

#### **Trener**

### **Przemysław Lalak**

Python Engineer & Team Leader

@

Adverity

- Slack: @Przemysław Lalak <u>isapythonergohestia1.slack.com</u>
- Repozytorium:

https://github.com/infoshareacademy/pythonprogramista1EH



# Poznajmy się!



# Python dla programistów!















# Organizacja, materiały i pomoc

#### Warto rozmawiać!

Pytać, zgłaszać się, itd.

- https://docs.python.org
- Dokumentacja bibliotek i pakietów
- Trener
- StackOverflow
- Repozytorium



# Harmonogram i plan na najbliższe kilka tygodni



# Przygotowanie



# Sprawdźmy narzędzia Git i Github





https://github.com/infoshareacademy/infoshareacademy-pythonergohestia



# Sprawdźmy narzędzia PyCharm i Python

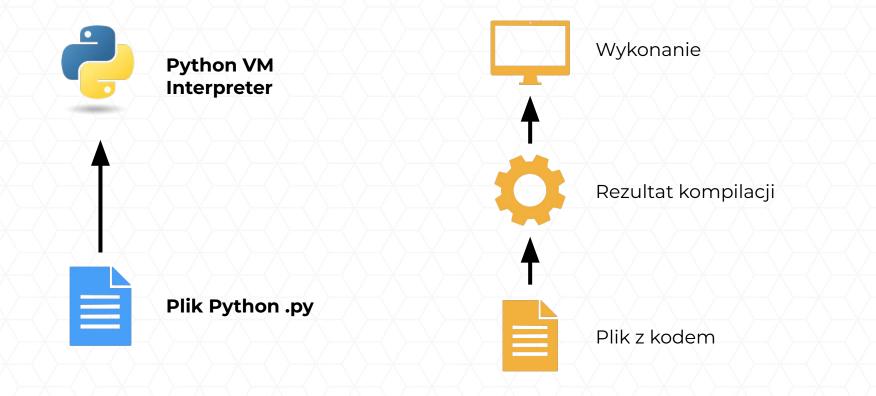




Nowy projekt, wirtualne środowisko



# Interpretowany vs kompilowany





# Jak PyCharm będzie pomagał



Python VM Interpreter

Integracja z Gitem



Dokumentacja

Zarządzanie pakietami



Plik Python .py

Code linters

https://google.github.io/styleguide/pyguide.html https://peps.python.org/pep-0008/



# Sprawdźmy narzędzia Jupyter



- > pip install jupyter
- > jupyter notebook



Zmienne, typy danych i cała reszta



### **Zmienne**

**Zmienna** to nazwany obszar w pamięci komputera.

Posiada wartość i typ.

Zarządzanie pamięcią jest po stronie Pythona. Nieużywane zmienne są usuwane przez Garbage Collector.



# Przypisywanie wartości

Aby utworzyć zmienną i przypisać wartość wykorzystujemy operator "=".

**Typ zmiennej** jest określany na podstawie **typu przypisywanej wartości** w czasie wykonywania programu - **dynamic typing.** 

result 
$$= 5 + 10$$

**NOTE**: Do nazw zmiennych stosujemy **lower\_case\_with\_underscores**, również nazywane **sneak\_case**.



# Podstawowe typy liczbowe

#### **INTEGER** int()

my integer = 1

#### **FLOATING POINT** float()

my float = 1.0

Możemy je mieszać i używać wszystkich matematycznych operatorów:



# Typ string

#### **STRING** str()

Sekwencja znaków.

```
user name = "Adam"
```

user surname = 'Smith'

Stringi można łączyć dodając je do siebie. Można też mnożyć przez liczbę całkowitą.



# Typ string jako iterable

**STRING** str() jako sekwencja znaków jest "iterowalny". Możemy sięgnąć do poszczególnych elementów stosując **indexing**.

```
user_name = "Adam"

user_name[0] \( \times \) Zwraca pierwszy znak (index 0)

user_name[1] \( \times \) Drugi znak (index 1)

user_name[-1] \( \times \) Ostatni znak (index -1)
```



## Typ string jako iterable

Możemy również pobrać fragment stosując slicing.

Slicing to pobieranie fragmentu formułą:

```
[start : stop (nie uwzględnione) : krok]
```



Typy mogą posiadać zdefiniowane **metody**. Metody to funkcje powiązane z określonym typem i obiektami tego typu.

Żeby dostać się do metod typu Str, potrzebujemy obiektu typu Str oraz operatora DOT (".").



Metody posiadają dostęp do wartości obiektu który je wywołuje. Nie musimy jej przekazywać tak jak do funkcji.

"rafal".capitalize() rezultat to "Rafal"

**NOTE**: Str jest typem niemutowalnym! Metody Str zwracają nową wartość. Nie modyfikują obiektu który je wywołał.



Co jest do naszej dyspozycji?

https://docs.python.org/3/library/stdtypes.html#string-methods





# Formatowanie typu string

Jedna z metod pozwala nam formatować string. Klamry { } oznaczają gdzie zostaną wstawione wartości.

```
user_value = 42
"Your value is: {}".format(user_value) → "Your value is: 42"
```

To samo możemy osiągnąć przy pomocy stringa formatowanego:

```
user_value = 42

f"Your value is: {user_value}" → "Your value is: 42"
```



# Formatowanie typu string

Klamry { } mogą również zawierać formułę formatowania.

https://docs.python.org/3/library/string.html#formatstrings

#### Np.:

```
{:.2f} Decimal precision, two places, format as floating point number
{:.3%} Decimal precision, three places, format as percentage
{:02} Fill with zeros, up to two places
```



# Dziękuje!