

# PROGR@MUJ W ZESPOLE MATERIAŁY DLA UCZNIÓW



Progr@muj w zespole

# PROGR@MUJ W ZESPOLE MATERIAŁY DLA UCZNIÓW

ADAM JURKIEWICZ  
RAFAŁ KAMIŃSKI  
KONRAD KOSIERADZKI  
ELŻBIETA PIOTROWSKA-GROMNIAK

2022

Innowacja społeczna „**Progr@muj w zespole**”  
została zrealizowana dzięki wsparciu uzyskanemu  
w ramach projektu „POPOJUTRZE 2.0 – KSZTAŁCENIE”  
nr POWR.04.01.00-00-I108/19  
współfinansowanego ze środków Unii Europejskiej  
w ramach Europejskiego Funduszu Społecznego  
oraz środków budżetu państwa

**Realizacja projektu:**

Liceum Ogólnokształcące z Oddziałami Dwujęzycznymi  
im. Adama Mickiewicza w Piastowie

**Autorzy:**

Adam Jurkiewicz  
Rafał Kamiński  
Konrad Kosieradzki  
Elżbieta Piotrowska-Gromniak

**Projekt graficzny i skład:**

Zuzanna Piwońska

**Konsultacje i wsparcie:**

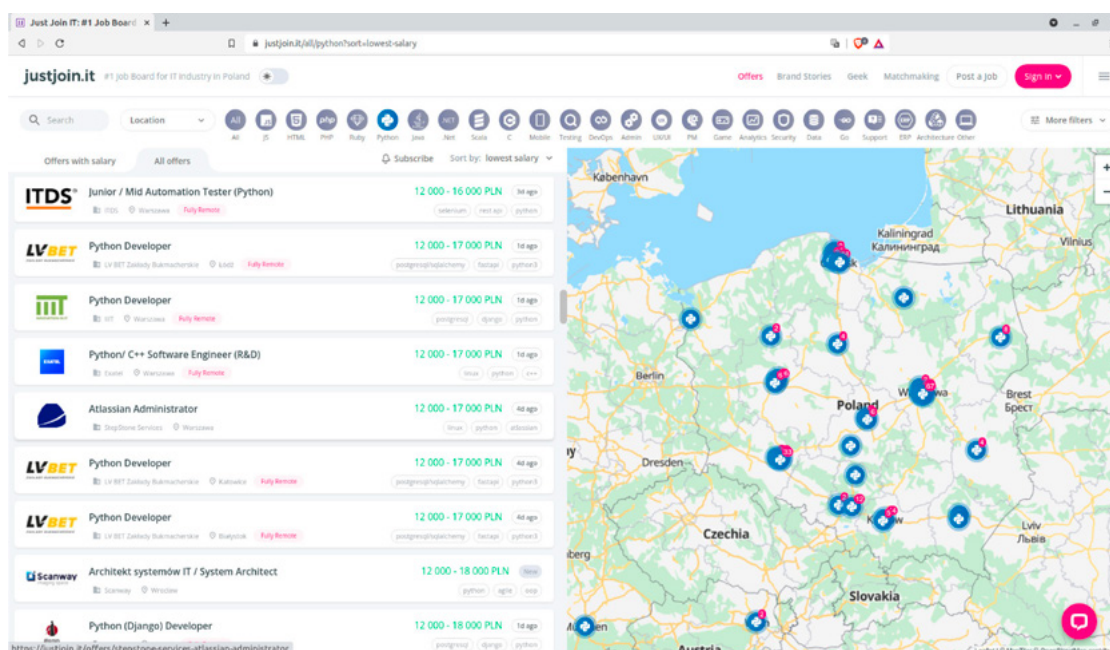
Marcin Bielicki  
Agata Pakieła  
Łukasz Wolff

# PRZEWODNIK DLA UCZNIÓW.

## *Droga Uczennico, Drogi Uczniu,*

Wyobraź sobie, że dzięki serwisowi <https://JustJoin.it> znajdujesz pracę jako Mid Software Developer, widełki 17 000 ~ 20 000 PLN miesięcznie netto B2B.

Nieźle - prawda?



Powyżej strona <https://justjoin.it> z przykładowymi ofertami pracy.

OK, lecz w zamian wykaż się wiedzą, znajomością Pythona, programowania... i, co najważniejsze, otwartością oraz chęcią współpracy z kolegami i kolegami z zespołu.

W Twojej nowej pracy znajdujesz się w zespole, który opracowuje aplikację dla klienta - holenderskiej firmy, która chce wejść na rynek amerykański z nowymi aplikacjami. Do tego celu chce opracować 3 aplikacje w modelu MVP.

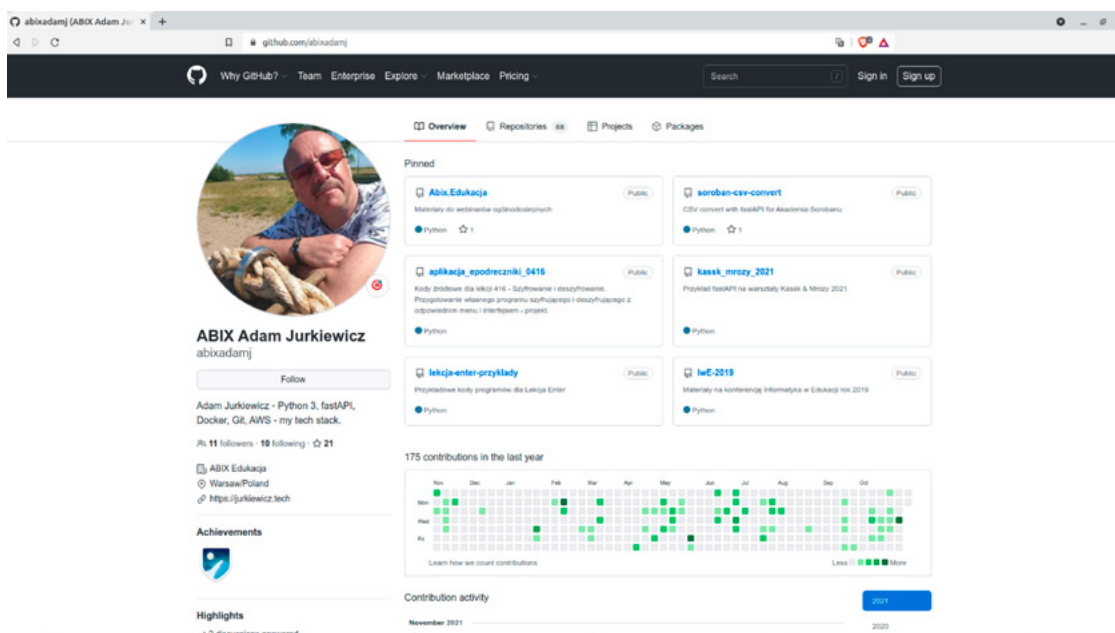
MVP to skrót od Minimum Viable Product. Tłumacząc na język polski będzie to „Produkt o Minimalnych Funkcjonalnościach”. Co to dokładnie oznacza? Z założenia jest to pierwsza wersja projektu, produktu, aplikacji czy progra-

mu, która jest gotowa, by wprowadzić go na rynek. Taka wersja naszego produktu skupia się na jego głównych cechach i na tym etapie pominięte są wszelkie szczegóły – to jest właśnie cecha odróżniająca MVP od skończonego projektu. Dzięki MVP można wypuścić projekt na rynek i zbadać, czy klienci są nim zainteresowani. Takie rozwiązanie pozwala niekiedy zaoszczędzić sporo czasu i pieniędzy w przypadku, gdyby okazało się, że projekt nie podbił rynku albo część funkcjonalności skomplikowanego projektu jest zbędna. Gdy projekt MVP zostaje pozytywnie przyjęty przez klientów, staje się świetną bazą do rozbudowy i może zostać wzbogacony i udoskonalony.

Klient chce realizować jedną z trzech poniższych aplikacji desktopowych (a więc działających na komputerze, z systemami: Linux, MacOS, Windows):

- Lokalizacje lotów samolotów: <https://aviationstack.com> (w filmach używamy tego projektu),
- Weryfikacja numerów telefonów z podaniem danych o operatorze i kraju: <https://numverify.com>,
- Alert pogodowy dla 3-dniowej prognozy: <https://wttr.in/>.

W trakcie kilkunastu etapów pracy Twój zespół zbuduje wybraną aplikację, a wszyscy w zespole będą ze sobą współpracować, wykorzystując narzędzia informatyczne, m.in. serwis GitHub do przechowywania plików projektu i, przede wszystkim, komunikując się ze sobą wszystkimi dostępnymi kanałami, w „realu” i online.



Z wideotutoriali dowiesz się, jak wykorzystać potencjał języka programowania Python do wytworzenia MVP.

Twoja nauka i praca będzie podzielona na 11 zajęć/spotkań - tzw. sprintów, dokładnie tak, jak to wygląda w prawdziwym miejscu pracy w dzisiejszych czasach.

Podczas wykonywania zadania nauczysz się wykorzystywać środowisko programistyczne PyCharm, a w nim tworzyć wirtualne środowisko Pythona. Dowiesz się, jak manipulować złożonymi typami danych w Pythonie na przykładzie wymiany danych pomiędzy serwisami z wykorzystaniem typu JSON (w Pythonie to słownik). Te umiejętności przydadzą Ci się w przyszłości - niezależnie od tego, czy zostaniesz programistą, czy też będziesz miał do czynienia z aplikacjami jako zlecający, klient, lub, po prostu, jako pasjonat Pythona.

Podczas tworzenia aplikacji konieczne będzie stworzenie grafiki dla aplikacji, strony internetowej reklamującej produkt i dokumentacji użytkowej. Oczywiście, w modelu MVP wszystkie te elementy nie muszą być rozbudowane, wystarczające jest opracowanie podstawowych elementów. Zapoznasz się z określonymi zasadami dotyczącymi ich tworzenia, a więc z podstawami HTML, oprogramowania Bluefish, GIMP i Libre Office Writer - gdyż do tworzenia tych elementów użyjesz tylko oprogramowania Open Source, a więc:

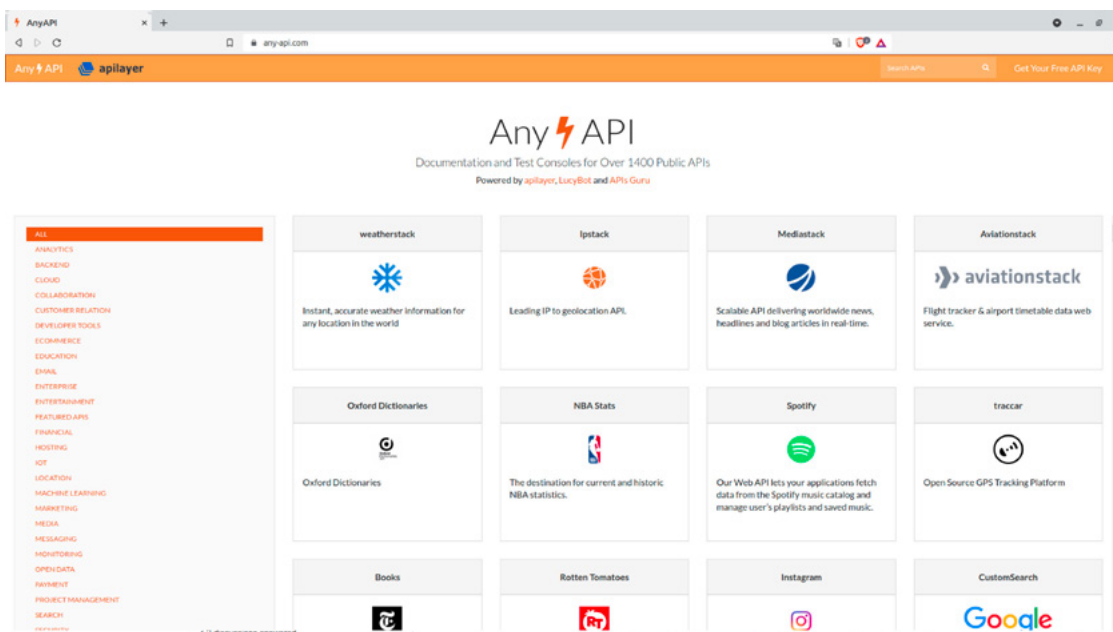
- legalnego,
- darmowego,
- dostępnego dla różnych systemów operacyjnych (Linux, macOS, Windows).

Wybrane projekty MVP to aplikacje korzystające z otwartych danych, dostępnych za pomocą API (Application Programming Interface). Cenny opis API znajdziesz w serwisie:



<https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/>

Do opracowania swojej aplikacji możesz korzystać z ogólnodostępnej bazy API:



<https://any-api.com/>

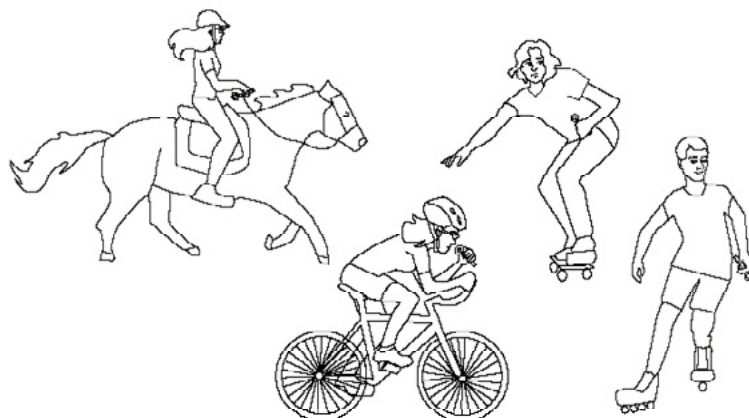
Podczas projektu zaznajomisz się z różnymi sposobami tworzenia aplikacji w Pythonie, które wykorzystasz do pracy nad wybranym MVP. Będą to wideotutoriale osadzone w środowisku Moodle, które jest dedykowane nauczaniu samodzielnemu w oparciu o pewną ścieżkę działania.

Dodatkowo otrzymasz wsparcie w postaci przykładowych kodów źródłowych, które będą stanowić dla Ciebie punkt odniesienia wobec aplikacji opracowywanej w zespole. Przygotowaliśmy również tzw. „cheatsheet” – ściągawkę z najważniejszymi konstruktami języka Python, abyś mógł błyskawicznie przypomnieć sobie poprawną składnię tworzonego kodu. Prawda, że ta praca zapowiada się całkiem interesująco?

Tak, jak dowiedziałeś się na samym początku, nad projektem pracuje zespół, a Ty jesteś jego członkiem.

Dlaczego tak wygląda to zadanie? Bo, jak pokazuje praktyka w firmach IT, taka forma najlepiej się sprawdza w wielowątkowych zadaniach/projektach, w których potrzebne są różne umiejętności i kompetencje.





Każdy z nas jest inny, każdy ma swoje unikalne talenty i mocne strony (choćby najbardziej oryginalne), zaś różny sposób patrzenia na rzeczywistość pomaga zespołowi znaleźć najlepsze rozwiązania. Żeby tę różnorodność osobowości Twoich kolegów i koleżanek z zespołu wykorzystać w sposób optymalny, trzeba najpierw zdać sobie sprawę, że ona jest, czyli po prostu przyjąć do wiadomości, że ktoś, kogo spotykasz, z kim pracujesz, uczysz się, może być zupełnie inny niż Ty sam. Jaki ten ktoś jest, masz szansę dowiedzieć się, jeśli będziesz rozmawiał, słuchał, zadawał pytania, starał się zrozumieć nie oceniając, czyli: gdy będziesz komunikował się. Niby proste, ale jak życie pokazuje, trzeba w to włożyć trochę wysiłku. Ale spokojnie, firma zadba o Ciebie. Dostaniesz wsparcie w postaci krótkich filmów, które rozjaśnią tę sferę.

Dowiesz się:

- kiedy grupa ludzi ma szansę stać się zespołem,
- jak formułować swoje wypowiedzi w sposób klarowny i jak sprawdzić czy zostały właściwie zrozumiane,
- jak zadawać pytania i upewniać się, że właściwie zrozumiałeś odpowiedź,
- jak poradzić sobie, gdy pojawi się różnica zdań lub konflikt w zespole,
- od czego zależy sukces Waszego zespołu - bo przecież chcecie go osiągnąć.





Sukcesem będzie wykonanie przyjętego zadania, osiągnięcie celu projektu. Ten cel jest wspólny i ważny dla wszystkich członków zespołu. Każdy z Was ma na ten cel wpływ i odpowiada za niego w równym stopniu, bo jeśli ktoś nawali i nie zrobi tego, czego się podjął, to zniweczy pracę innych. Meczu nie wygrywa jeden zawodnik, tylko cały zespół.

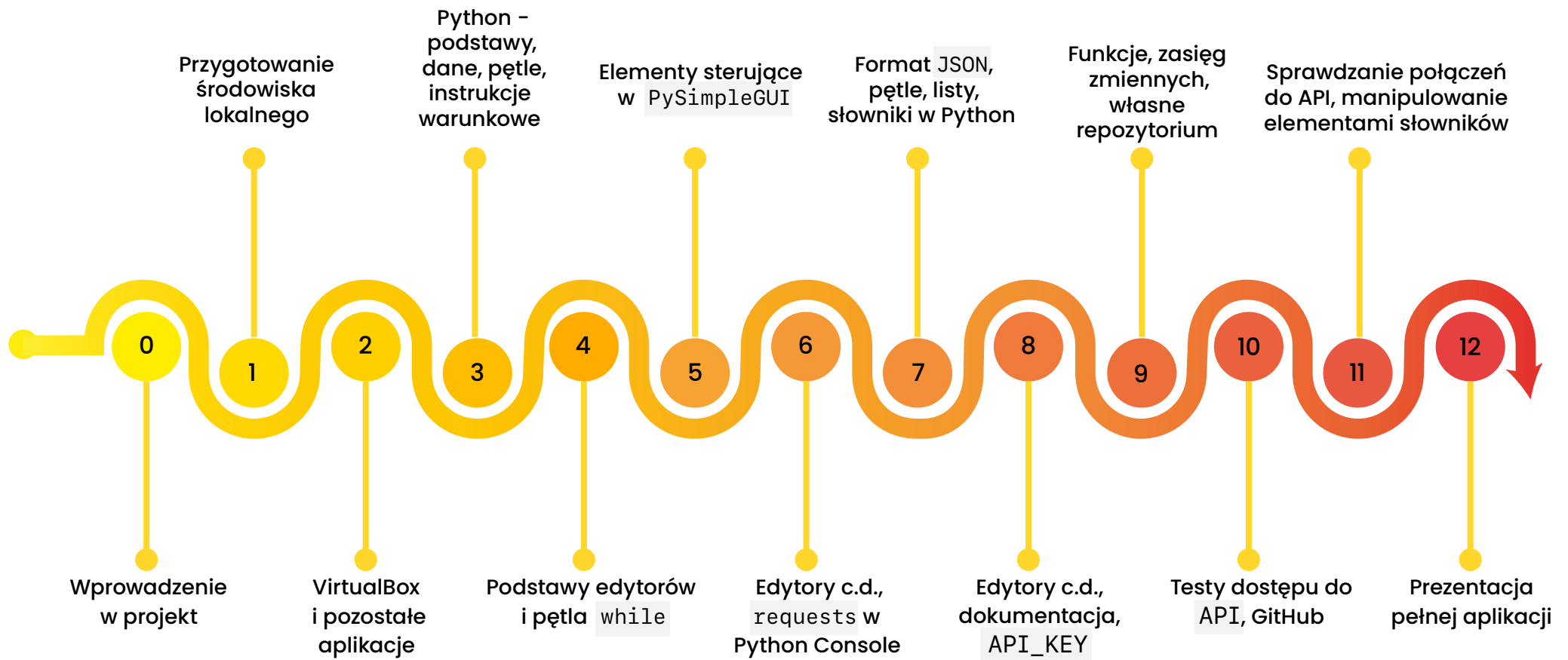
W życiowych realiach bardzo rzadko masz wpływ na skład zespołu, w którym się znalazłeś. I tak jest też w tym przypadku. Nie masz żadnego wpływu na to z kim przyszło Ci pracować - ale masz wpływ na samego siebie.



Dlatego tak ważna jest otwartość na wspomnianą już różnorodność i odmienność u innych. Opuuszczając trochę własne ambicje na rzecz wspólnego celu, dzieląc zadania do wykonania, wspierając się i współpracując możesz odkryć bogactwo wspólnych zasobów zespołu i ów cel osiągnąć.

***Po to przecież firma powołała Wasz zespół!***

***Zapraszamy do tego projektu!***



## plan pracy



## **Zajęcia 0 – Wprowadzenie w projekt** **Podczas zajęć dowiesz się:**

- w jaki sposób będzie przebiegać praca w trakcie cyklu zajęć,
- jakie zasady warto stosować w pracy zespołowej,
- jakie są Twoje kompetencje w zakresie komunikacji i współpracy w zespole.



## **Zajęcia 1 - Przygotowanie środowiska – konta, oprogramowanie** **Podczas zajęć dowiesz się:**

- jakie są podstawowe zasady funkcjonowania środowiska GitHub,
- jak założyć konto w serwisie GitHub oraz zmienić jego ustawienia,
- jak skonfigurować wirtualne środowisko dla pojedynczego projektu.



## **Zajęcia 2 - Virtualbox i pozostałe aplikacje** **Podczas zajęć dowiesz się:**

- jakie są podstawowe zasady funkcjonowania systemu VirtualBox,
- jakie znaczniki meta występują w dokumencie HTML,
- jak tworzyć dokumenty w edytorze tekstów,
- jakie są rodzaje bitmapowych (rastrowych) formatów graficznych.



## **Zajęcia 3 – Python - podstawy, dane, pętle, instrukcje warunkowe** **Podczas zajęć dowiesz się:**

- jakie są podstawowe typy danych w Python,
- w jaki sposób stosować pętle iteracyjne w połączeniu z kolekcjami danych,
- jak używać instrukcji warunkowej w Python,



## **Zajęcia 4 - Podstawy edytorów i pętla `while` w Python** **Podczas zajęć dowiesz się:**

- o rodzajach elementów containers dokumentu HTML,
- jak użyć listy numerowanej lub nienumerowanej w edytorze tekstu,
- jakie warstwy istnieją w edytorze grafiki i jak nimi manipulować,
- jak korzystać z biblioteki PySimpleGUI,
- jak wykorzystać pętlę nieskończoną `while True` i w jaki sposób ją przerwać.



## **Zajęcia 5 – Elementy sterujące w PySimpleGUI** **Podczas zajęć dowiesz się:**

- jak stworzyć aplikację zorientowaną na zdarzenia,
- jak zdefiniować wygląd aplikacji desktopowej,
- w jaki sposób wywoływać funkcję w zależności od zdarzenia,
- jak pobierać dane wpisane przez użytkownika.



## Zajęcia 6 – Dalsze działania w edytorach, moduł requests w Python Console

### Podczas zajęć dowiesz się:

- jakie są podstawowe zasady wywołań modułu `requests`,
- jak wykorzystać Python Console w środowisku PyCharm,
- jak wywołać połączenie `GET` do zewnętrznego API w języku Python.



## Zajęcia 7 – Format JSON, pętle, listy, słowniki w Python

### Podczas zajęć dowiesz się:

- jakie są zasady tworzenia list i słowników w Python,
- jak wykorzystać pętlę `for` do przetwarzania obiektów sekwencyjnych,
- jak wykorzystać połączenie `requests.get()` w skrypcie,
- które obiekty w Python są `mutable`, a które `immutable`.



## Zajęcia 8 – Edytory, dokumentacja i `API_KEY` dla przykładowych projektów

### Podczas zajęć dowiesz się:

- jakie są formaty plików graficznych oraz tekstowych,
- jak wykorzystać różne obiekty w języku HTML,
- jak zalogować się do serwisu API i wygenerować `API_KEY`.



## **Zajęcia 9 – Funkcje, zasięg zmiennych - NAMESPACE, własne repozytorium**

### **Podczas zajęć dowiesz się:**

- jak stworzyć repozytorium na GitHub i wybrać odpowiednią licencję,
- jak zdefiniować i wywołać funkcję w języku Python,
- jak działają przestrzenie nazw i jak dostosować do nich kod.



## **Zajęcia 10 – Testy dostępu do API oraz współpraca z GitHub** **Podczas zajęć dowiesz się:**

- jak dodać współpracowników do repozytorium GitHub,
- jaki jest zakres testowania dostępu do zewnętrznego API.



## **Zajęcia 11 – Sprawdzanie połączeń do API, manipulowanie elementami słowników**

### **Podczas zajęć dowiesz się:**

- w jaki sposób analizować kod programu w poszukiwaniu błędów,
- w jaki sposób przetestować dostęp do zewnętrznego API za pomocą skryptu.

# ● co ciekawego w programie? ●

15

## **Zajęcia 12 – Prezentacja rezultatów pracy zespołów i podsumowanie projektu**

### **Podczas zajęć:**

- wraz z zespołem zaprezentujesz efekty Waszej pracy,
- poznać wyniki pracy innych zespołów i będziesz mógł je skomentować,
- dokonasz oceny własnych kompetencji współpracy i komunikacji w zespole.



## APLIKACJA DEMO - PRZYKŁADOWY KIERUNEK PRAC

*Jeśli nie chcesz czekać do rozpoczęcia zajęć - możesz już dziś przekonać się, jak może wyglądać i działać podobna aplikacja. Wybraliśmy wykonanie prostej aplikacji, która sprawdza publiczny adres IP komputera, z którego jest uruchamiana. Wykorzystujemy do tego celu darmowe API zawarte w serwisie <https://ipify.org>.*

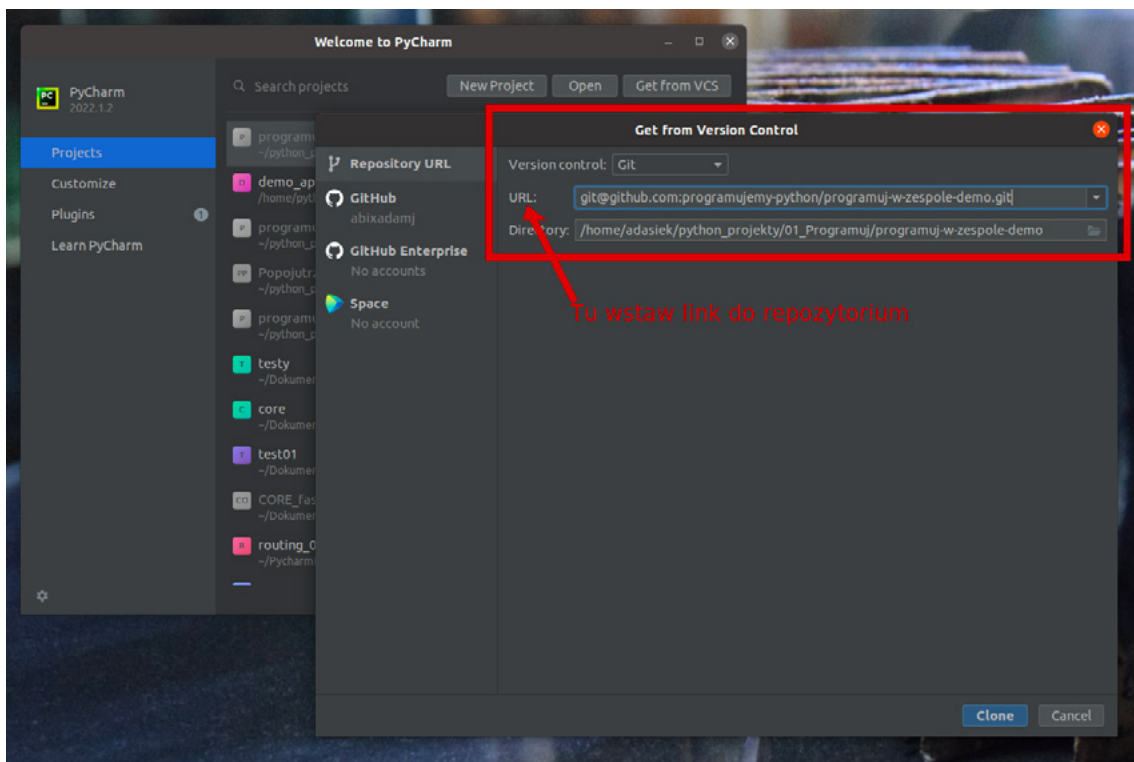
*Poniżej instrukcja, która pokazuje, w jaki sposób możesz samodzielnie testować aplikację DEMO.*



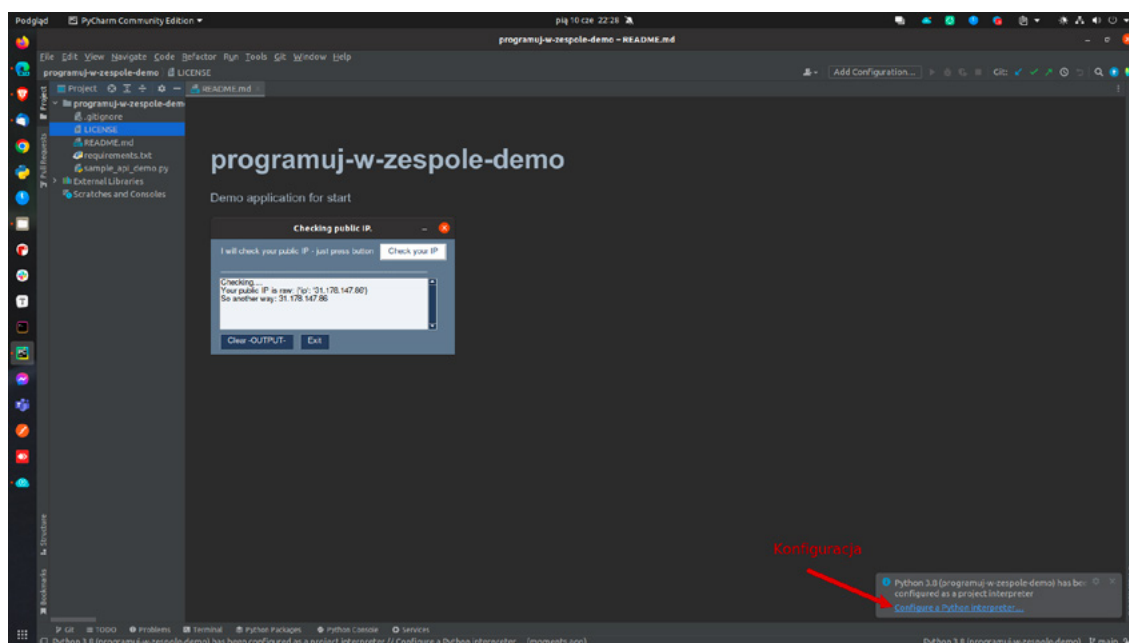
Pełne kody: <https://github.com/programujemy-python/programuj-w-zespole-demo>

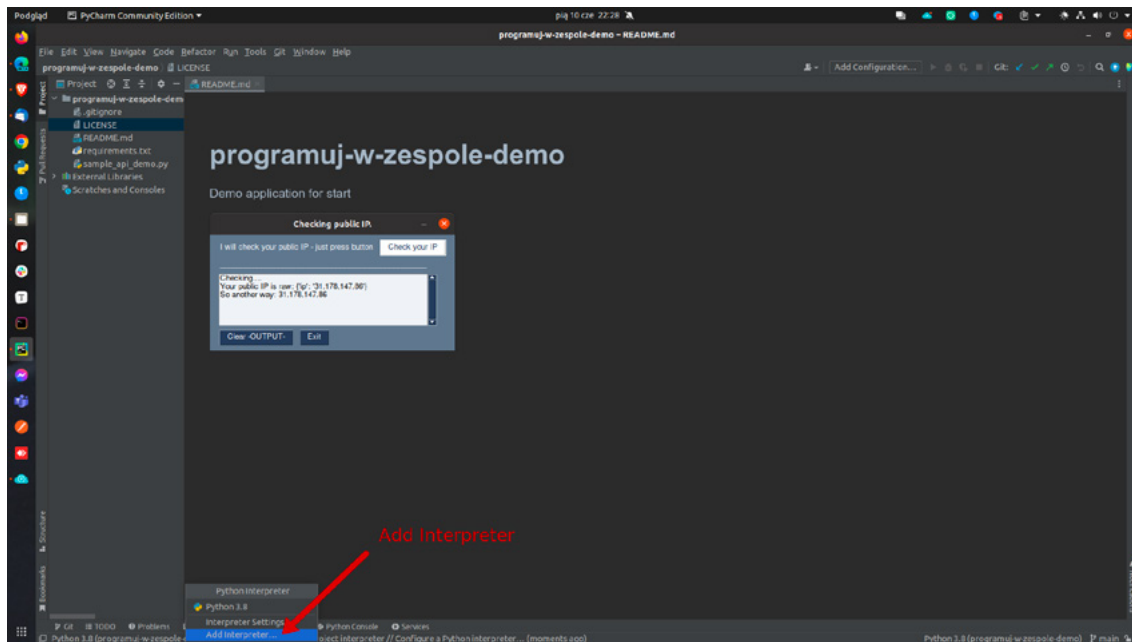
Kolejne kroki, aby uruchomić aplikację:

- W PyCharm wybieramy opcję Get from VCS, a w URL wpisujemy:-  
`git@github.com:programujemy-python/programuj-w-zespole-demo.git`

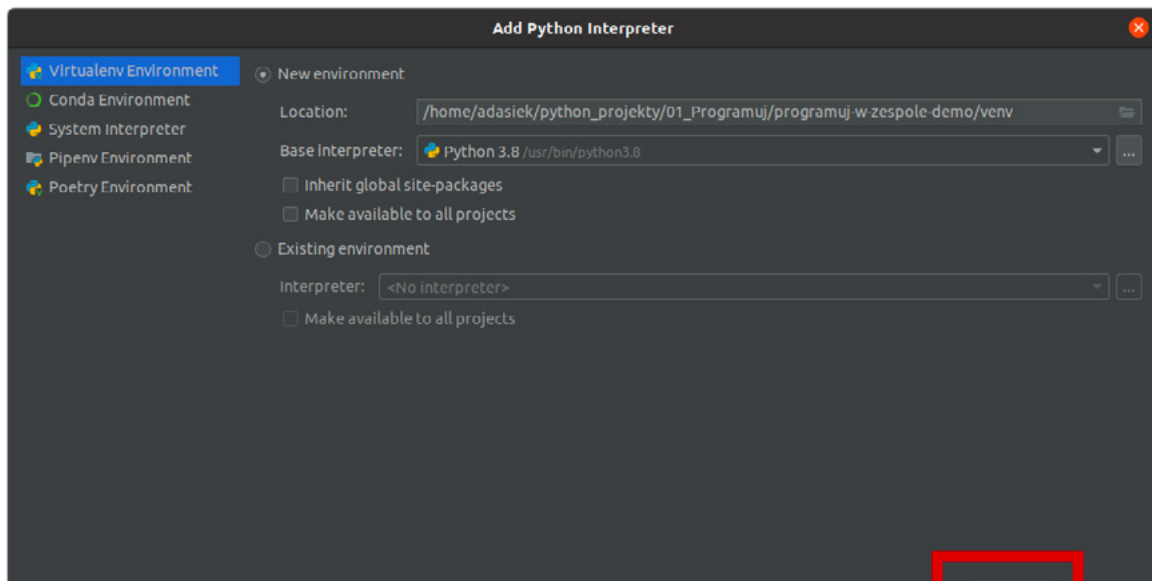


- Wybieramy konfigurację Interpretera

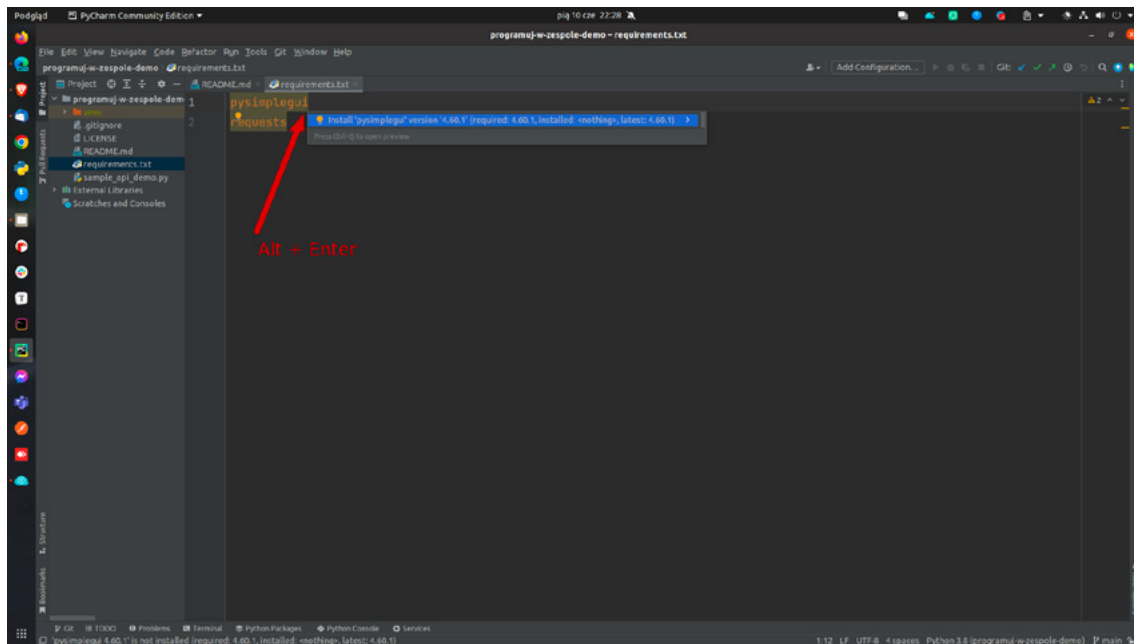




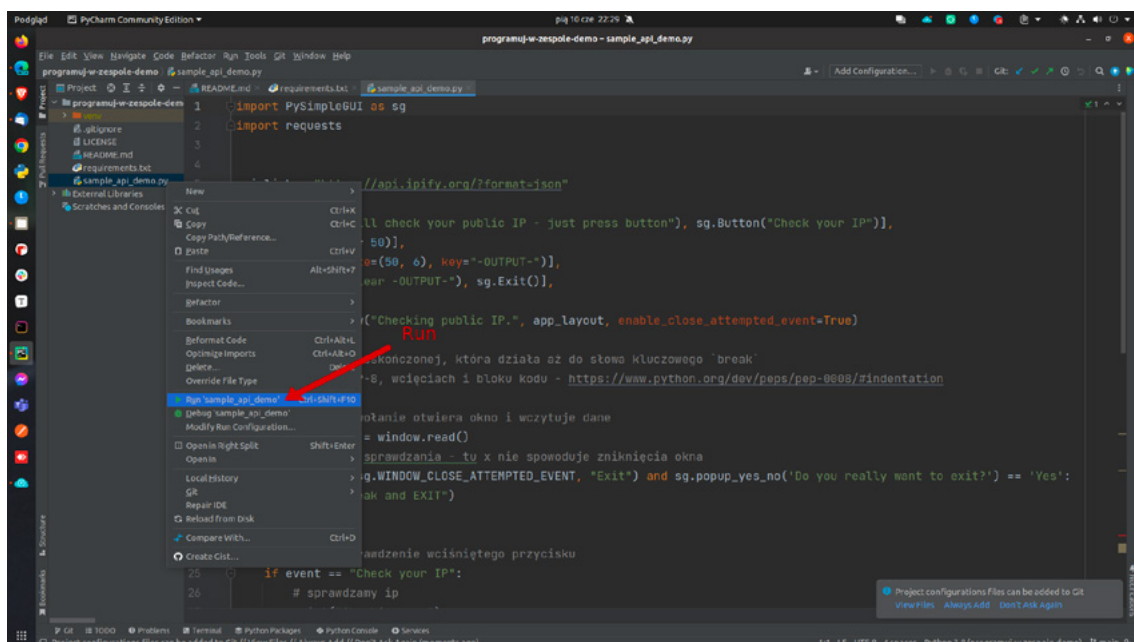
- Dodajemy nowe `venv` - w opcjach nic nie zmieniamy, wybieramy wartości domyślne.



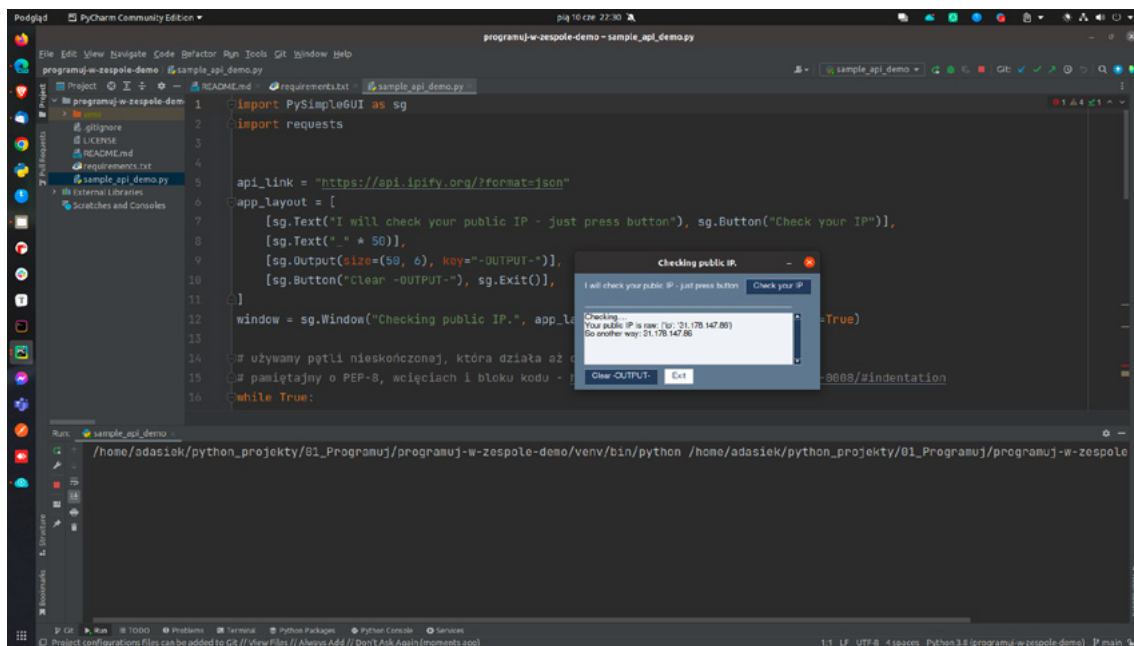
- W pliku `requirements.txt` mamy opisane niezbędne biblioteki - instalujemy je za pomocą skrótu `Alt+Enter` i wybraniu `Install` na końcu linii w pliku



- Wybieramy Run z menu kontekstowego (dostępnego po naciśnięciu prawego przycisku myszy na nazwie pliku sample\_api\_demo.py)



- Obserwujemy działającą aplikację;



```

1 import PySimpleGUI as sg
2 import requests
3
4
5 api_link = "https://api.ipify.org/?format=json"
6
7 app_layout = [
8     [sg.Text("I will check your public IP - just press button"), sg.Button("Check your IP")],
9     [sg.Text("- " * 50)],
10    [sg.Output(size=(50, 6), key="-OUTPUT-")],
11    [sg.Button("Clear -OUTPUT-"), sg.Exit()],
12]
13
14 window = sg.Window("Checking public IP.", app_layout)
15
16 # używamy pętli nieskończonej, która działa aż do przycisku Exit
17 # pamiętajmy o PEP-8, wcięciach i bloku kodu
18 while True:
19     window.refresh()
20     if sg.ButtonClicked(window["Check your IP"]):
21         response = requests.get(api_link)
22         ip_data = response.json()
23         ip_address = ip_data["ip"]
24         sg.Print(f"Your public IP is now: {ip_address}")
25         sg.Print(f"Go another way: 31.178.147.80")
26
27     if sg.ButtonClicked(window["Clear -OUTPUT-"]):
28         sg.Print("")
29
30     if sg.ButtonClicked(window["Exit"]):
31         window.close()
32         break
33
34 window.close()
  
```

## Podstawowe typy danych

```
# komentarz – nic nie wnosi, poza informacjami

# definiujemy zmienne – typy podstawowe
# nazwy zmiennych zapisujemy małymi literami,
# oddzielając słowa znakami podkreślenia
language = "Python" # str – ciąg znaków
year_of_birth = 1991 # int – liczba całkowita
value_pi = 3.1415927 # float – liczba rzeczywista, zmiennoprzecinkowa
python_is_cool = True # bool – wartość logiczna (prawda/fałsz)

# F-string
info = f"The language {language} was made in {year_of_birth} year."
```

## Importowanie zewnętrznych bibliotek

```
# importujemy i nazywamy naszą nazwą
import urllib.request as ureq

# inne sposoby importowania
from random import randint, random
from time import sleep
from turtle import *
```

## Listy

```
# definiujemy listę – kolekcję elementów,
# które zachowują kolejność wprowadzania
```

```
planes = [
    "Boeing 737-3600",
    "Boeing 747 Jumbo Jet",
    "Airbus A3806",
    "Bombardier CRJ-900",
]
```

```
# sprawdzamy czy element istnieje w liście – operator 'in'
print("Bombardier Dash 8" in planes)
```

```
# definiujemy słownik – kolekcję elementów
# o strukturze klucz:wartość
# które nie muszą zachowywać kolejności
```

```
planes = {
    "Airbus A380": 850,
    "Bombardier CRJ-900": 90,
    "Boeing 747 Jumbo Jet": 550,
    "Airbus A330": 300,
}
```

```
# sprawdzamy czy element istnieje w słowniku
# – szukamy tak tylko wśród kluczy
print("Boeing 737-300" in planes)
```

## Pętle

```
# pętla iteracyjna – wykonuje się tyle razy,
# ile elementów posiada kolekcja
for plane in planes:
    print(f"Samolot: {plane}")

# dla zainteresowanych – wykorzystanie funkcji enumerate()
for counter, each_header in enumerate(headers):
    print(f"Header number {counter} is: {each_header}")
```

## Instrukcje warunkowe if ... else ...

```
if age >= 18:
    print(f"Oh, I see, {my_name} – you are an adult now.")
    print(f"You are {age} years old.")
else:
    print(f"You are young – {age} years old.")

# inny sposób
if "Apache" in server_name:
    print("OK, Apache server once again...")
else:
    print("Some exception.")
```

## Tuple – rozpakowywanie tupli...

```
# tupla – prawie jak lista – lecz nie można dodawać, odejmować
# czy zmieniać jej elementów
my_tuple = ("Windows", "Linux", "macOS")

# rozpakowywanie tupli – przypisanie kolejnych elementów do
# kolejnych zmiennych
os_a, os_b, os_c = my_tuple

# można zapisać to również w inny sposób
os_a, os_b, os_c = ("new Windows", "new Linux", "new macOS")
```

## Pętla nieskończona – wykorzystana z PySimpleGUI...

```
# używamy pętli nieskończonej, która działa aż do słowa
# kluczowego 'break'
# pamiętajmy o PEP-8, wcięciach i bloku kodu –
# https://www.python.org/dev/peps/pep-0008/#indentation
while True:
    # poniższe wywołanie otwiera okno i wczytuje dane
    event, values = window.read()
    if event == sg.WIN_CLOSED or event == "Exit":
        print("Hard EXIT")
        break
```



## PySimpleGUI

```
# wyświetlanie informacji tekstowych wewnątrz okien PySimpleGUI
# wczytujemy niezbędne elementy
import PySimpleGUI as sg

# proste okno z informacją i zdefiniowanym tytułem
sg.popup("Hej – witamy w zespole w projekcie Progr@muji w zespole!", title="Progr@muji w zespole")
```