

PODRÓŻE Z PYTHONEM – JAK UCZYMY PROGRAMOWANIA

Agnieszka Borowiecka, Katarzyna Olędzka
Ośrodek Edukacji Informatycznej i Zastosowań Komputerów
w Warszawie, ul Raszyńska 8/10
{agnieszka.borowiecka, katarzyna.oledzka}@oeiizk.edu.pl

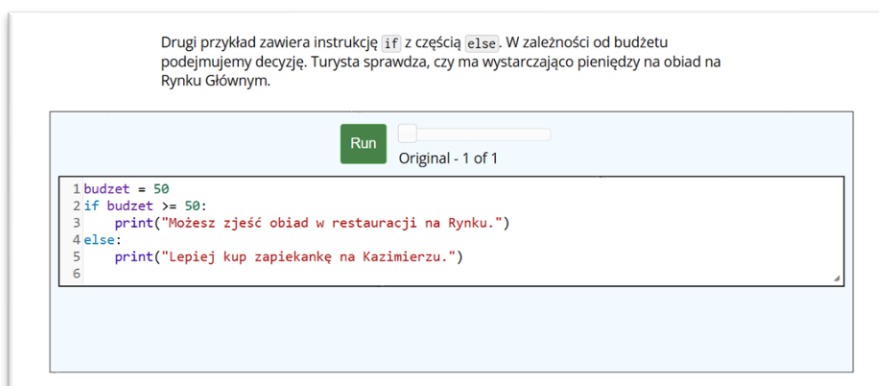
Abstract. *Learning programming shouldn't be about learning the language's syntax, but rather about solving problems. We will present a case study of the training course for computer science teachers. "Traveling with Python" combined learning the basics of Python with a travel narrative, practical tasks, and participants' own work. In subsequent stages, participants learned variables, functions, conditional statements, loops, and turtle graphics, solving problems embedded in the context of expeditions, route planning, calculations, and decision-making.*

1. Wprowadzenie

Nauczanie programowania pozostaje jednym z istotnych wyzwań edukacyjnych, także w czasach dynamicznego rozwoju sztucznej inteligencji. Choć narzędzia sztucznej inteligencji mogą wspierać pisanie programów, podpowiadać rozwiązania i przyspieszać pracę, nie zastępują programisty, który nie tylko rozumie problem, ale posiada umiejętności formułowania algorytmu i co ważne – krytycznej analizy otrzymanych wyników. Szkolenie „Podróże z Pythonem” powstało, by wspierać nauczycieli informatyki. Szkolenie nie ograniczało się do prezentowania składni języka Python, lecz prowadziło uczestników przez proces samodzielnego rozwiązywania problemów programistycznych. Ta forma doskonalenia została zaprojektowana z myślą o nauczycielach o różnicowanym poziomie doświadczenia: zarówno tych, którzy dopiero rozpoczynają pracę z programowaniem, jak i tych, którzy potrzebują uporządkowania wiedzy oraz praktycznych materiałów do wykorzystania z uczniami. Punktem wyjścia była obserwacja, że uczestnicy szkoleń często oczekują prostych, gotowych i szybko przyswajalnych rozwiązań, podczas gdy rzeczywiste kompetencje programistyczne rozwijają się dopiero poprzez własną aktywność, eksperymentowanie, popełnianie błędów i stopniowe doskonalenie zapisanego algorytmu. „Podróże z Pythonem” to angażujące szkolenie z podstaw programowania, w którym narracja podróżnicza, zadania praktyczne i praca własna uczestników tworzą spójną koncepcję dydaktyczną.

2. Koncepcja dydaktyczna

Koncepcja szkolenia „Podróże z Pythonem” została oparta na założeniu, że nauka programowania jest skuteczniejsza, gdy nowe pojęcia pojawiają się w rozpoznawalnym, spójnym i angażującym kontekście. Szkolenie było adresowane do nauczycieli informatyki prowadzących lub planujących prowadzić zajęcia z programowania i zostało przewidziane łącznie na 60 godzin pracy. Dokładniej, obejmowało 6 webinarów po 2 godziny oraz pracę asynchroniczną w sześciu blokach tematycznych. Każdy blok zakładał około 8 godzin pracy, w tym konsultacje. Poszczególne części szkolenia zostały powiązane z motywem podróży, który pełnił funkcję ramy narracyjnej dla zadań programistycznych. Uczestnicy rozpoczynali od pisania prostych skryptów z użyciem instrukcji wejścia i wyjścia, poprzez stosowanie zmiennych, a następnie przechodzili do obliczeń, funkcji, instrukcji warunkowych, pętli *for*, grafiki żółwia i pętli *while*. Każdy z tych tematów był osadzony w sytuacjach bliskich codziennemu doświadczeniu lub wyobraźni uczestników: wyprawie na zamek, zwiedzaniu Krakowa, pobycie w Krynicy Górskiej czy spacerze po Sandomierzu. Dzięki temu programowanie nie było przedstawiane jako zbiór abstrakcyjnych reguł składniowych, lecz jako narzędzie służące do rozwiązywania konkretnych problemów: obliczania kosztów, podejmowania decyzji, planowania trasy, zliczania zdarzeń czy wykonywania powtarzalnych czynności. Lekki kontekst podróżniczy obniżał barierę wejścia, porządkował materiał i pozwalał stopniowo budować kompetencje algorytmiczne. W realizacji szkolenia wykorzystano edytor Mu, Microsoft Teams oraz interaktywne materiały Pretextbook. Głównym celem było wsparcie nauczycieli w nauce programowania oraz rozwijanie umiejętności rozwiązywania prostych problemów algorytmicznych.



Rysunek 1. Przykład materiału interaktywnego

W szkoleniu spotkania synchroniczne wyprzedzały samodzielną pracę uczestników. Webinaria pełniły funkcję wprowadzenia do nowych zagadnień, przedstawienia składni

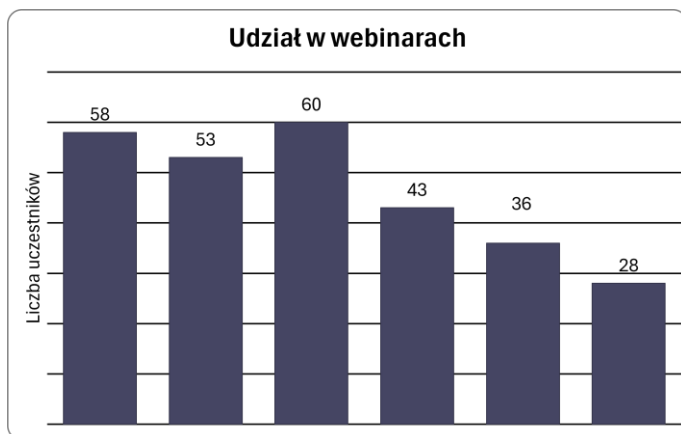
języka i rozwiązań przykładowych problemów oraz przestrzeni do omawiania typowych trudności pojawiających się podczas nauki programowania. Ważnym elementem było samodzielne rozwiązywanie zadań programistycznych i automatyczne ich sprawdzanie na platformie Moodle. Na platformie udostępniano materiały, zadania, quizy oraz aktywności pozwalające utrwalać kolejne zagadnienia. Zadania miały zróżnicowany charakter – od przykładów utrwalających nowe konstrukcje języka, po problemy wymagające samodzielnego zaplanowania rozwiązania. Każde zadanie programistyczne wymagało napisania krótkiego programu. Kluczowym założeniem dydaktycznym było przekonanie, że programowania nie można nauczyć się wyłącznie przez obserwowanie prowadzącego ani przez bierne odtwarzanie gotowych przykładów. Uczący się rozwija kompetencje dopiero wtedy, gdy samodzielnie pisze program, testuje go, poprawia błędy i stopniowo rozumie zależność między zapisem kodu, a działaniem algorytmu. Dlatego błędy traktowano nie jako porażkę, lecz jako naturalny i potrzebny element procesu uczenia się. W tym sensie szkolenie opierało się na zasadzie, że nie da się nauczyć programowania bez wysiłku uczącego się i samodzielnego rozwiązywania zadań.

Ważnym założeniem było takie projektowanie aktywności, aby uczestnicy mogli przechodzić zagadnienia krok po kroku. Zastosowano zasadę stopniowania trudności, skupiając się na rozwiązywaniu dużej liczby krótkich zadań. Istotne było również konsekwentne nawiązywanie do praktycznych sytuacji, dzięki czemu uczestnicy mogli zobaczyć, że programowanie nie jest oderwanym od rzeczywistości ćwiczeniem technicznym, ale narzędziem przydatnym w opisywaniu, porządkowaniu i rozwiązywaniu codziennych problemów. Takie połączenie prostoty, stopniowania trudności i praktycznego kontekstu sprzyjało budowaniu poczucia sprawczości oraz rozwijaniu myślenia algorytmicznego.

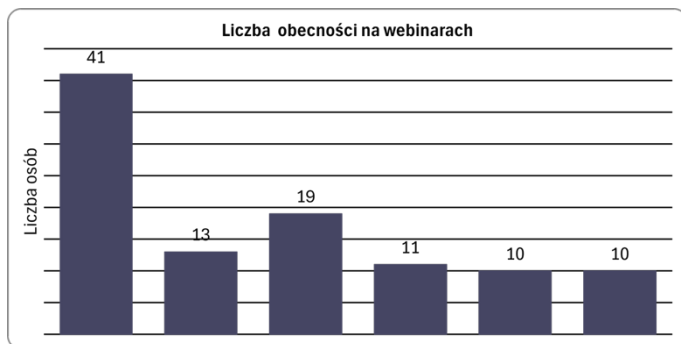
3. Realizacja szkolenia

Analizę przebiegu szkolenia przeprowadzono nie na podstawie deklaracji uczestników, lecz na zebranych danych o ich rzeczywistej aktywności. Pokazują one kilka wymiarów zaangażowania: obecność na webinarach, liczbę uczestników biorących udział w kolejnych spotkaniach, godziny pracy w środowisku Moodle oraz liczbę rozwiązanych zadań i quizów.

Dane dotyczące obecności wskazują, że zainteresowanie szkoleniem było wysokie na początku, jednak w kolejnych etapach widoczny był spadek udziału w spotkaniach synchronicznych.

**Rysunek 2.** Udział w webinarach

Wykres przedstawia obecność uczestników na sześciu kolejnych webinarach. W pierwszym spotkaniu uczestniczyło 58 osób, w drugim 53 osoby, a najwyższą frekwencję odnotowano podczas trzeciego webinarium – 60 osób. Od czwartego spotkania widoczny jest wyraźny spadek obecności: w czwartym webinarium uczestniczyły 43 osoby, w piątym 36 osób, a w szóstym 28 osób. Dane pokazują więc dwa trendy. Początkowo zainteresowanie było bardzo wysokie i utrzymywało się na podobnym poziomie, a nawet wzrosło w trzecim module, poświęconym instrukcjom warunkowym. W drugiej połowie cyklu nastąpiło stopniowe zmniejszanie liczby uczestników spotkań synchronicznych. Można to interpretować jako naturalne zjawisko w dłuższych szkoleniach online, w których wraz z upływem czasu maleje motywacja. Dodatkowo warto zauważyć, że kolejne spotkania – 4 i 6 dotyczące pętli były trudniejsze, a 5 – dotyczyło specyficznego aspektu programowania – grafiki żółwia.

**Rysunek 3.** Liczba obecności na spotkaniach

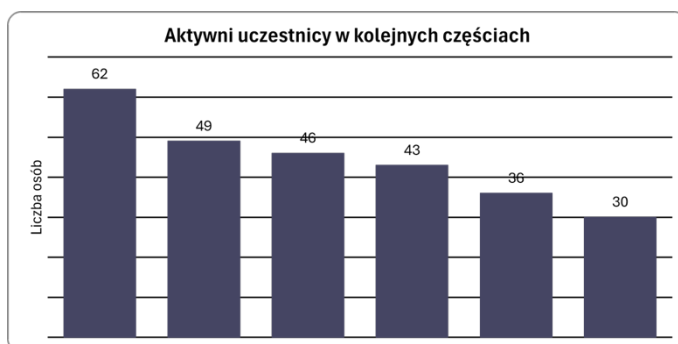
Warto zwrócić uwagę, w ilu spotkaniach uczestniczyły poszczególne osoby. Największą grupę stanowią uczestnicy, którzy wzięli udział tylko w jednym webinarium – było ich 41. W dwóch spotkaniach uczestniczyło 13 osób, w trzech 19 osób, w czterech 11 osób, a w pięciu i sześciu spotkaniach po 10 osób.

Dane wskazują na duże zróżnicowanie poziomu uczestnictwa w części synchronicznej kursu. Z jednej strony widoczna jest szeroka grupa osób, które pojawiły się jedynie okazjonalnie, prawdopodobnie korzystając z wybranego tematu lub sprawdzając formułę szkolenia. Z drugiej strony można wyróżnić stabilną grupę uczestników systematycznych: 20 osób brało udział w pięciu lub sześciu spotkaniach, a więc uczestniczyło w większości cyklu. W kontekście analizy szkolenia wykres pokazuje, że webinaria pełniły różne funkcje dla różnych uczestników. Dla części były podstawową formą udziału w szkoleniu, dla innych podstawą pracy własnej lub kontaktem z prowadzącym.

W części asynchronicznej brało łącznie udział 75 nauczycieli.

Tabela 1. Dane statystyczne szkolenia – wyniki

Wskaźnik	Wartość
Liczba uczestników	75
Maksymalna liczba punktów	600
Średni wynik	242
Najwyższy wynik	600
Osoby z wynikiem ≥ 300 pkt	30
Osoby z wynikiem ≥ 500 pkt	9
Osoby z pełnym wynikiem	1



Rysunek 4. Liczba osób, które rozwiązały przynajmniej jedno zadanie

Analiza liczby osób uzyskujących punkty w kolejnych częściach szkolenia pokazuje stopniowe zmniejszanie się aktywności uczestników. W pierwszym module dodatkia liczbę punktów otrzymały 62 osoby, natomiast w ostatnim 30 osób. Warto jednak dodać,

że część osób zaczęła później niż od pierwszego spotkania bądź z przyczyn organizacyjnych, bądź świadomie pomijając łatwiejsze części. Jednocześnie utrzymanie grupy 30 osób do szóstej części szkolenia można uznać za znaczące, ponieważ końcowe moduły obejmowały bardziej wymagające zagadnienia, takie jak pętla *while* i łączenie poznanych konstrukcji programistycznych.

Zniżka dla grupy

Jeden bilet do Muzeum Narodowego kosztuje 36 zł. Jeśli jest ich więcej niż 4, kupującym przysługuje 50% zniżki na cały zakup. Napisz program, który wczyta liczbę osób w grupie, a wypisze cenę, jaką trzeba zapłacić za bilety.

Wejście: liczba
Wyjście: liczba

Na przykład:

Dane wejściowe	Wynik
3	108
5	90
2	72

Rysunek 5. Przykład zadania programistycznego na Moodle, moduł CodeRunner

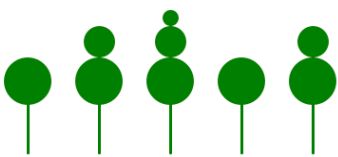
Tabela 2. Dane statystyczne szkolenia – quizy programistyczne

Aktywność	Liczba osób	Średnia punktów	Mediana
Instrukcje wejścia i wyjścia	61	17,3	20,0
Wejście jako liczba	51	17,1	20,0
Wzory ze znaków	47	16,5	20,0
Zadania tekstowe z obliczeniami	45	18,2	20,0
Dzielenie całkowite i reszta z dzielenia	41	16,4	20,0
Wzorce	36	16,2	20,0
Podstawy instrukcji warunkowej	43	17,1	20,0
Porządkowanie	35	16,6	20,0
Sprytne obliczenia	36	16,2	20,0
Pierwsze zadania z pętli	40	16,7	20,0
Pętle rozmaite	35	14,9	19,0
Iterujemy po wielu liczbach	30	16,9	20,0
Podstawy grafiki żółwia	36	18,3	19,4
Pierwsze rysunki	34	17,1	20,0
Funkcje graficzne	31	14,1	18,0
Odkryj pętlę <i>while</i>	24	15,3	20,0
Wędrówki po mieście – licz i programuj	17	14,6	18,0
While i funkcje – krok po kroku	17	13,6	20,0

Analiza wyników quizów programistycznych – pakietów od 8 do 10 krótkich zadań – pokazuje dwa równoległe zjawiska. Uczestnicy, którzy rozwiązywali quizy, często osiągnęli wysokie wyniki, o czym świadczy mediana równa 20 punktów w większości aktywności. Jest to efektem zastosowania metody małych kroków. Największe trudności pojawiały się w zadaniach wymagających łączenia konstrukcji programistycznych, zwłaszcza pętli, funkcji oraz elementów grafiki żółwia.

Oprócz zadań zamkniętych w każdej części były takie, które wymagały większej kreatywności i nie były ściśle wyspecyfikowane.

Tadeusz jest miłośnikiem formowanych choinek. Najbardziej podobają mu się drzewka wyglądające jakby były złożone z kilku kulek.



Przygotuj funkcję pozwalającą Tadeuszowi zaprojektować jego ogród. Choinki mają być sadzone w szpalerze, jak na rysunku powyżej. Wysokość pnia choinki jest taka sama jak średnica najniższej kuli, choinka może składać się maksymalnie z 3 kul, każda kolejna jest wtedy o $\frac{1}{3}$ mniejsza (średnice a , $\frac{2}{3}a$ i $\frac{1}{3}a$). Ogród jest nieduży, więc średnica największej kuli nie przekracza 50, a odstęp między sąsiednimi drzewkami do $\frac{3}{2}a$.

Napisz następujące funkcje:

- **choinka(a)** - rysuje choinkę składającą się z jednej kuli o wysokości pnia a
- **drzewko(a, n)** - rysuje choinkę składającą się z 1, 2 lub 3 kul o wysokości pnia a
- **szpaler(a, ile)** - rysuje szpaler złożony z *ile* choinek o wysokości pnia a . Szpaler może składać się z takich samych choinek, lub z dowolnego układu choinek różnej wysokości, np. takiego jak na rysunku powyżej.

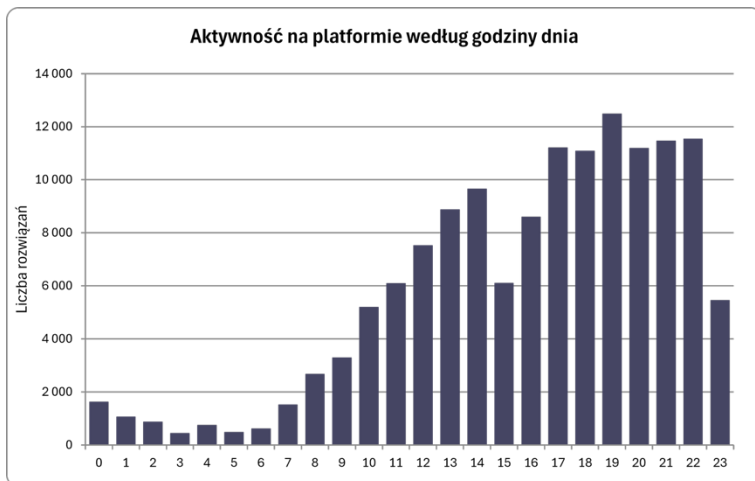
Rysunek 6. Przykład zadania otwartego *Ogród Tadeusza*

Tabela 3. Dane statystyczne szkolenia – zadania otwarte

Część	Zadanie	Liczba ocenionych rozwiązań
Część 1	Historia podróżnicza	40
Część 2	Kolejka górską na zamek	30
Część 3	Gra tekstowa	19
Część 4	Posadzka	27
Część 5	Ogród Tadeusza	18
Część 6	Sandomierski pierścień	11

Zadania otwarte były aktywnościami wymagającymi samodzielności. Uczestnicy musieli nie tylko zastosować poznaną konstrukcję języka, lecz także zaplanować rozwiązanie i podjąć decyzje dotyczące sposobu zapisu programu. Z tego powodu liczba oddanych zadań była niższa niż liczba rozwiązanych quizów, ale aktywności te stanowiły ważny wskaźnik rzeczywistej pracy programistycznej.

Zestawienie godzin pracy na platformie Moodle można potraktować jako ciekawą obserwację dotyczącą rytmu uczenia się uczestników w środowisku online. Pokazuje ono, że aktywność na platformie nie rozkładała się równomiernie w ciągu dnia. Najmniej działań odnotowano w godzinach nocnych i wczesnoporannych, natomiast wzrost zaczyna się około godziny 7.00. Aktywność utrzymuje się na wysokim poziomie w godzinach popołudniowych i wieczornych, a najwyższe wartości pojawiają się około godziny 19.00. Wysoka liczba działań widoczna jest także między 17.00 a 22.00.



Rysunek 7. Godziny pracy na platformie Moodle

Taki rozkład jest charakterystyczny dla szkolenia kierowanego do nauczycieli. Uczestnicy prawdopodobnie realizowali zadania po zakończeniu własnych obowiązków w szkole, dlatego największa aktywność przypadła na późne popołudnie i wieczór. W kontekście projektowania szkoleń online jest to ważna wskazówka organizacyjna: materiały, zadania i terminy powinny uwzględniać fakt, że praca własna nauczycieli często odbywa się poza standardowymi godzinami pracy szkoły.

Warto też zauważyć, że wykres dobrze pokazuje znaczenie asynchronicznej formy szkolenia. Gdyby szkolenie opierało się wyłącznie na spotkaniach na żywo, część tej aktywności mogłaby się nie wydarzyć. Platforma Moodle umożliwia uczestnikom pracę w dogodnym dla nich czasie, a więc dostosowanie tempa nauki do realnego rytmu dnia. To szczególnie istotne w kursach programowania, ponieważ samodzielne pisanie programów wymaga czasu, skupienia i możliwości powracania do materiałów.

Należy podkreślić, że przedstawione dane mają charakter opisowy i dotyczą aktywności uczestników w środowisku Moodle oraz udziału w spotkaniach online. Nie stanowią pełnego pomiaru przyrostu kompetencji programistycznych, ponieważ nie obejmują np. badania przed i po szkoleniu, ani jakościowej analizy wszystkich rozwiązań.

Pozwalają jednak uchwycić dynamikę uczestnictwa, poziom zaangażowania oraz obszary, które okazały się dla uczestników bardziej wymagające.

4. Podsumowanie

Opisane doświadczenie może być wykorzystane zarówno przy projektowaniu innych form doskonalenia nauczycieli, jak i przy planowaniu zajęć z programowania dla uczniów. W obu przypadkach istotne jest, aby celem nie było jedynie poznanie narzędzia lub składni języka, lecz rozwijanie myślenia algorytmicznego, samodzielności w rozwiązywaniu problemów oraz gotowości do testowania, poprawiania i doskonalenia własnych rozwiązań. Szczególnie wartościowe wydaje się przeniesienie do pracy z uczniami zasady małych kroków, osadzania zadań w bliskim i zrozumiałym kontekście oraz stopniowego przechodzenia od prostych ćwiczeń do bardziej otwartych problemów programistycznych.

Studium przypadku szkolenia „Podróże z Pythonem” pokazuje, że skuteczne nauczanie podstaw programowania wymaga połączenia kilku elementów: spójnej narracji, stopniowania trudności, dużej liczby krótkich zadań oraz przestrzeni do samodzielnej pracy. Motyw podróży pełnił funkcję nie tylko atrakcyjnej oprawy, lecz także ramy porządkującej kolejne problemy programistyczne i ułatwiającej przejście od składni języka do myślenia algorytmicznego. Dane z Moodle wskazują, że aktywność uczestników malała w kolejnych częściach szkolenia, co jest typowe dla dłuższych form online, jednak spora grupa systematycznie pracujących nauczycieli utrzymała się do końca szkolenia. Szczególnie wartościowe okazały się zadania wymagające samodzielnego napisania programu, ponieważ to one najlepiej realizowały założenie, że programowania nie można nauczyć się wyłącznie przez obserwację, lecz przez własne działanie, testowanie i poprawianie błędów.

Literatura

1. Borowiecka A., Olędzka K., *PRIMM: nowy sposób na stare problemy z nauką programowania*, IWE 2025
2. Kurpiewski D., *Materiały edukacyjne z informatyki – jak uczyć złożonych zagadnień w najprostszy sposób?*, IWE 2025
3. Olędzka K., *Metoda PRIMM – nauka programowania krok po kroku*, kwartalnik W cyfrowej szkole 1/2025, OEliZK, Warszawa 2025
4. *Python w szkole*, <https://python.oeiizk.waw.pl>, ostatni dostęp 10.06.2026 roku
5. Sysło M.M., *Informatyka – fundamenty wdrażania*, IWE 2019
6. Sysło M.M., *Informatyka z komputerem w tle (unplugged revisited)*, kwartalnik W cyfrowej szkole 3/2023, OEliZK, Warszawa 2023