

Tak łatwy że aż...



Kamil Cecherz
Łukasz Bednarski

MONTY PYTHON'S
FLYING
CIRCUS

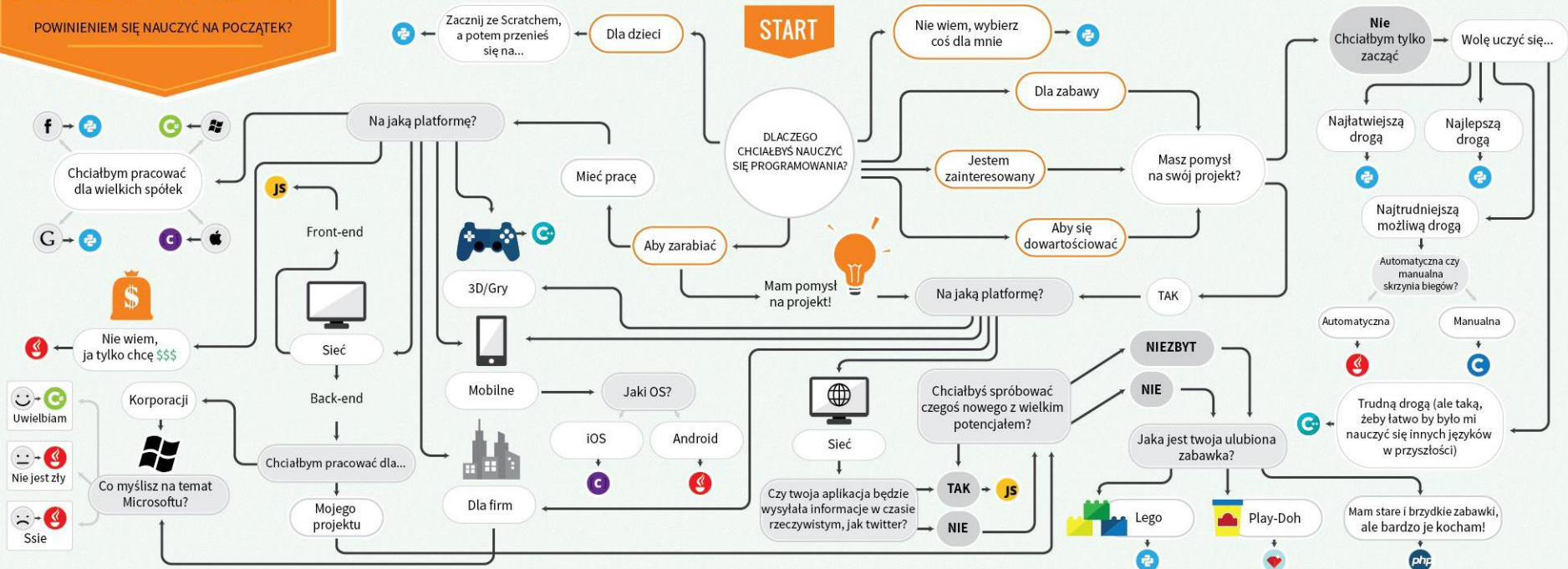
Agenda

1. Historia, statystyki, ciekawostki.
2. Środowisko pracy, specyfika technologii.
 - Wady i zalety technologii.
 - Dynamiczne typy danych a zastosowanie biblioteki MyPy.
 - Praca na jednym corze (programowanie współbieżne)
 - Magic methods, parametry immutable
 - Podstawowe narzędzia i techniki pracy.
 - Virtual Env
 - Konfiguracja zależności: Poetry
 - Testy jednostkowe
3. Pytania o zastosowania Pythona:
 - Aplikacje webowe
 - Django
 - Flask
 - Data Science i AI
 - Pandas
 - Tensor Flow
 - Gry komputerowe
 - PyGame
4. Charakterystyka budowy języka:
 - Python 2.x a Python 3.x

Uwaga Python !

- **Popularność Pythona** rośnie z każdym rokiem do tego stopnia, że jest niemal najpopularniejszym językiem. Na pewno przyczyniło się do tego wzrost zainteresowania tematami jak **machine learning i data science** w ostatnich latach.
- Według **TIOBE index**, język **Python** jest na 3 pozycji pod względem popularności. W ubiegłym roku był na pozycji 4.
Z kolei według rankingu Spectrum jest na 5 pozycji, a rok temu był na miejscu 1.
- Na **Stack Overflow** jest 5 najczęściej tagowanym językiem z 1,280,907 zapytaniem.
- Na **Github** zajmuje 2 pozycję z liczbą 864,307 projektów, z których 2,459 ma więcej niż 1000 gwiazdek.
- Jeśli chodzi o organizowane spotkania to według Meetup.com jest to 2 co do wielkości społeczność z liczbą 2,490 grup, w skład których wchodzi 2,275,430 członków na całym świecie. W Polsce mamy odpowiednio 45 grup, w skład których wchodzi 36,901 członków.

POWINNIENIEM SIĘ NAUCZYĆ NA POCZĄTEK?



PORÓWNANIE POSTACI Z WŁADZY PIERŚCIENI DO JĘZYKÓW PROGRAMOWANIA

Python The Ent Pomaga małym Hobbitom (początkującym) w zrozumieniu ich programistycznych konceptów. Pomaga czarodziejom (informatykom) w przeprowadzaniu badań Szeroko uważany za najlepszy (i najłatwiejszy) język dla początkujących Szeroko używany w naukowym, technicznym polu jak np. sztuczna inteligencja Możesz budować stronę www używając Django - popularny Pythonowy framework POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA YouTube, Instagram, Spotify SR. ROCZNA PŁACA W USA \$107,000	Java Gandalf Pragnie pokoju oraz pracuje z każdym (przełożony) Bardzo popularny na wszystkich platformach, systemy oraz urządzenia ze względu na multiplatformowość Duże zapotrzebowanie na rynku oraz najlepiej opłacany język programowania Slogan: „Napisać, pracuj wszędzie” POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Gmail, Minecraft, Most Android Apps, Enterprise applications SR. ROCZNA PŁACA W USA \$102,000	C One Ring Moc C jest znana wszystkim Każdy chce jego mocy Mieszanka wszystkich języków programowania Jeden z najstarszych i najszerzej używanych języków programowania na świecie Popularny do programowania hardware Podzbiór C++ z wyjątkiem małych detali POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Operating systems and hardware SR. ROCZNA PŁACA W USA \$102,000	C++ Saruman Wszyscy myślą, że jest dobry Jednak gdy już go poznasz, zrozumiesz że chce mocy, nie dobrych uczynków Kompleksowa wersja C z wieloma nowymi funkcjami Szeroko pojęty do produkcji gier, przemysłu oraz bardzo wydajnych aplikacji Uczenie się C++ jest jak nauka jak wyprodukować, zbudować i prowadzić samochód Polecany tylko wtedy, kiedy masz osobę która będzie Cię prowadzić POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Operating systems, hardware, and browsers SR. ROCZNA PŁACA W USA \$104,000	JavaScript Hobbit Często lekceważony (potężny) Bardzo znany z powolnego, delikatnego życia Shire (przeglądarka) „Java i JavaScript są tak samo podobne jak Samochód i dywan” - Greg Gervill Najbardziej popularny backendowy język webowy Konieczność dla front-endowego developera (Jak również HTML & CSS) Jeden z aktualnie najciekawszych języków (ze względu na node.js) POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Paypal, front-end of majority websites SR. ROCZNA PŁACA W USA \$99,000	C# Elf Piękna Istota (język), zwykle zostaje na swojej ziemi, Rivendell (Platforma Microsoft), jednak ostatnimi czasami zaczyna otwierać się na swoich sąsiadów (open source) Popularny wybór dla korporacji do tworzenia stron internetowych oraz aplikacji Windowsowych z użyciem .NET framework. Może zostać użyty aby zbudować stronę z ASP.NET - internetowym frameworku od Microsoft Podobny do Javy w podstawowej składni i części funkcji POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Enterprise oraz aplikacji Windowsowych SR. ROCZNA PŁACA W USA \$94,000	Ruby Man Bardzo uczuciowa postać Oni (developerzy Ruby) myślą, że są najlepszą chł chł rządzić Śródziemiem Głównie znany jako najpopularniejszy webowy framework, Ruby on Rails Skupia się na skończeniu projektu Zaprojektowany do zabawy i wydajnego kodowania Najlepszy dla zabawy i osobistych projektów oraz szybkiego rozwoju POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Hulu, Groupama i Slideshare SR. ROCZNA PŁACA W USA \$107,000	PHP Orc Brydki facet (język). Nie ma szacunku do zasad (jest niespójny i nieprzewidywalny) Powoduje wielki ból głowy dla osób (developerów) które chcą nim rządzić (kodem) Dominuje Śródziemie (najpopularniejszy język web developerki) Odpowiedni do tworzenia małych i prostych stron w krótkim czasie Wspierają go prawie każdy tani hosting POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Wordpressa, Wikipedi oraz Flickr SR. ROCZNA PŁACA W USA \$89,000	Objective-C Smaug Samotny i kocha złoto Główny język używany przez Apple dla Mac OS i IOS Wybierz czy chcesz się skupić na tworzeniu aplikacji dla IOS lub OS X Rozważ naukę Swift (wypuszczony przez Apple w 2014) jako twojego następnego języka POPULARNOŚĆ ★★★★★ UŻYTY DO ZROBIENIA Wielkośći aplikacji na IOS i części Mac OS X SR. ROCZNA PŁACA W USA \$107,000
---	---	--	--	--	---	---	---	---

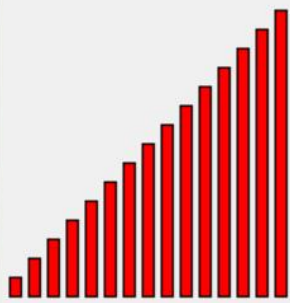
	Nov 2019	Nov 2018	Change	Programming Language	Ratings	Change
1		1		Java	16.246%	-0.50%
2		2		C	16.037%	+1.64%
3		4	⬆	Python	9.842%	+2.16%
4		3	⬇	C++	5.605%	-2.68%
5		6	⬆	C#	4.316%	+0.36%
6		5	⬇	Visual Basic .NET	4.229%	-2.26%
7		7		JavaScript	1.929%	-0.73%
8		8		PHP	1.720%	-0.66%
9		9		SQL	1.690%	-0.15%
10		12	⬆	Swift	1.653%	+0.20%
11		16	⬆	Ruby	1.261%	+0.17%
12		11	⬇	Objective-C	1.195%	-0.28%
13		13		Delphi/Object Pascal	1.142%	-0.28%
14		25	⬆	Groovy	1.099%	+0.50%
15		15		Assembly language	1.022%	-0.09%
16		14	⬇	R	0.980%	-0.43%
17		20	⬆	Visual Basic	0.957%	+0.10%
18		23	⬆	D	0.927%	+0.25%
19		17	⬇	MATLAB	0.890%	-0.14%
20		10	⬇	Go	0.853%	-0.64%

Zastosowanie

- **Python** jest językiem o bardzo szerokim zastosowaniu, pozwalającym na napisanie dosłownie wszystkiego przy użyciu odpowiednich bibliotek lub frameworków, przez co bardzo wiele firm używa go w swoich aplikacjach.
- Jego interpretery są dostępne dla wielu systemów, co czyni go językiem wieloplatformowym.
- Króluje w dziedzinach jak **data science** czy **machine learning**. **Python** może być także używany w **Raspberry Pi**
- Używany jako **język skryptowy** w aplikacjach napisanych w innych językach. Np. w takich programach jak **Gimp**, **Inkscape** czy **Blender**.
- W **Pythonie** można pisać też **aplikacje desktopowe**, ale raczej odchodzi się od tego zastosowania. Popularne z nich to np. pierwotny klient **Dropbox** czy starsze wersje **BitTorrent**.
- Dla **Pythona** nigdy priorytetem nie była wydajność. **Python** skupia inne mocniejsze strony m.in. czytelność, łatwość użycia i produktywność.

Programy demo

Sort demo, size 15



0 cmps, 0 swaps

Quicksort	Randomize
Insertion sort	Uniform
Selection sort	Distinct
Bubble sort	Demo
15	Cancel
normal	Quit
Step	

Spreadsheet: sheet1.xml

A1 fgf

	A	B	C	D	E
1	fgf				
2	dfdf	NameError			
3					
4					
5					
6					
7					
8					
9					
10					

Windows PowerShell

```
n\Python38-32\Tools\demo
PS C:\Users\kacec1\AppData\Local\Programs\Python\Python38-32\Tools\demo> dir

Directory: C:\Users\kacec1\AppData\Local\Programs\Python\Python38-32\Tools\demo

Mode                LastWriteTime         Length Name
----                -
d-----         26.11.2019        12:50          __pycache__
-a----         14.10.2019         20:42           591 beer.py
-a----         14.10.2019         20:42          4057 eiffel.py
-a----         14.10.2019         20:42          4755 hanoi.py
-a----         14.10.2019         20:42          9249 life.py
-a----         14.10.2019         20:42          3816 markov.py
-a----         14.10.2019         20:42          2305 mcast.py
-a----         14.10.2019         20:42          2355 queens.py
-a----         14.10.2019         20:42          5920 redemo.py
-a----         14.10.2019         20:42           848 rpython.py
-a----         14.10.2019         20:42          1378 rpythond.py
-a----         14.10.2019         20:42         20616 sortvisu.py
-a----         14.10.2019         20:42         26452 ss1.py
-a----         14.10.2019         20:42          1526 vector.py
```

PS C:\Users\kacec1\AppData\Local\Programs\Python\Python38-32\Tools\demo> python sortvisu.py



Python vs Java

Dimensions

Python

Java

Verbosity

Concise

Verbose

Performance

Interpreted, slower

Faster

Learning Curve

Easier than Java

Easy

Typing discipline

Dynamically-typed (duck-typing)

Statically-typed

Best for

Data Science, AI, Machine Learning

Embedded and cross-platform applications

Python 3: Fibonacci series up to n

```
1  def fib(n):  
2      a, b = 0, 1  
3      while a < n:  
4          print(a, end=' ')  
5          a, b = b, a+b  
6      print()  
7  fib(1000)
```

Własny prosty program

```
1  # memorem 0.1 by Kamil Cecherz #
2  print("*** Witaj w programie memorem ***")
3  operation = None
4
5  while(operation != "exit"):
6
7      print("exit - wyjście z programu")
8      print("open - przegląd bazy słówek")
9      print("write - edycja bazy słówek")
10
11     operation = input("Wydaj polecenie: ")
12     if(operation == "exit"):
13         print("\nZakończenie programu:")
14     elif(operation == "open"):
15         print("Otwarto bazę słówek")
16         basefile = open("baza.txt", "r")
17
18         i = 0
19         for line in basefile:
20             print(str(i) + ". " + line)
21             i += 1
22
23         basefile.close()
24
25     elif(operation == "write"):
26         print("Zapis słówek do bazy:")
27         basefile_user = open("user_base.txt", "w")
28         basefile_user.write((input("Wpisz nową notatkę i zatwierdź (ENTER):\n Poprzedni plik zostanie nadpisany.)))
29         basefile_user = open("user_base.txt", "r")
30         i = 0
31         for line in basefile_user:
32             print(str(i) + ". " + line)
33             i += 1
34
35         print("Dadano nową notatkę !")
36         basefile_user.close()
```

Dynamiczne typy danych a MyPy

- **MyPy** to moduł sprawdzania typów statycznych dla Pythona, który ma na celu połączenie zalet dynamicznego i statycznego typowania.
- **MyPy** łączy ekspresyjną moc i wygodę Pythona z potężnym systemem typów i sprawdzaniem typu podczas kompilacji.
- Typ **MyPy** sprawdza standardowe programy w języku Python; uruchamia się przy użyciu dowolnej maszyny wirtualnej Python bez generowania narzutu w czasie wykonywania.

Dynamiczne typy danych a MyPy

Simple class

Mypy with dynamic typing

```
class BankAccount:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance
    def deposit(self, amount):
        self.balance += amount
    def withdraw(self, amount):
        self.balance -= amount
    def overdrawn(self):
        return self.balance < 0
```

```
my_account = BankAccount(15)
my_account.withdraw(5)
print(my_account.balance)
```

Mypy with static typing

```
class BankAccount:
    def __init__(self, initial_balance: int = 0) -> None:
        self.balance = initial_balance
    def deposit(self, amount: int) -> None:
        self.balance += amount
    def withdraw(self, amount: int) -> None:
        self.balance -= amount
    def overdrawn(self) -> bool:
        return self.balance < 0
```

```
my_account = BankAccount(15)
my_account.withdraw(5)
print(my_account.balance)
```

Praca z wątkami



Wydajność

Java

```
import java.math.BigDecimal;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        System.out.println("Podaj potęgę liczby 2");
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        BigDecimal b1 = new BigDecimal("2");
        BigDecimal result = b1.pow(n);
        System.out.println(result);
    }
}
```

2 do potęgi 9999999 = 2 sek. <

2 do potęgi 99999999 = 30 sek <

Wydajność

Python

```
def my_function():  
    x = int(input("Podnieś do potęgi liczbę 2\n"))  
    y = 2**x  
    print(y)  
|  
my_function()
```

2 do potęgi 9999999 = 10 sek. <

2 do potęgi 99999999 = ?

Magic Method	When it gets invoked (example)	Explanation
<code>__new__(cls [,...])</code>	<code>instance = MyClass(arg1, arg2)</code>	<code>__new__</code> is called on instance creation
<code>__init__(self [,...])</code>	<code>instance = MyClass(arg1, arg2)</code>	<code>__init__</code> is called on instance creation
<code>__cmp__(self, other)</code>	<code>self == other</code> , <code>self > other</code> , etc.	Called for any comparison
<code>__pos__(self)</code>	<code>+self</code>	Unary plus sign
<code>__neg__(self)</code>	<code>-self</code>	Unary minus sign
<code>__invert__(self)</code>	<code>~self</code>	Bitwise inversion
<code>__index__(self)</code>	<code>x[self]</code>	Conversion when object is used as index
<code>__nonzero__(self)</code>	<code>bool(self)</code>	Boolean value of the object
<code>__getattr__(self, name)</code>	<code>self.name</code> # name doesn't exist	Accessing nonexistent attribute
<code>__setattr__(self, name, val)</code>	<code>self.name = val</code>	Assigning to an attribute
<code>__delattr__(self, name)</code>	<code>del self.name</code>	Deleting an attribute
<code>__getattribute__(self, name)</code>	<code>self.name</code>	Accessing any attribute
<code>__getitem__(self, key)</code>	<code>self[key]</code>	Accessing an item using an index
<code>__setitem__(self, key, val)</code>	<code>self[key] = val</code>	Assigning to an item using an index
<code>__delitem__(self, key)</code>	<code>del self[key]</code>	Deleting an item using an index
<code>__iter__(self)</code>	<code>for x in self</code>	Iteration
<code>__contains__(self, value)</code>	<code>value in self</code> , <code>value not in self</code>	Membership tests using <code>in</code>
<code>__call__(self [,...])</code>	<code>self(args)</code>	"Calling" an instance
<code>__enter__(self)</code>	<code>with self as x:</code>	with statement context managers
<code>__exit__(self, exc, val, trace)</code>	<code>with self as x:</code>	with statement context managers
<code>__getstate__(self)</code>	<code>pickle.dump(pk_l_file, self)</code>	Pickling
<code>__setstate__(self)</code>	<code>data = pickle.load(pk_l_file)</code>	Pickling

Magiczne metody

Narzędzia

- **Virtualenv** to narzędzie służące do tworzenia izolowanego środowiska dla Pythona. To środowisko ma własne katalogi instalacyjne, które nie współużytkują bibliotek z innymi środowiskami **virtualenv** (i opcjonalnie nie mają dostępu do bibliotek zainstalowanych globalnie).
- **Poetry** to narzędzie do zarządzania zależnościami i pakowania w Pythonie. Pozwala zadeklarować biblioteki, od których zależy Twój projekt, i zarządza nimi (instaluje / aktualizuje).
- **pip** to instalator pakietów dla Pythona. Możesz użyć pip do zainstalowania pakietów z indeksu pakietów Python i innych indeksów.

```
pipx install poetry
```

```
pipx upgrade poetry
```

```
pipx uninstall poetry
```

Instalacja Virtualenv z użyciem pip

```
[server]$ python3 -m pip install --upgrade pip
```

```
[server]$ pip3 install virtualenv
```

```
Collecting virtualenv
```

```
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
```

```
    100% |████████████████████████████████████████████████████████████████████████████████| 1.8MB 367kB/s
```

```
Installing collected packages: virtualenv
```

```
Successfully installed virtualenv-15.1.0
```

```
[server]$ which virtualenv
```

```
/home/username/opt/python-3.6.2/bin/virtualenv
```

Tworzenie wirtualnego środowiska dla wybranej wersji Pythona

```
[server]$ which python3  
/home/username/opt/python-3.6.2/bin/python
```

```
[server]$ cd ~/example.com
```

```
[server]$ . ~/.bash_profile
```

```
[server]$ virtualenv -p /home/example_username/opt/python-3.6.2/bin/python3 my_project
```

```
Running virtualenv with interpreter /home/example_username/opt/python-3.6.2/bin/python3  
Using base prefix '/home/example_username/opt/python-3.6.2'  
New python executable in /home/example_username/example.com/env/bin/python3  
Also creating executable in /home/example_username/example.com/env/bin/python  
Installing setuptools, pip, wheel...done.
```

```
[server]$ source my_project/bin/activate
```

```
(my_project) [server]$
```

```
[server]$ python -V  
Python 3.6.2
```

Testy jednostkowe

- **Pytest** to framework, który ułatwia tworzenie prostych i skalowalnych testów. Testy są ekspresyjne i czytelne - nie jest wymagany dodatkowy kod.

```
# content of test_sample.py
def func(x):
    return x + 1

def test_answer():
    assert func(3) == 5
```

test session starts, platform linux, Python 3.8.10, pytest-6.2.5, py-1.11.0, pluggy-0.13.1

```
$ pytest
===== test session starts =====
platform linux -- Python 3.x.y, pytest-5.x.y, py-1.x.y, pluggy-0.x.y
cachedir: $PYTHON_PREFIX/.pytest_cache
rootdir: $REGENDOC_TMPDIR
collected 1 item

test_sample.py F [100%]

===== FAILURES =====
_____ test_answer _____

    def test_answer():
>         assert func(3) == 5
E         assert 4 == 5
E         + where 4 = func(3)

test_sample.py:6: AssertionError
===== 1 failed in 0.12s =====
```

Django



- Brak migracji między bazami danych
- Trudność dostosowania gotowych rozwiązań pod własne potrzeby
- Aplikacje Web
- Wzorzec MVC
- Wysoka skalowalność
- Praca z kilkoma bazami danych jednocześnie
- Django CMS
- ORM
- Wygodne tworzenie formularzy
- Python - URLconf
- PythonAnywhere
- Panel administracyjny
- Dynamiczny interfejs administracyjny
- Wydajny system cache'owania
- Wsparcie przed atakami internetowymi
- Kompatybilność wsteczna

Django

Przed instalacją Django, należy najpierw zainstalować pip i virtualenv oraz virtualenvwrapper

C:\Windows\System32\cmd.exe

```
E:\Development\VAIBHAV\Django\Tutorial>mkdir demo_django && cd demo_django
```

```
E:\Development\VAIBHAV\Django\Tutorial\demo_django>python -m virtualenv venv
```

```
Using base prefix 'C:\Users\shvaibha\AppData\Local\Programs\Python\Python37-32'
```

```
C:\Users\shvaibha\AppData\Roaming\Python\Python37\site-packages\virtualenv.py:1041: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
```

```
import imp
```

```
New python executable in E:\Development\VAIBHAV\Django\Tutorial\demo_django\venv\Scripts\python.exe
```

```
Installing setuptools, pip, wheel...done.
```

```
E:\Development\VAIBHAV\Django\Tutorial\demo_django>venv\scripts\activate
```

```
(venv) E:\Development\VAIBHAV\Django\Tutorial\demo_django>pip install django
```

```
Collecting django
```

```
Using cached https://files.pythonhosted.org/packages/ca/7e/fc068d164b32552ae3a8f8d5d0280c083f2e8d553e71ecacc21927564561/Django-2.1.1-py3-none-any.whl
```

```
Collecting pytz (from django)
```

```
Using cached https://files.pythonhosted.org/packages/30/4e/27c34b62430286c6d59177a0842ed90dc789ce5d1ed740887653b898779a/pytz-2018.5-py2.py3-none-any.whl
```

```
Installing collected packages: pytz, django
```

```
Successfully installed django-2.1.1 pytz-2018.5
```

```
(venv) E:\Development\VAIBHAV\Django\Tutorial\demo_django>
```

Django – struktura aplikacji

A terminal window titled '/bin/sh' showing the execution of 'python manage.py' and 'tree' commands in a Django project directory. The 'tree' command output lists the standard Django application files: admin.py, apps.py, __init__.py, migrations (with its own __init__.py), models.py, tests.py, and views.py. It also indicates there is 1 directory and 7 files in total.

```
/bin/sh
(.pve) root@kzkbox:~/Django/malybar$ python manage.py
(.pve) root@kzkbox:~/Django/malybar$ tree
.
├── admin.py
├── apps.py
├── __init__.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py

1 directory, 7 files
(.pve) root@kzkbox:~/Django/malybar$
```

apps.py – ustawienia aplikacji

admin.py – konfiguracja panelu administracyjnego

models.py – modele danych przechowywanych w bazie

views.py – widoki

Flask



- Aplikacje Webowe
- Działa w oparciu o Werkzeug i szablony Jinja 2
- Zintegrowana obsługa testów jednostkowych
- Obsługa bezpiecznych plików cookie
- Zgodność w WSGI
- Dostępne rozszerzenia w celu ulepszenia potrzebnych funkcji
- Kompatybilność z Google App Engine
- Obsługa Unicode
- Routing URL

Pandas



Biblioteka, służąca do manipulacji i analizy danych

- Zapis i odczyt danych między strukturami danych i różnymi formatami plików
- Zintegrowana obsługa brakujących danych
- Przekształcanie i przestawianie zestawami danych
- Indeksowanie dużych zestawów danych
- Wstawianie i usuwanie kolumn struktur danych
- Łączenie różnych zestawów danych
- Hierarchiczne indeksowanie
- Generowanie zakresu dat(przesunięcia dat,opóźnienia)
- Filtracja danych

Tensor Flow



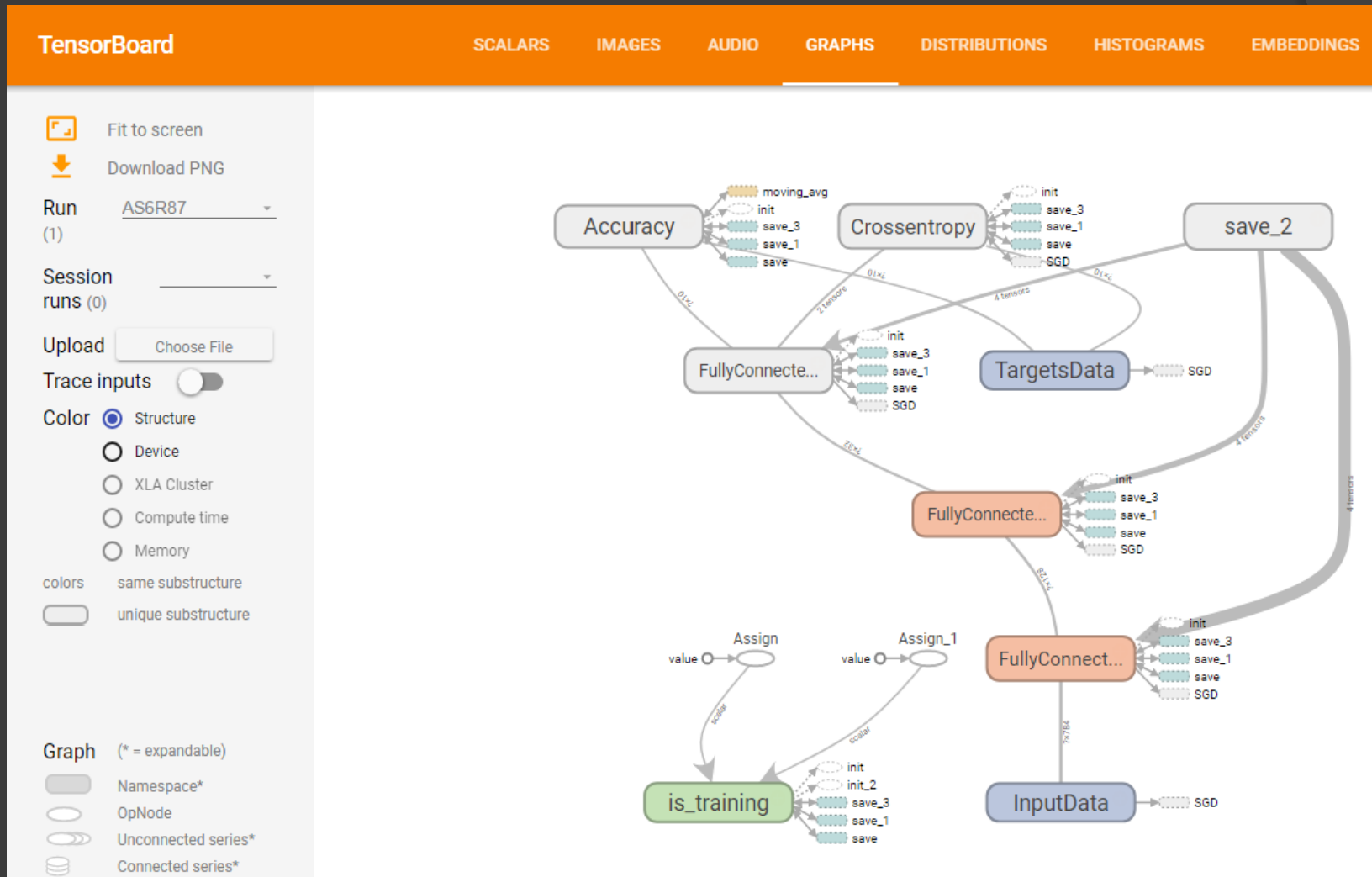
Biblioteka programistyczna, wykorzystywana do obliczeń metod numerycznych, uczenia maszynowego i głębokich sieci neuronowych.

- Umożliwia wdrożenie obliczeń na kilku procesorach
- Bazuje na interfejsie Pythona
- Wysoka wydajność i skalowalność
- Przedstawia wyniki obliczeń w formie grafu
- Obsługuje wiele języków programowania
- TensorBoard
- Łatwe dzielenie się wytrenowanymi modelami

Wady

- Nieuporządkowany kod
- Konieczność dopisywania dodatkowego kodu

TensorFlow



Graficzny przykład kodu w TensorBoard

PyGame



Biblioteka przeznaczona do tworzenia gier komputerowych i aplikacji multimedialnych

Podstawowe moduły w PyGame

- `pygame.init()` – ładowanie modułu pyGame odpowiedzialnego np za dźwięk, grafikę.
- `pygame.display()` – kontrola ekranu lub okna aplikacji.
- `pygame.image()` – zapis, odczyt grafiki.
- `pygame.mouse()` – obsługa myszy.
- `pygame.time()` – obsługa czasu.
- `pygame.transform()` – obróbka grafiki.

Python 2.x a Python 3.x

- Instrukcja `print` staje się funkcją. Nie można więc już w Pythonie 3 napisać: `print 12` można natomiast – tak jak dotąd: `print(12)`
- Iterator `range` wykorzystywany w pętli `for` zwracał w Pythonie 2 listę. W Pythonie 3 jest to obiekt zwracający kolejne elementy (taką funkcję pełni w Pythonie 2 `xrange`). Podobnie jest z funkcją globalną `zip`, która w Pythonie 2 zwracała listę, a w Pythonie 3 zwraca iterator (generator).
- W Pythonie 2 do wprowadzania danych służyło `input` (gdy zachęta była w podwójnych cudzysłowach zwracał napis; inaczej – zależnie od tego wprowadzono). Natomiast `raw_input` zawsze zwracał łańcuch (string). W Pythonie 3, funkcja `raw_input()` jest przestarzała, a `input` zawsze zwraca ciąg znaków (string).
- Dzielenie liczb w Pythonie 2 zależy od tego, czy liczba jest całkowita. Całkowite (integer) zaokrągla w dół. $5.0/2 == 2,55/2 == 2$
- W Pythonie 3 dzielenie bez reszty jest zmiennoprzecinkowe.
- Inaczej rozwiązano znaki narodowe (w Pythonie 3 napisy są w Unicode)
- W Pythonie 3 zyskujemy większe możliwości formatowania wyjścia na konsolę.
- Python w wersji 2.x nie będzie już wspierany

Materiały dodatkowe:

za friko

- ◉ <https://www.flynerd.pl/tag/python-kurs>
- ◉ <https://www.youtube.com/watch?v=rfscVS0vtbw>
- ◉ <https://www.learnpython.org/pl/>
- ◉ <https://checkio.org/>
- ◉ <https://www.techbeamers.com/python-tutorial-step-by-step/>
- ◉ <https://github.com/kitao/pyxel>