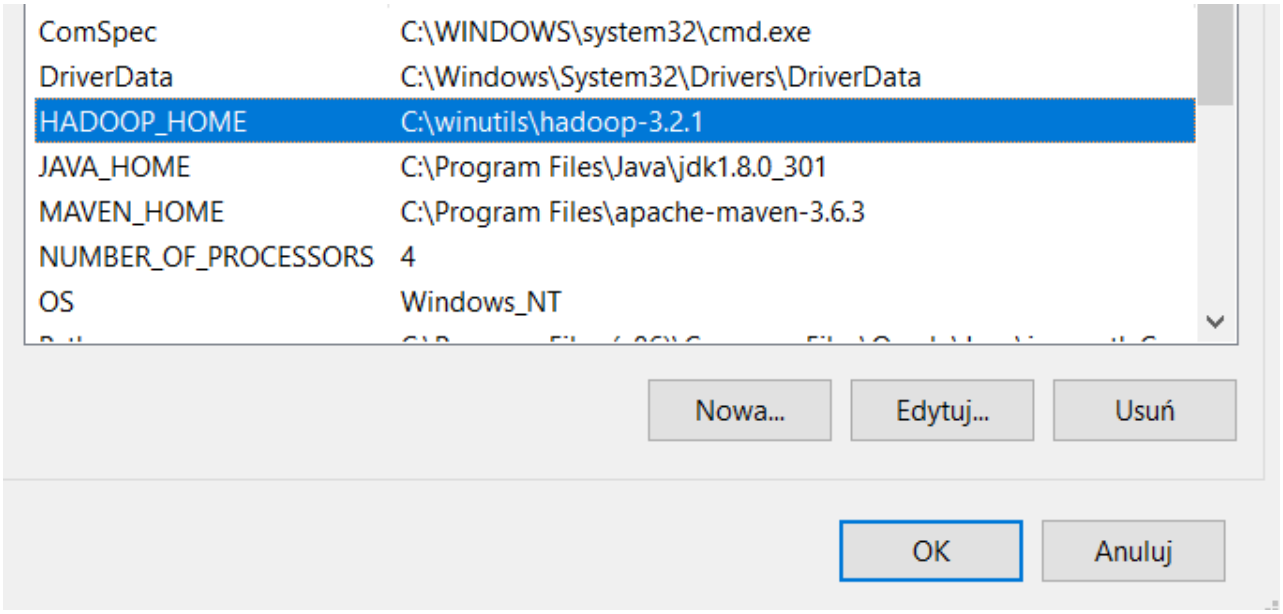

4. Dodaj ścieżkę folderu bin do **Path**



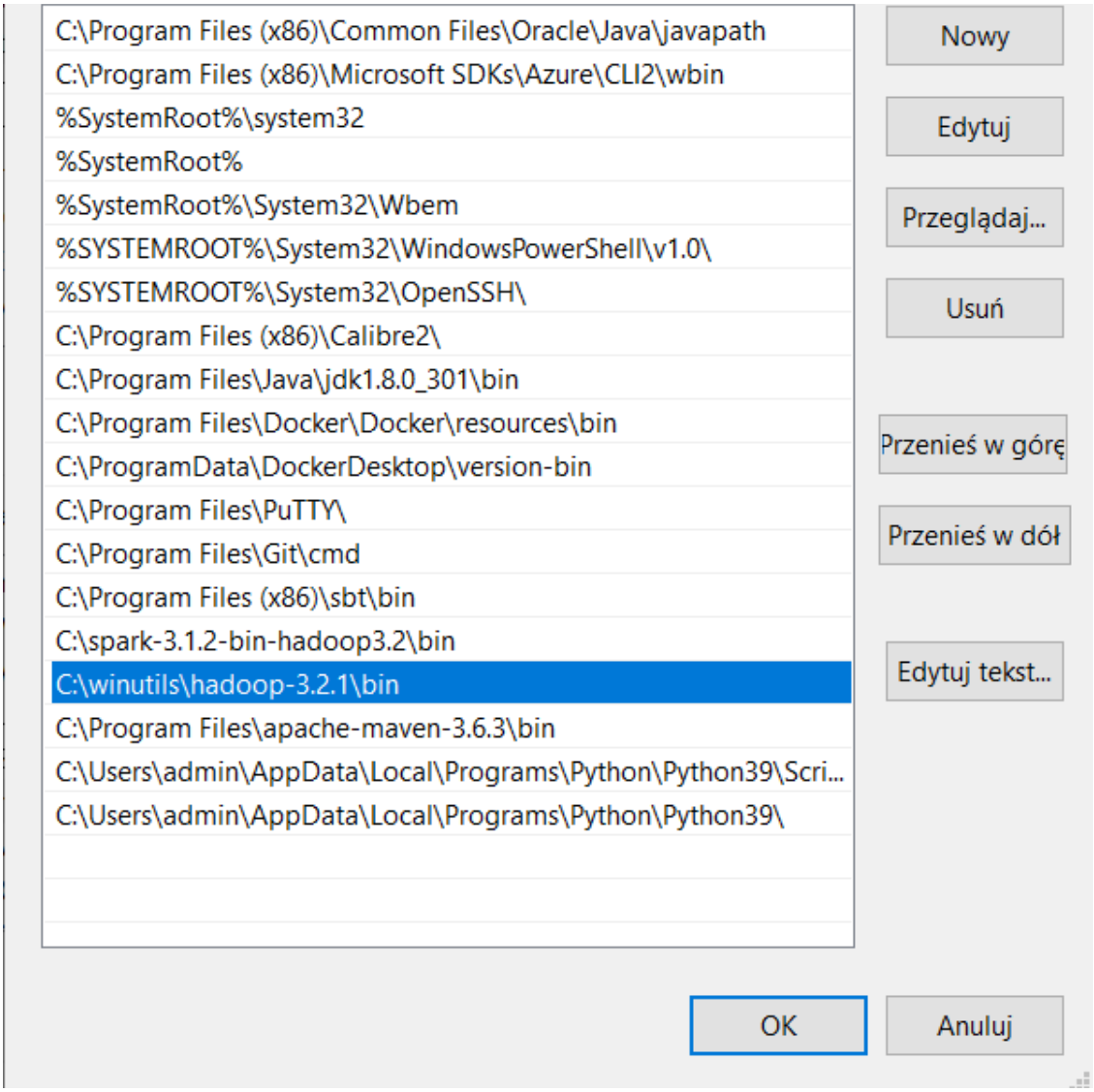

## Winutils

Winutils to plik binarny, który umożliwia działanie kodu hadoopa (HDFS) na windowsie. Możesz pobrać gotowy plik ja używam tego: winutils, lub się pobawić i wygenerować własny. Jest sporo instrukcji na necie ja trafiłem na tą stronę ale tego nie testowałem.

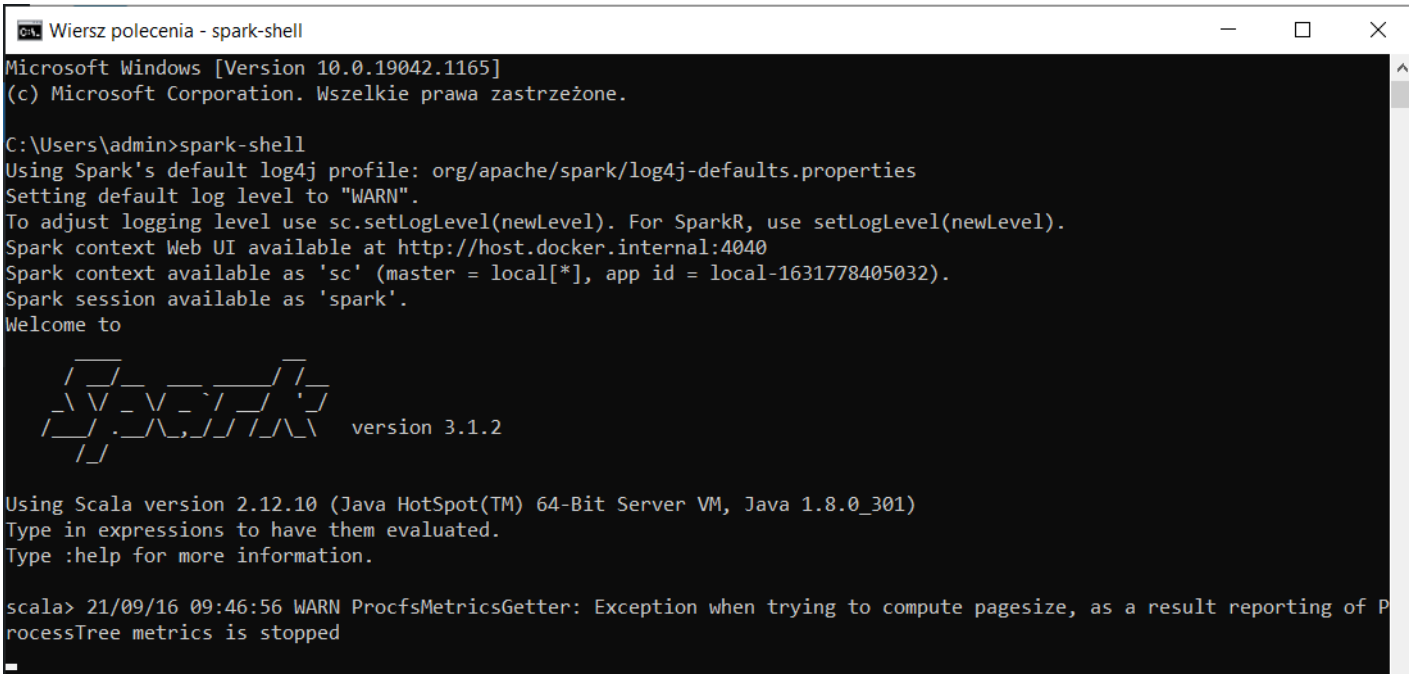
1. Pobierz pliki lub stwórz własnoręcznie w zależności od preferencji. Jak już zdobędziesz plik to umieść go na dysku, ja wrzucam na C:/
2. Dodaj Zmienną środowiskową (‘Environment Variables’).
    - Dodaj HADOOP_HOME do Zmienne Systemowe




- Dodaj ścieżkę folderu bin do **Path**




3. Sprawdź w konsoli czy Spark działa. Konsole uruchomisz jak w przypadku powyżej z Java. Wpisz ‘spark-shell’. Jeśli wszystko jest dobrze będzie to wyglądało tak:

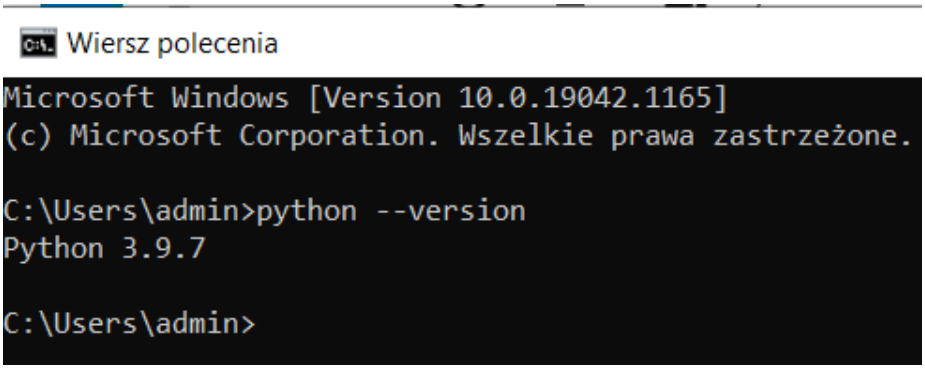


może się wydawać ekscentryczna. Ja bym polecił jakieś porządne narzędzie. Jeśli zainstalujesz Pythona, to w konsoli będziesz mógł uruchomić **Pyspark**.

# Python

Bez Pythona świat nie byłby taki sam, więc warto go poznać. Obecnie, Python jest bardzo popularnym językiem używanym w środowisku Spark, więc jeśli jesteś nim zainteresowany to polecam go wypróbować. Pobierz Pythona z oficjalnej strony.

1. Podczas instalacji wybierz opcje dodaj Python to PATH.
2. Sprawdź w konsoli wersje Phytona. Jeśli wysztko poszlo dobrze zobaczysz version w konsoli.




Teraz powinieneś uruchomić Pyspark żeby sprawdzić czy wszystko działa. Wróć do konsoli i wpisz **pyspark**. Zobaczysz środowisko w którym możesz używać Python w Sparku.

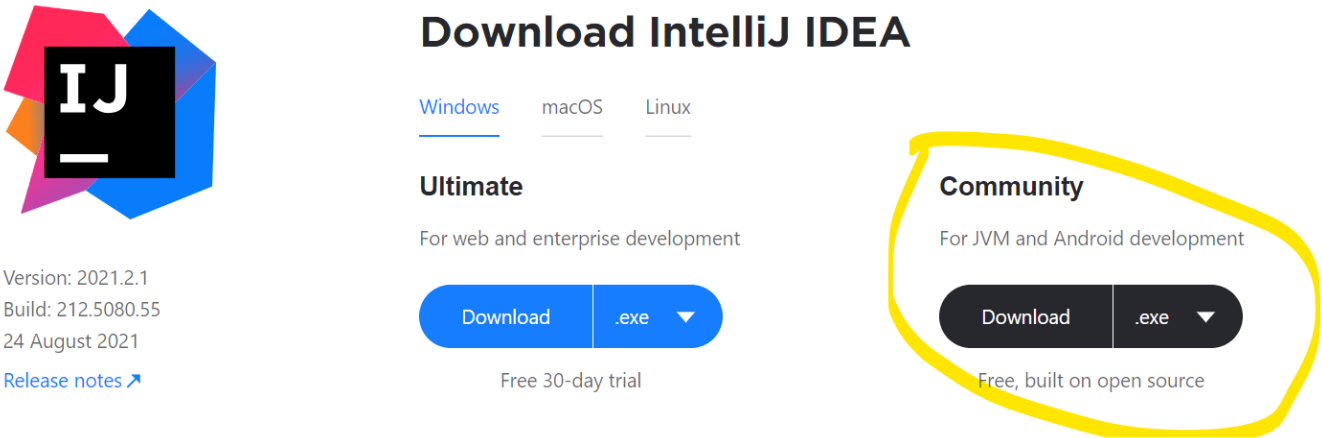
```
C:\Users\admin>pyspark
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Python version 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021 20:19:38)
Spark context Web UI available at http://host.docker.internal:4040
Spark context available as 'sc' (master = local[*], app id = local-1631779595416).
SparkSession available as 'spark'.
>>>
```

# Intellij

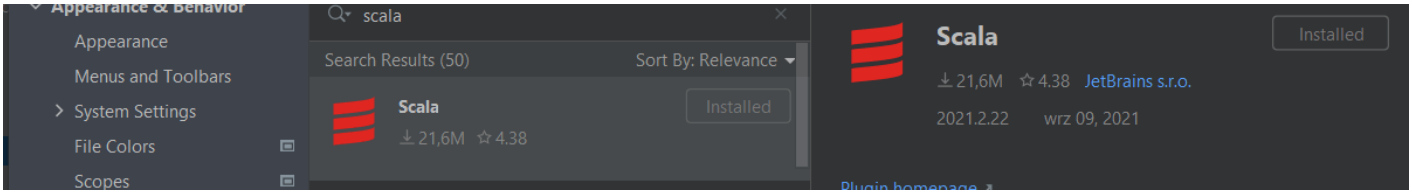
Jeśli chcesz działać w świecie scali to zapraszam do Intellij, jest to bardzo popularne narzędzie i działa jak należy.

Wystarczy pobrać Intellij Community i zainstalować. Tutaj nie ma specjalnych wymagań, wybierz wszystkie opcje i będzie dobrze.

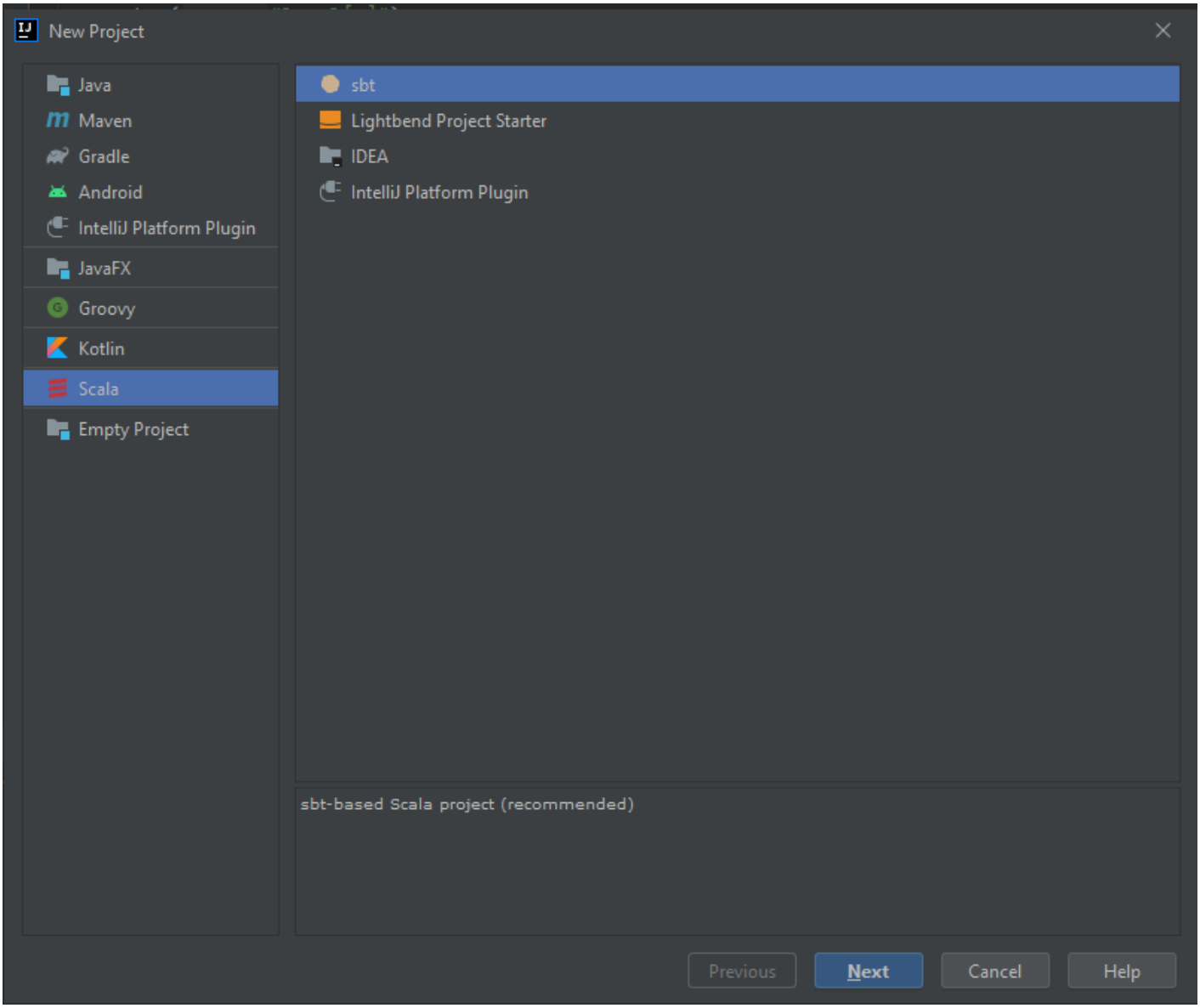



Po zainstalowaniu najważniejsze elementy, które trzeba doinstalować to plugin Scala. Żeby zainstalować plugin:

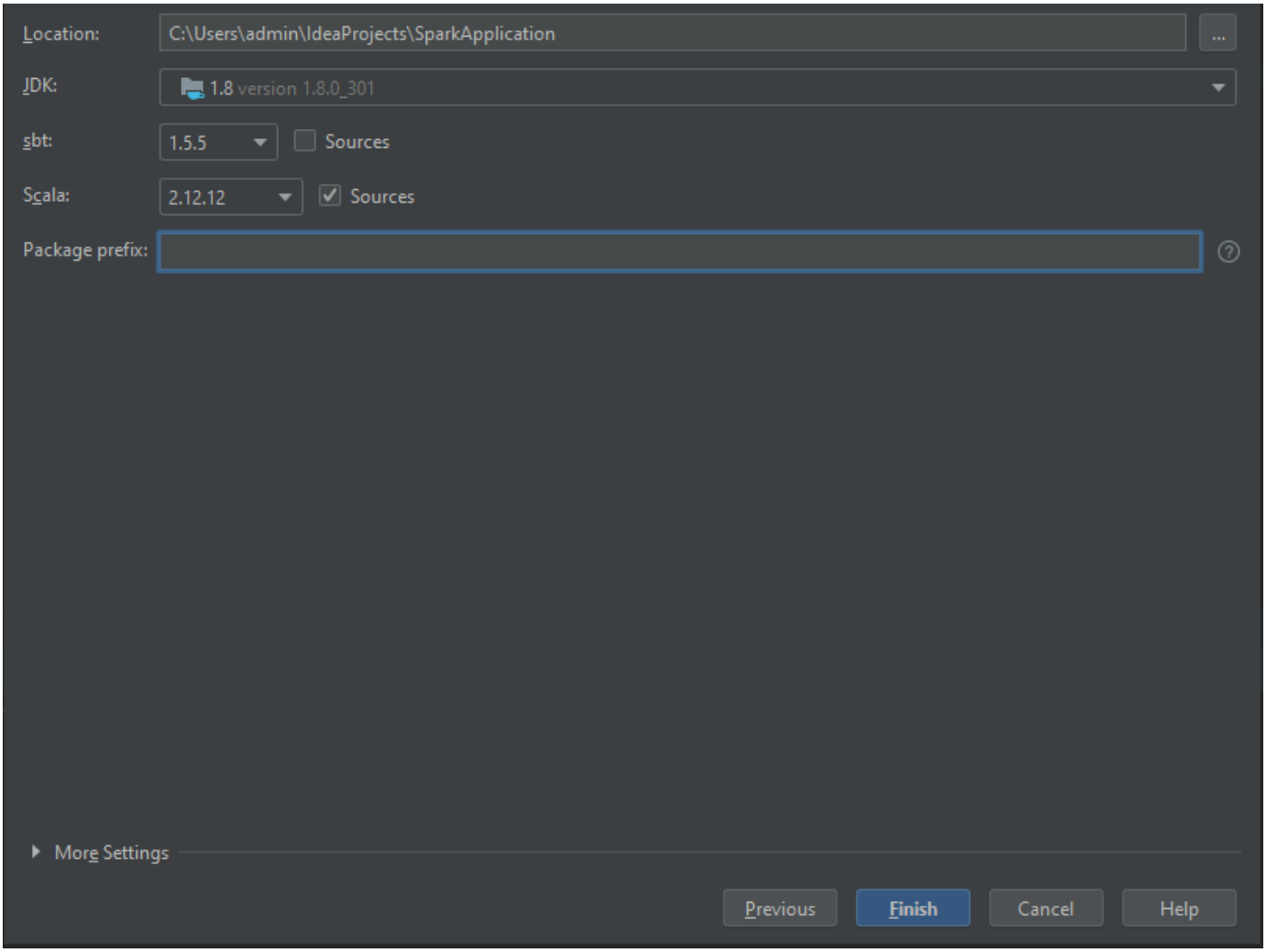
- File > Settings > Plugins




1. Stwórz nowy projekt. Z okna startowego Intellij lub z Menu File > New > Project. Wybierz Scala > sbt



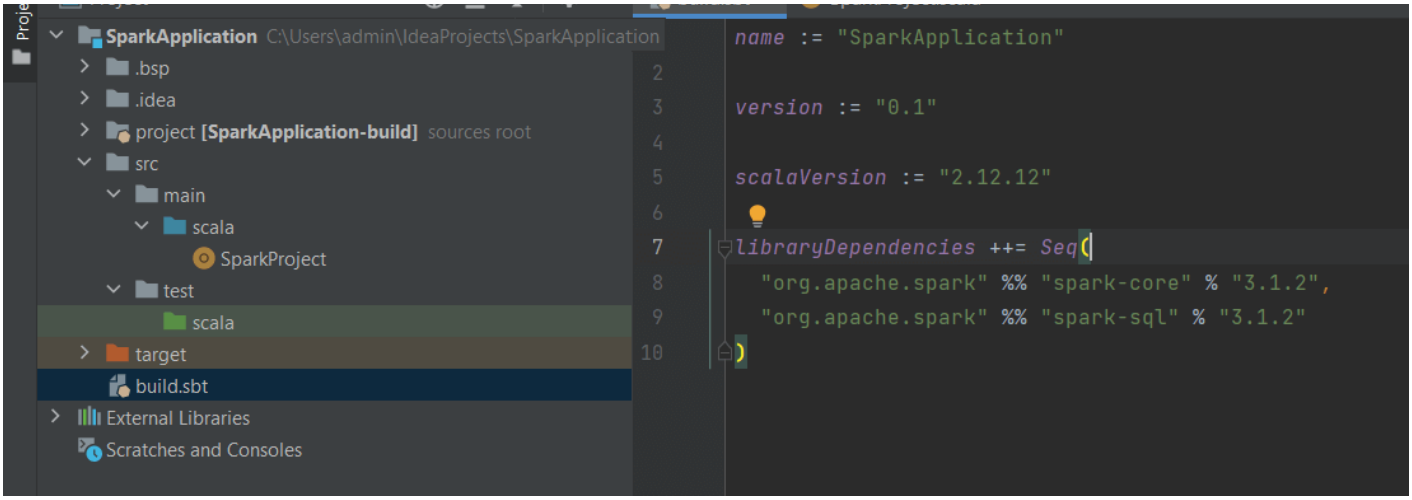

2. Podaj nazwę projektu “<SparkApplication>.”



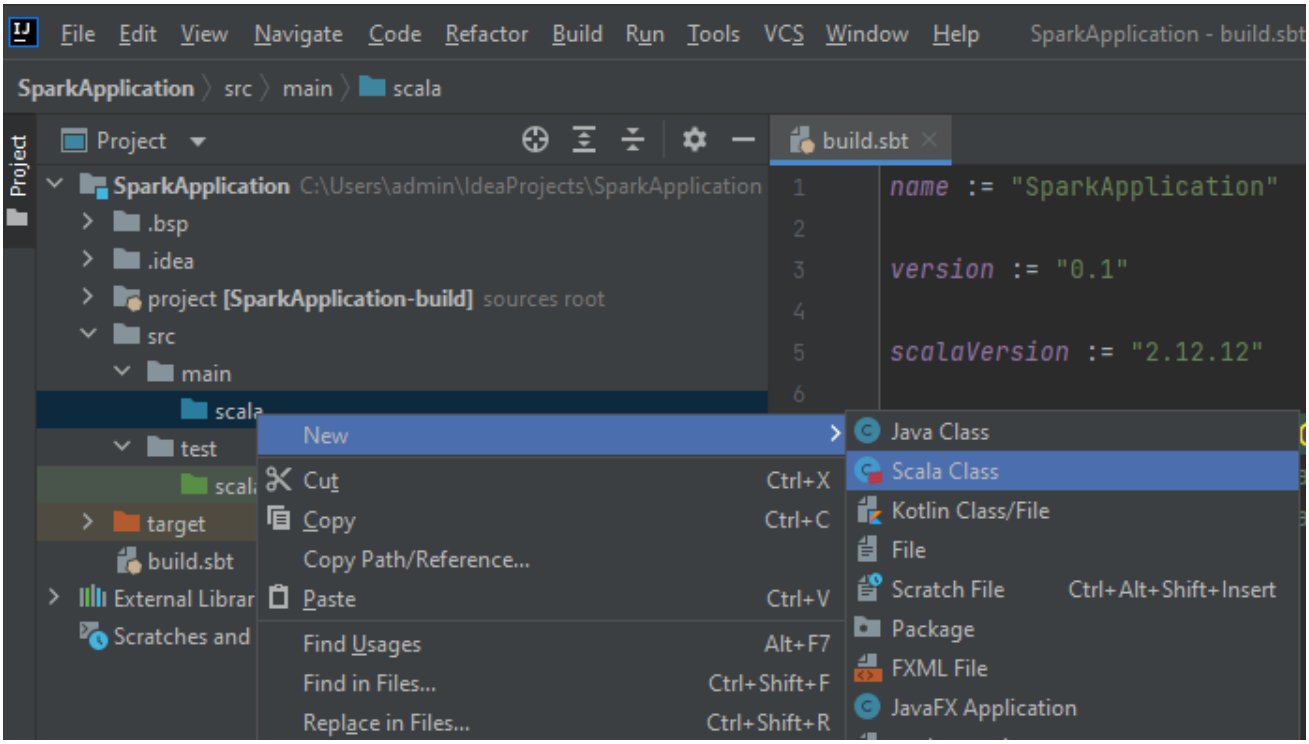

3. Kliknij Finish

Otwórz plik **build.sbt** i dodaj biblioteki Sparka. Pamiętaj, że bardzo ważne są wersje wszystkich bibliotek jakie dodasz. Najczęstsze błędy to niekompatybilne wersje języka i/lub wersji bibliotek.

```
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "3.1.2",
  "org.apache.spark" %% "spark-sql" % "3.1.2"
)
```



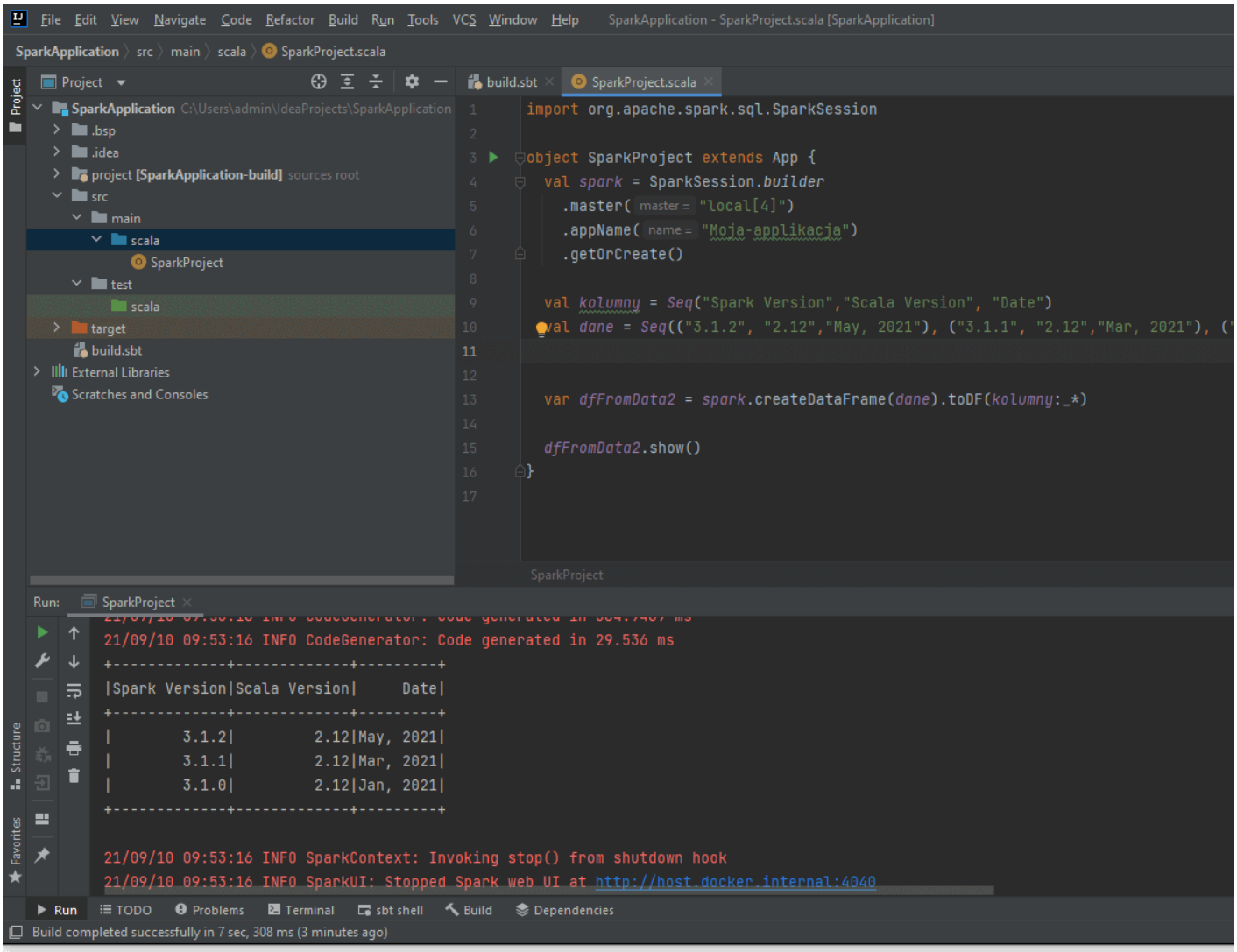

4. Dodaj Scala class “SparkProject” jako “Object ”




5. Dodaj kod wymagany do uruchomienia Sparka

```
1  import org.apache.spark.sql.SparkSession
2
3  object SparkProject extends App {
4    val spark = SparkSession.builder
5      .master("local[4]")
6      .appName("Moja-applikacja")
7      .getOrCreate()
8
9    val kolumny = Seq("Spark Version","Scala Version", "Date")
10   val dane = Seq(("3.1.2", "2.12","May, 2021"), ("3.1.1", "2.12","Mar, 2021"),
11
12
13   var dfFromData2 = spark.createDataFrame(dane).toDF(kolumny:_*)
```

6. Klikasz prawym gdziekolwiek w polu tekstowym i wybierasz “Run SparkProject”. Jeśli wszystko poszło dobrze to zobaczysz DataFrame. Teraz możesz zacząć eksperymentować ze Sparkiem. Milej zabawy.



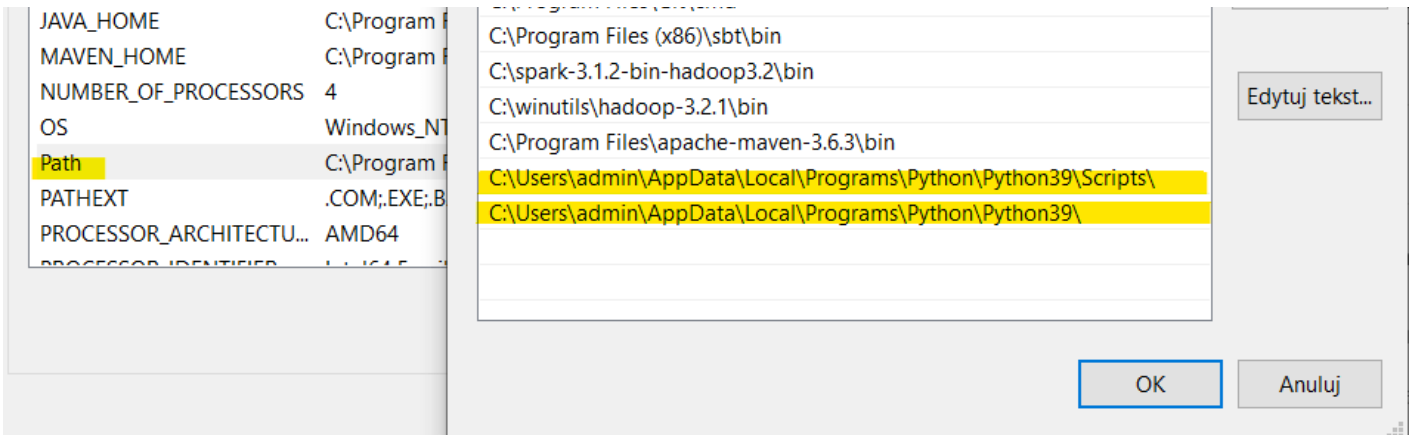

## PyCharm

Jeśli wybrałeś Python to dobrym narzędziem będzie **PyCharm**. Zainstaluj Community Edition. Ja zawsze wybieram wszystkie opcje jak coś instaluję żeby się nie ograniczać 🙂

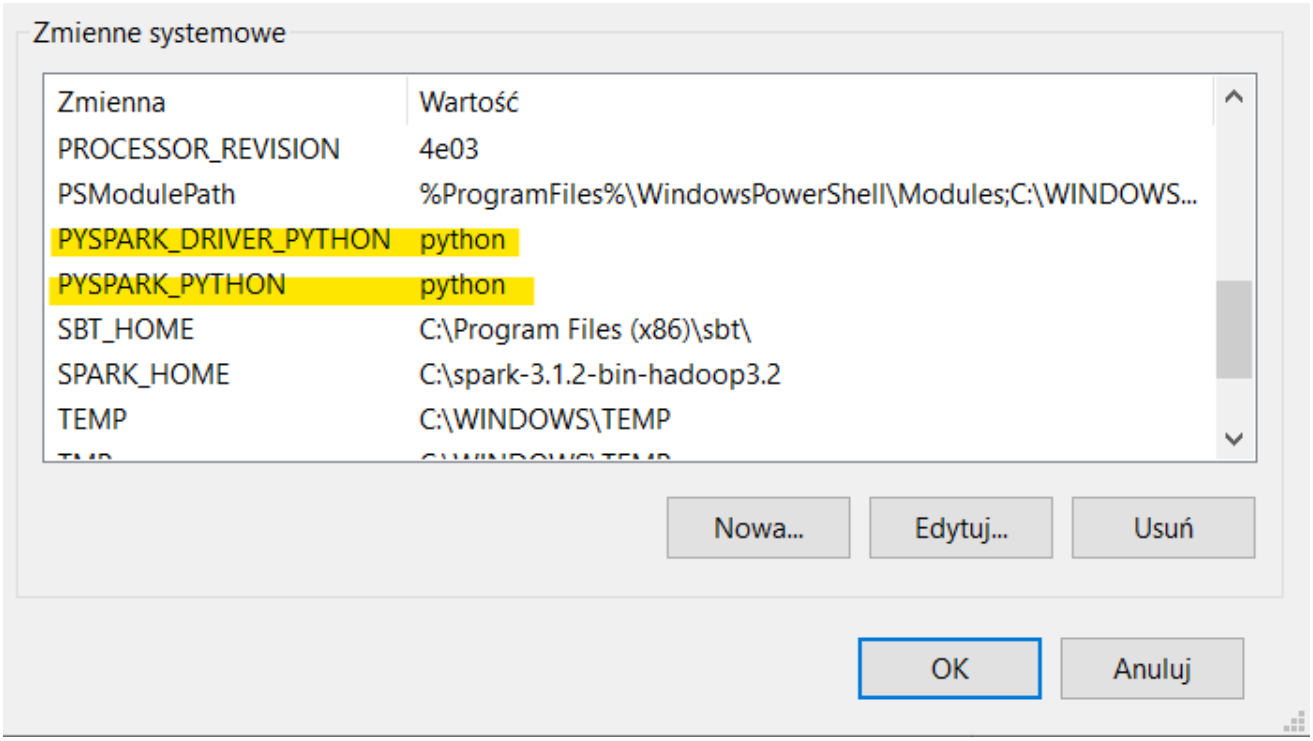
### Konfiguracja dla PyCharm

#### Zmienne środowiskowe

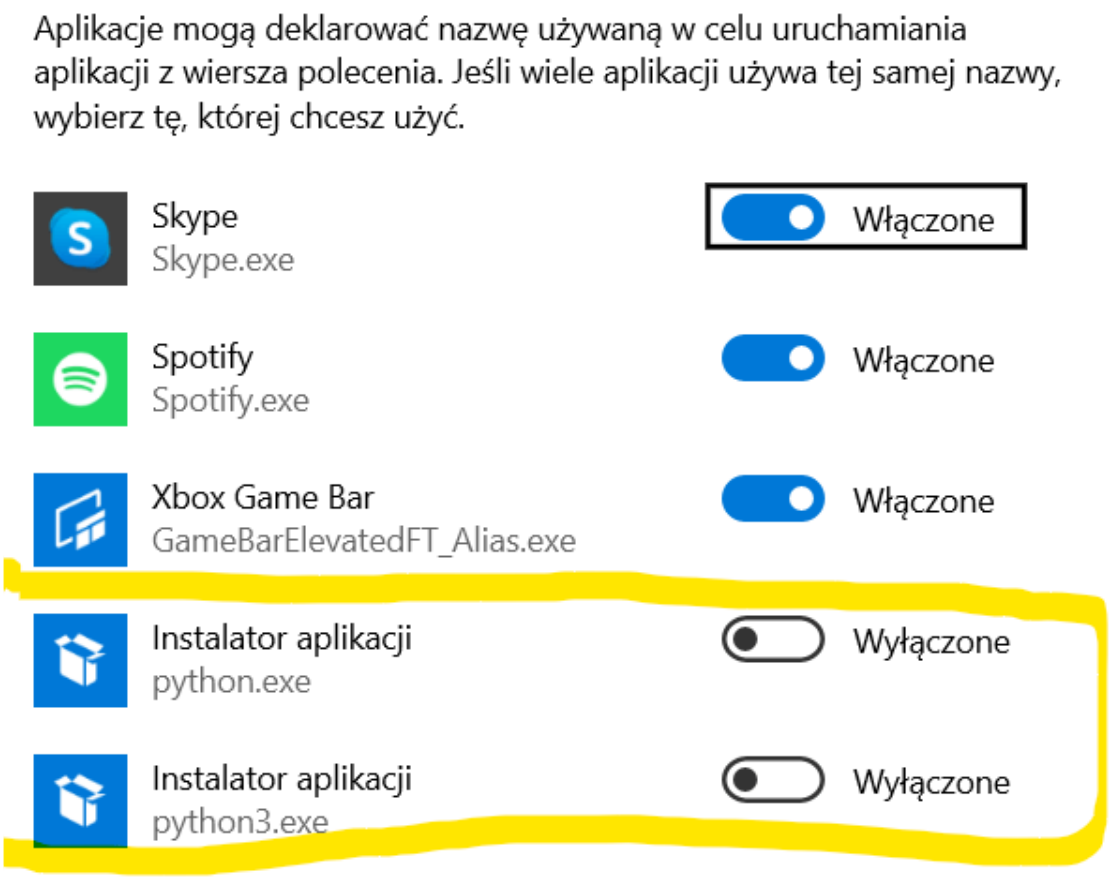
1. Upewnij się ze ścieżki Python są dodane do systemowych zmiennych środowiskowych



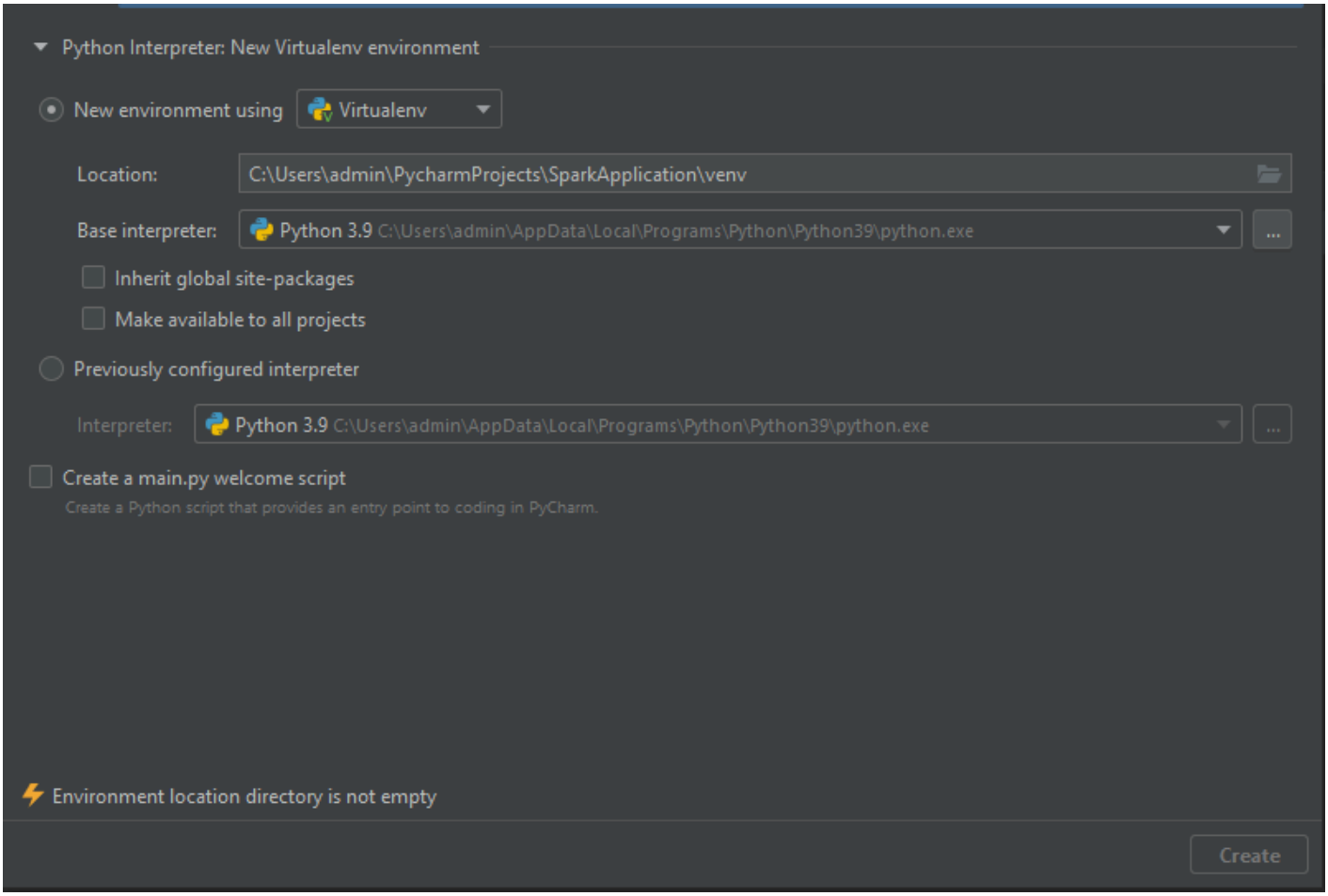

2. Dodaj wartości dla zmiennych PYSPARK_DRIVER_PYTHON i PYSPARK_PYTHON



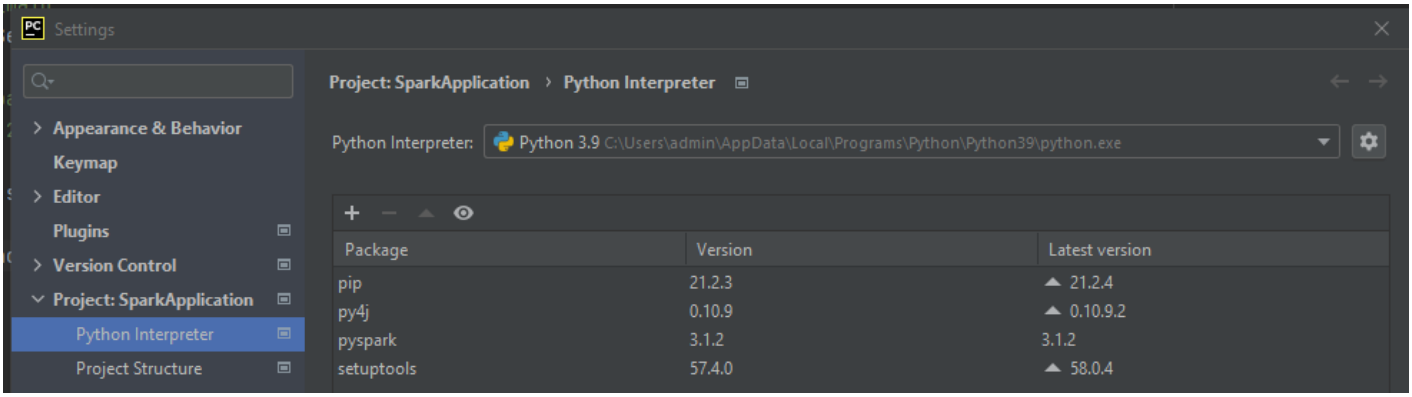

3. Wyłącz “Manage App Execution Aliases” (Alliasy wykonania aplikacji) dla Pythona



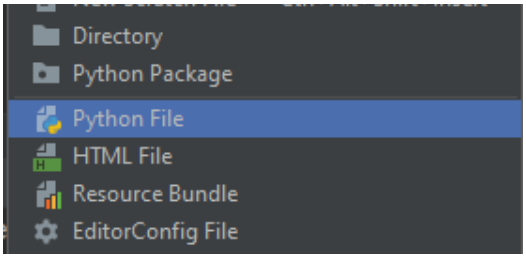

1. Stwórz nowy projekt, tutaj powinna wystarczyć tylko nazwa.




2. Po odpaleniu PyCharm trzeba doinstalować dwie paczki py4j i pyspark. Moża to zrobić z File > Settings > Project: <Project Name> – Kliknij znak plus nad tabelką. Wpisujesz py4j a potem pyspark (masz opcję wyboru wersji, upewnij się że pasuje do wersji Spark, którą pobrałeś)




3. Dodaj plik File > New > Python File




4. Dodaj kod

```
from pyspark.sql import SparkSession

if __name__ == '__main__':

    spark = SparkSession.builder.master("local[*]").appName("HelloWorld").getOr(

     kolumny = ["Spark Version", "Scala Version", "Date"]

     dane = [("3.1.2", "2.12", "May, 2021"), ("3.1.1", "2.12", "Mar, 2021"), ("

     dfFromData2 = spark.createDataFrame(dane).toDF(*kolumny)

     dfFromData2.show()
```

Klikasz prawym na Run i jeśli wszystko jest ok to powinieneś widzieć DataFrame.