

MATURA

Deklaracja Maturalna

- Każdy uczeń ma prawo wybrać:
 - System operacyjny
 - Język programowania
 - Środowisko programistyczne
 - Program użytkowy

Jakie mamy opcje?

- System operacyjny:
 - Windows
 - Linux
- Język programowania:
 - C++
 - Java
 - Python
- Środowisko programistyczne:
 - Zależne od języka

Czy wszystkie opcje są tak samo dobre?

WYKŁAD
OZNACZENIA
MOCY

Czy wszystkie opcje są tak samo dobre?

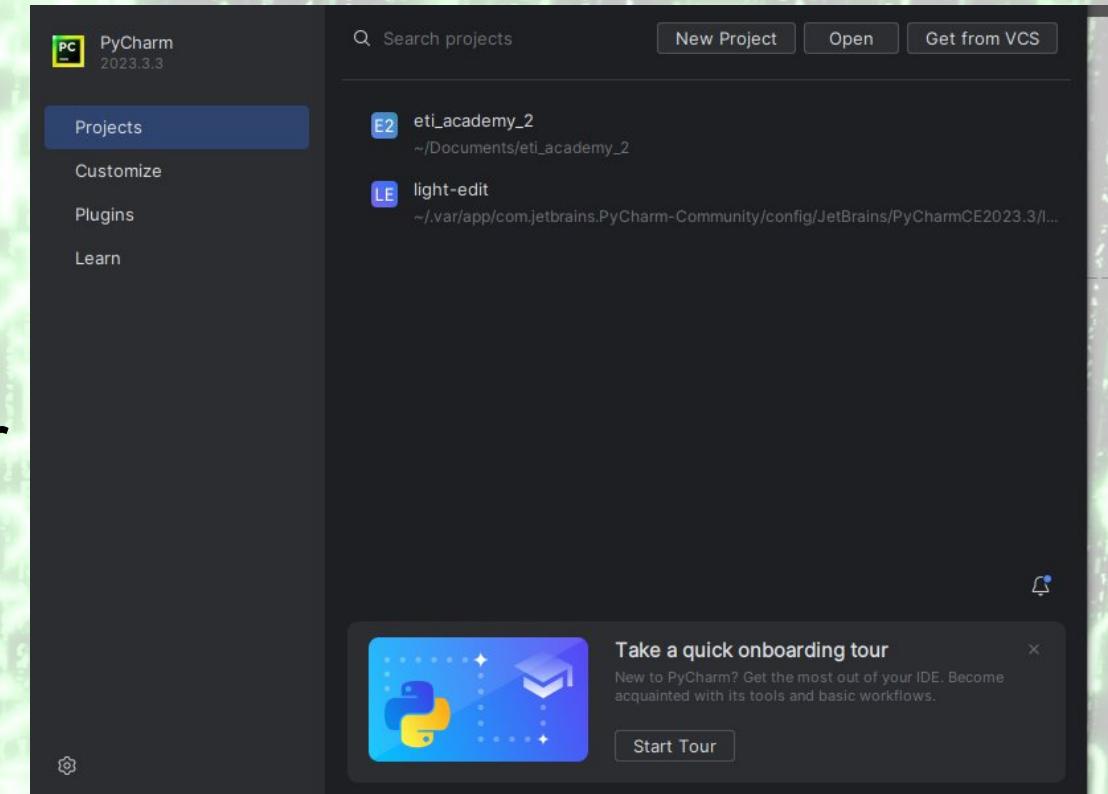
- Oczywiście, że nie:
 - C++ i Java mają stosunkowo skomplikowaną składnię
 - Pisząc kod w C++ musimy pamiętać o zarządzaniu pamięcią
 - Środowiska programistyczne Javy i C++ wymagają prawidłowej konfiguracji – jeśli „coś nam się kliknie”, to możemy mieć problem (szczególnie pracując na czas)
- Python jest pozbawiony powyższych wad
- Ponadto jego składnia ułatwia pracę na danych z plików – a na tym zwykle polegają zadania maturalne
- Łatwiejszy kod - oszczędność czasu

A co ze środowiskiem?

- Jeśli wybierzemy Pythona, mamy następujące opcje:
 - Idle
 - PyCharm Community Edition
 - (Oczywiście zawsze można działać w terminalu)
- Tutaj też opcje nie są równe:
 - Idle to tylko zaawansowany notatnik
 - PyCharm to pełnoprawne środowisko programistyczne z debuggerem, sprawdzaniem błędów, inspektorem zmiennych itd...
 - Do tego działają na każdym systemie operacyjnym
- Wybór jest oczywisty

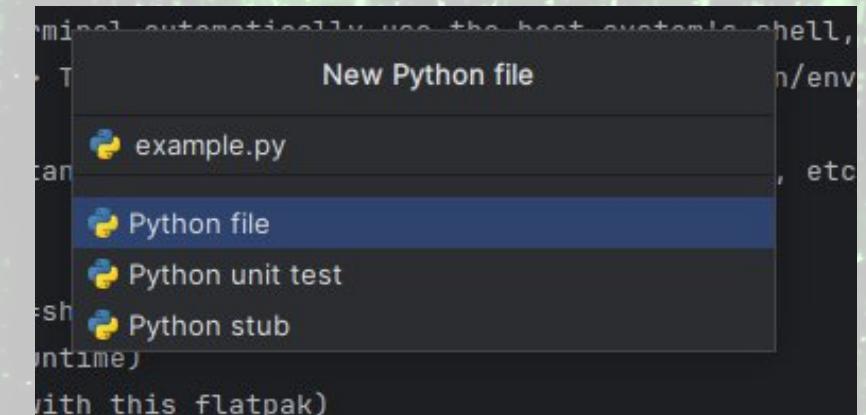
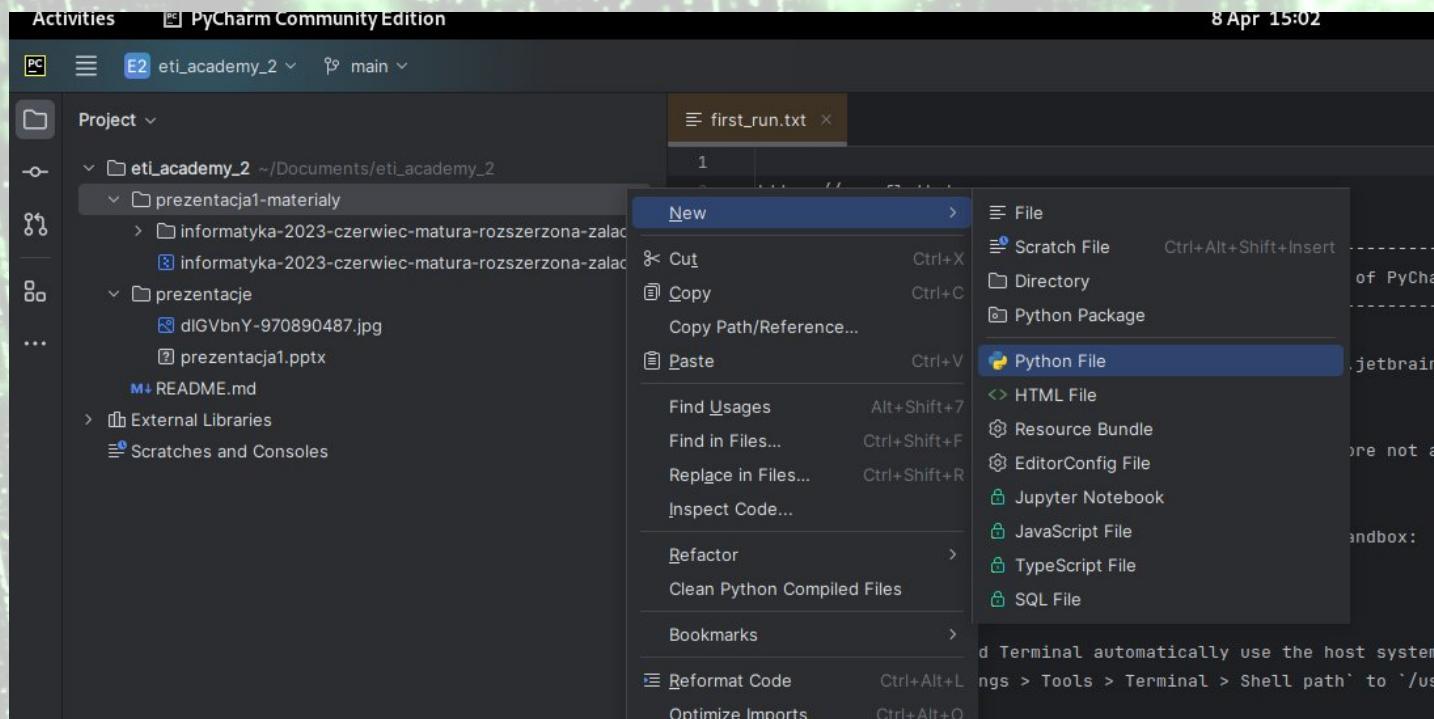
Przygotowanie środowiska pracy

- Stwórz nowy folder
- Uruchom PyCharm
- W zakładce „Projects” wybierz opcję „Open”
- Wybierz utworzony wcześniej folder



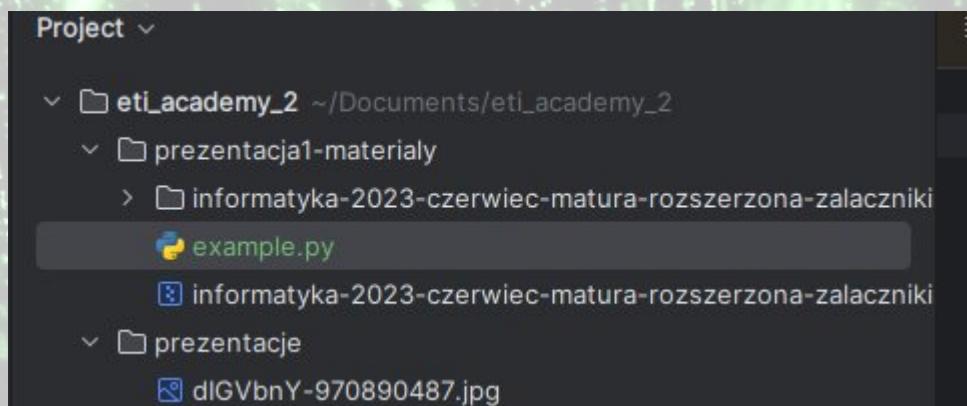
Utworzenie pliku

- Kliknij PPM na nazwę docelowego folderu
- Z rozwijanej listy wybierz New>Python File
- Pojawi się okienko - podaj tam nazwę nowego pliku

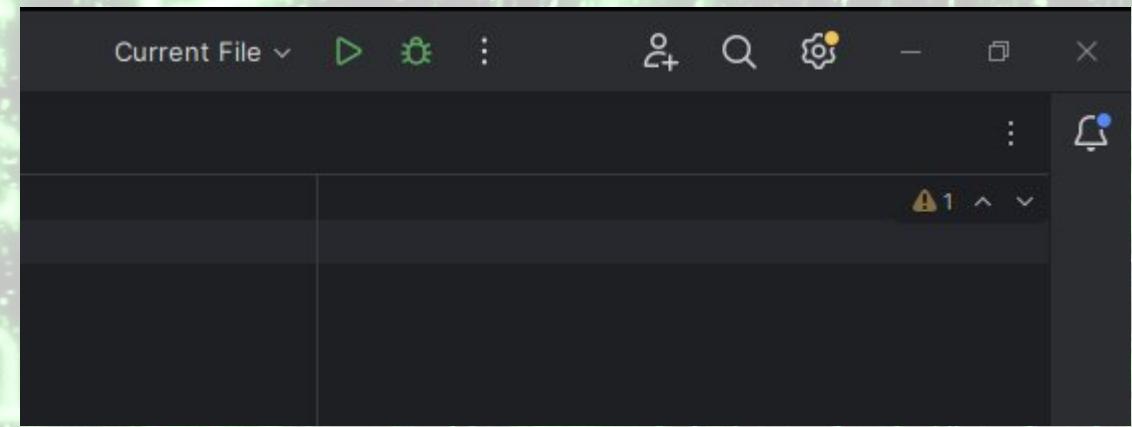


Uruchomienie

- Kliknij dwukrotnie na plik, który chcesz uruchomić. Powinien się on podświetlić, a jego treść wyświetli się w edytorze
- Kliknij na zielony przycisk „play” w prawym górnym rogu ekranu
- Wynik działania programu pojawi się w terminalu na dole



U mnie jest zielony, ponieważ dodałem ten plik do repozytorium git. U was nazwa będzie biała.

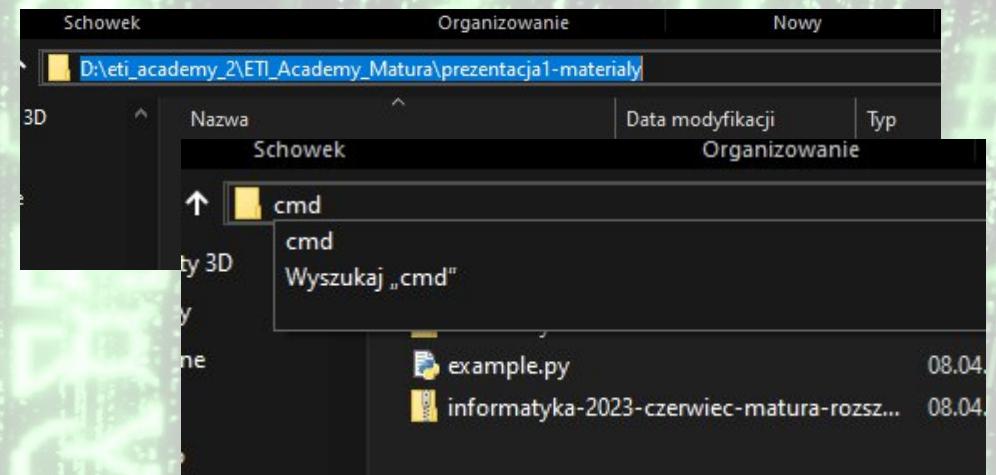


„Current file” oznacza, że odpalony zostanie plik, który jest obecnie otwarty

Opcja awaryjna - terminal

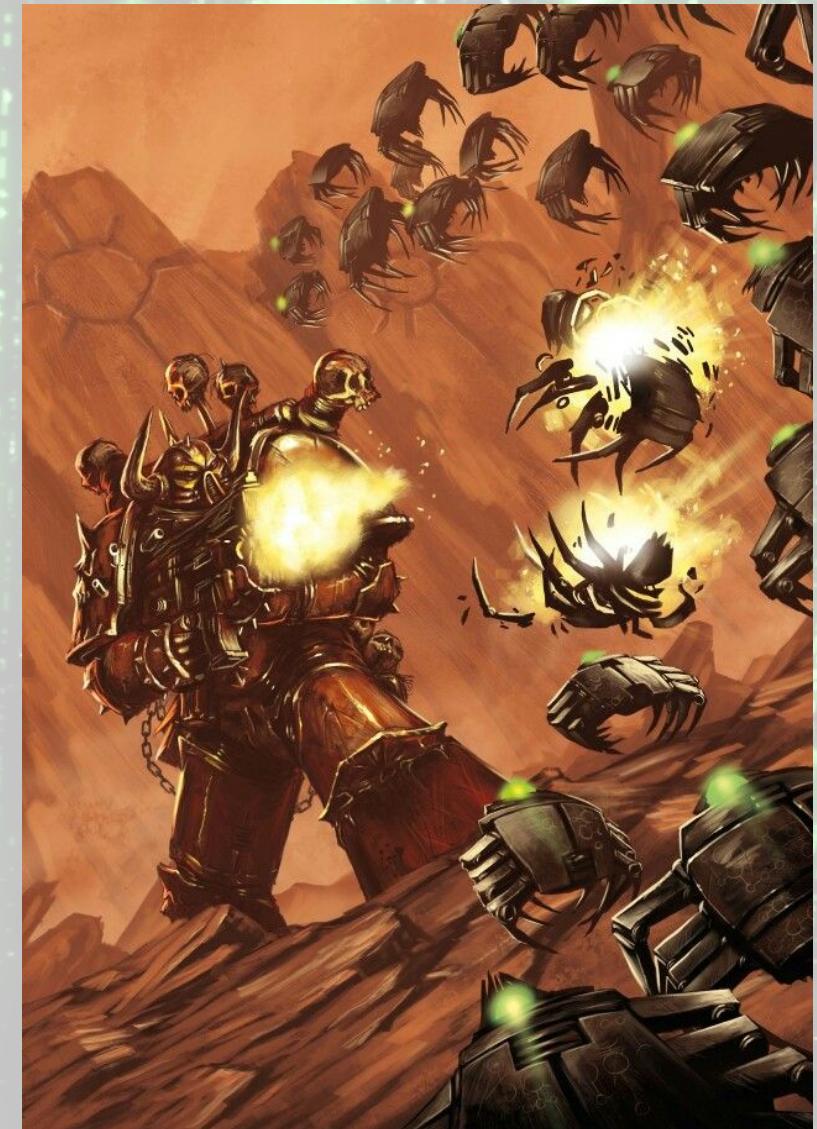
- Programy IDE są skomplikowane i łatwo można coś zepsuć
- Na maturze to byłoby niekorzystne
- W przypadku problemów z IDE, można uruchomić pythona z linii poleceń
- Komenda `python <nazwa_pliku>` wywołana w terminalu uruchamia dany plik
- Terminal musi być otwarty w folderze zawierającym plik .py
- Aby go otworzyć, w eksploratorze kliknij na pasek adresu i wpisz 'cmd'

```
D:\eti_academy_2\ETI_Academy_Matura\prezentacja1-materiały>python example.py
0
1
2
3
4
5
6
7
8
9
D:\eti_academy_2\ETI_Academy_Matura\prezentacja1-materiały>
```



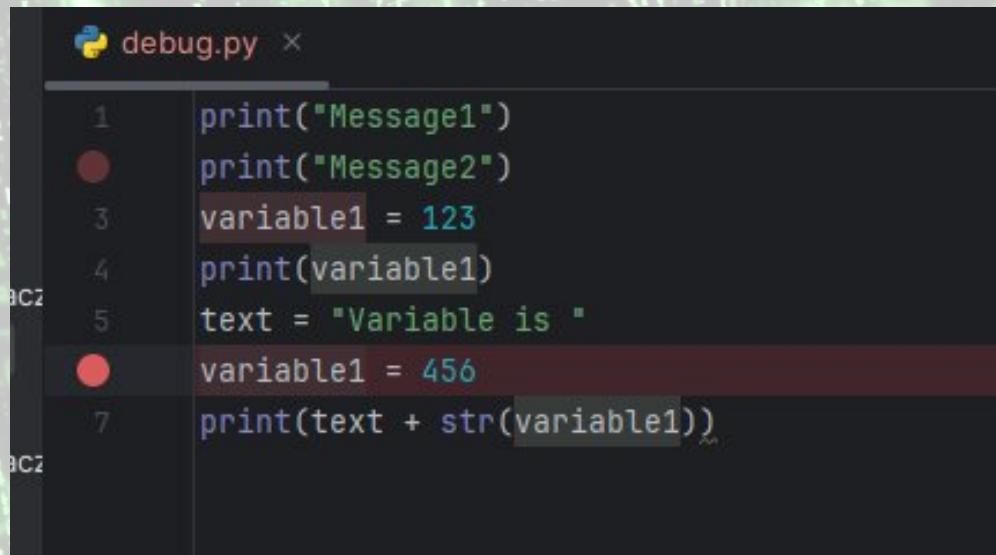
Debugger

- Wyobraźnia ludzka jest ograniczona – a programy działają bardzo szybko
- Z pomocą przychodzi nam debugger
- To narzędzie pozwala nam zatrzymać program w dowolnym momencie i zajrzeć do środka



Debugger - tutorial

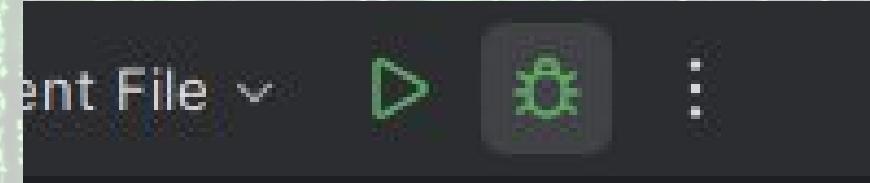
- Kliknięcie na numer linii pozwala dodać tam breakpoint
- Program zatrzyma się, kiedy osiągnie breakpoint'
- Program w trybie debug uruchamiamy, wciskając zielonego robaka w prawym górnym rogu



A screenshot of a Python debugger interface. The window title is "debug.py". The code editor shows the following Python script:

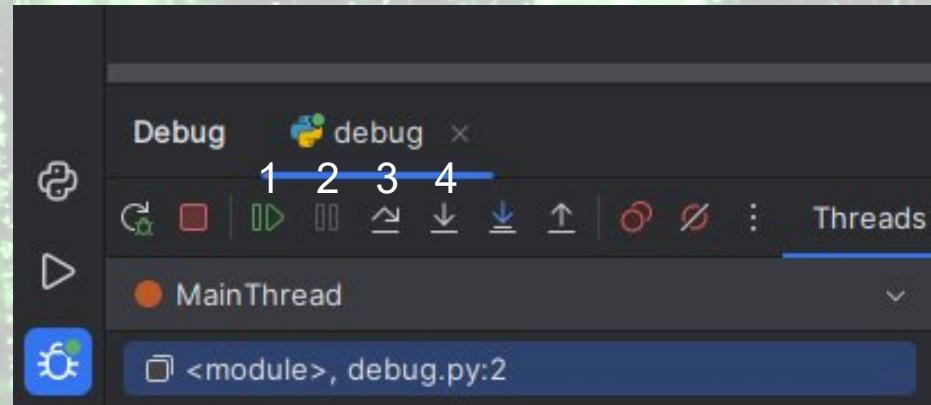
```
1 print("Message1")
2 ● print("Message2")
3 variable1 = 123
4 print(variable1)
5 text = "Variable is "
6 ● variable1 = 456
7 print(text + str(variable1))
```

The second and sixth lines have red circular markers at the start, indicating they are breakpoints. The code is syntax-highlighted in green and blue. Below the code editor is a dark toolbar with several icons: a file icon, a dropdown menu, a play button, a gear icon, and other symbols.



Debugger - tutorial

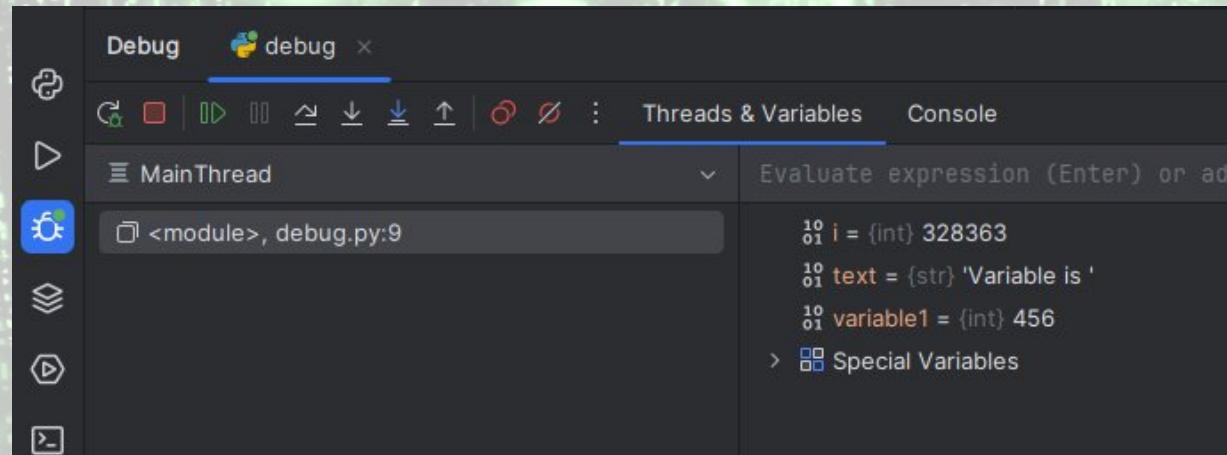
- Kiedy program się zatrzyma, na dole ekranu stanie się widoczne okienko debugera
- Można tam sterować przebiegiem programu:



- 1 - kontynuuj (do następnego breakpointa)
- 2 - zatrzymaj program (przydatne przy szczególnie długich instrukcjach)
- 3 - 'step over' - przejdź do kolejnej linii, ale nie wchodź do wnętrza funkcji
- 4 - 'step into' - przejdź do następnej linii

Debugger - tutorial

- Kiedy program jest zatrzymany, można podglądać aktualne wartości zmiennych
- To bardzo ważne – dzięki temu możemy wykryć błędy
- W okienku Debug są dwie zakładki:
 - **Console** - output programu
 - **Threads & Variables** - wartości i typy zmiennych



Operacje na plikach

- Zadania maturalne niemal zawsze sprowadzają się do pracy na danych z plików
- Python domyślnie wspiera odczyt i zapis plików – nie trzeba niczego importować
- Służy do tego funkcja `open(file, mode)`
- Jako argumenty podajemy ścieżkę do pliku oraz tryb, w którym otwieramy plik
- Funkcja zwraca plik jako obiekt
- Plik zamykamy metodą `file.close()`



Operacje na plikach - tryby

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open for updating (reading and writing)

Operacja na plikach – odczyt danych

- Aby odczytać zawartość pliku należy wywołać odpowiednie metody na obiekcie reprezentującym plik:
 - `file.read()` – zwraca całą treść pliku
 - `file.readlines()` – dzieli treść pliku na linie i zwraca listę
 - `file.readline()` – odczytuje jedną (kolejną) linijkę z pliku
- Można też zapisywać dane przy pomocy metody `file.write()`
- Miejsce, w którym odczytamy/zapiszemy dane zależy od pozycji „wirtualnego kurSORA” (a zatem od trybu w którym otwarto plik).

Źródła

<https://icon-icons.com/icon/open-file/40455>

www.jetbrains.com/help/pycharm

<https://docs.python.org/3/https://icon-icons.com/icon/open-file/40455>