Od programowania wizualnego do tekstowego[1]

Maciej BOROWIECKI, Krzysztof CHECHŁACZ

Wstęp

Nowa podstawa programowa przedmiotu informatyka kładzie duży nacisk na rozwiązywanie problemów z pomocą komputera oraz programowanie. Na wczesnych etapach edukacyjnych zwykle rozpoczynamy od programowania wizualnego. Podczas tworzenia projektu unikamy w ten sposób problemów związanych z zawiłościami składni języka programowania. Gdy próbujemy połączyć elementy, które do siebie nie pasują, mechanizm bloczków nie pozwala nam na to. Ponadto szybko uzyskujemy efekt w postaci działającego projektu i w ten sposób zachęcamy uczniów do kolejnych aktywności.

W pewnym momencie należy jednak przejść na programowanie tekstowe. Języki tekstowe mają zwykle większe możliwości i choćby dlatego warto zainteresować się programowaniem w środowisku tekstowym. Trudno też będzie podtrzymać zainteresowanie uczniów przez 12 lat, pozostając wyłącznie przy programowaniu wizualnym. Komentarz do nowej podstawy programowej wskazuje jako dobry moment na rozpoczęcie nauki w środowisku tekstowym VII-VIII klasę zreformowanej szkoły podstawowej. Uczniowie w tym wieku zaczynają myśleć także abstrakcyjnie.

Pisząc program w środowisku tekstowym, musimy znać składnię języka i sprawnie posługiwać się klawiaturą. Są to podstawowe problemy, na jakie napotykają osoby rozpoczynające naukę programowania tekstowego. Pojawiają się błędy składniowe. Wielu uczniów zniechęca się i wyraża chęć powrotu do programowania wizualnego. W dalszej części artykułu przedstawiamy propozycje przejścia od programowania wizualnego do tekstowego w sposób łagodny i akceptowalny dla uczniów. Co więcej, w ten sposób pokazujemy, że najważniejszy jest sam algorytm rozwiązania zadania, a sposób zapisu jest mniej istotny.

Wykorzystamy narzędzia, które umożliwiają napisanie programu w pewnym języku, a następnie automatyczne przełożenie kodu na inny język.

Algorytm Euklidesa

W nowej podstawie programowej znajdujemy wiele odwołań do różnych algorytmów. W szczególności uczeń na poziomie klas VII-VIII stosuje przy rozwiązywaniu problemów podstawowe algorytmy na liczbach całkowitych: bada podzielność liczb, wyodrębnia cyfry danej liczby, przedstawia działanie algorytmu Euklidesa w obu wersjach iteracyjnych (z odejmowaniem i z resztą z dzielenia). Zajmijmy się przykładowo algorytmem Euklidesa. Umożliwia on wyliczenie największego wspólnego dzielnika dwóch liczb naturalnych. W najprostszej wersji polega on na tym, że póki liczby są różne, od większej z nich odejmujemy mniejszą. Gdy są równe, oznacza to, że otrzymaliśmy wynik.

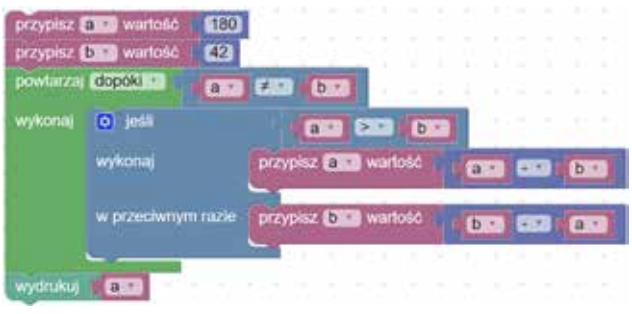
Rozwiązanie zapiszmy w języku programowania. Na razie nie zastanawiamy się nad kwestią

[1] Artykuł został opracowany na podstawie warsztatów przygotowanych przez autorów na konferencję Informatyka w Edukacji 2017 w Toruniu oraz artykułu „Od programowania wizualnego do tekstowego" [1].

eleganckiego przekazania danych do programu, wrócimy do tego później. Przykładowe rozwiązanie dla liczb 180 i 42 w najbardziej popularnym w Polsce środowisku programowania wizualnego Scratch może wyglądać następująco:

Rysunek 1. Algorytm Euklidesa w Scratchu.

Podobnie będzie ono wyglądać w środowisku *Blockly* dostępnym na stronie `https://blockly-demo.appspot.com/static/demos/code/index.html`. Środowisko to jest jednym z demonstracyjnych projektów inicjatywy *Google Blockly*, w którym powstała m.in. szeroko znana *Godzina Kodowania*. Podobnie jak w Scratchu możemy używać języka polskiego, mamy elementy – bloczki, które możemy łączyć, tworząc własny projekt, a następnie taki projekt zapisać i uruchomić.



Rysunek 2. Algorytm Euklidesa w Google Blockly.

W obu przypadkach otrzymujemy wynik 6. Jest on poprawny – w istocie dzielnikami 180 są 1, 2, 3, 4, 5, 6, 9, 10, 12, 15, 18, 20, 30, 36, 45, 60, 90 oraz 180, dzielnikami 42 są 1, 2, 3, 6, 7, 14, 21 oraz 42 – liczba 6 jest największą, która występuje w obu przypadkach.

W *Blockly* mamy możliwość obejrzenia, jak wyglądałoby rozwiązanie zapisane w kilku innych językach. Wystarczy wybrać jedną z zakładek.

Rysunek 3. Wybór języka docelowego.

Kilka ostatnich zakładek daje możliwość obejrzenia kodu w dość rzadko używanych językach lub kod jest mało czytelny. Zajmiemy się językami z dwóch pierwszych zakładek: *Python* i *JavaScript*.

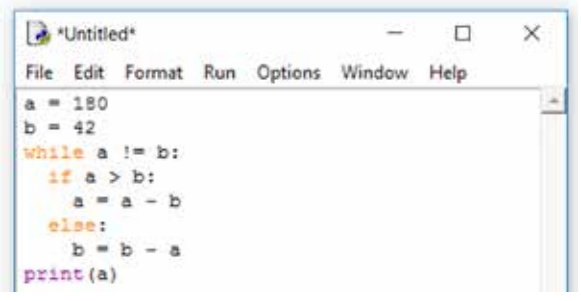
Python

Kod w języku *Python* jest krótki i czytelny, nie pozostawia wątpliwości co do sposobu działania; poniżej z pominiętymi dwoma pierwszymi wierszami.

```
a = 180
b = 42
while a != b:
  if a > b:
    a = a - b
  else:
    b = b - a
print(a)
```

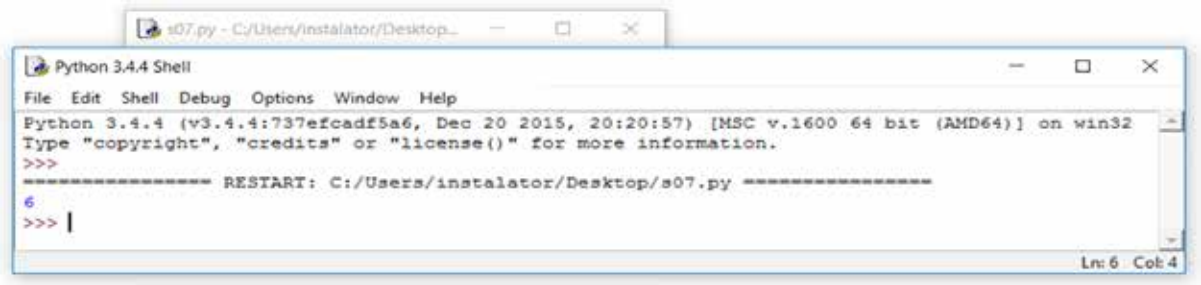
Rysunek 4. Kod w Pythonie wygenerowany automatycznie.

Tak uzyskany kod możemy skopiować do środowiska języka *Python*, najlepiej do nowego (pustego) okna.

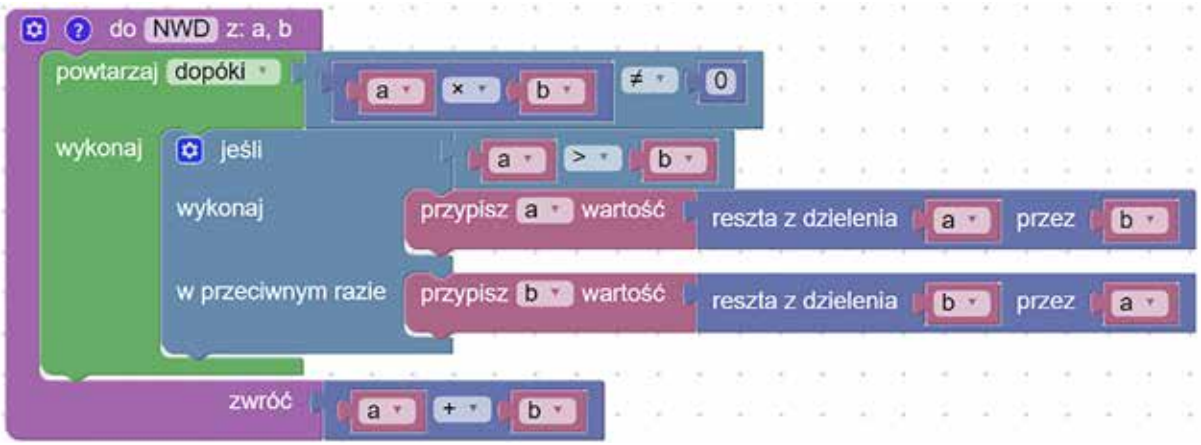


Rysunek 5. Kod źródłowy w Pythonie.

Od razu widać, jak elegancko środowisko *Pythona* koloruje składnię – dzięki temu wiemy, że słowa kluczowe wpisane zostały poprawnie. Po uruchomieniu programu wynik zostanie wypisany w głównym oknie.



Rysunek 6. Po uruchomieniu programu w Pythonie.



Rysunek 7. Kod funkcji w Google Blockly.

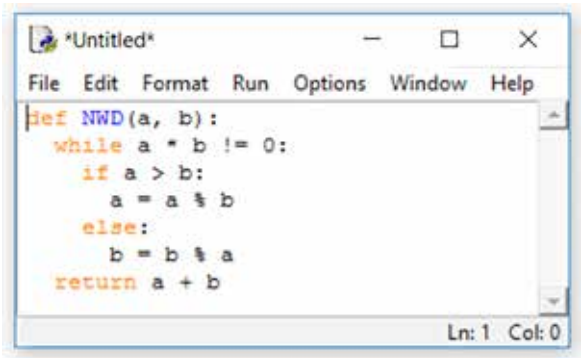
Tak więc, bez znajomości *Pythona*, udało się nam „napisać" i uruchomić program w tym języku. W sposób oczywisty, bez większej trudności, uczniowie poznali składnię najważniejszych instrukcji – przypisania wartości, warunkowej, pętli, a także zobaczyli sposób tworzenia wyrażeń i warunków logicznych. Warto w tym miejscu przeprowadzić dyskusję na temat znaczenia wcięć stosowanych w kodzie programu – *Python* wymusza ich stosowanie. Jest to sytuacja korzystna i nie rodzi dyskusji, bo to nie nauczyciel poleca stosowanie wcięć, ale jest to formalny wymóg składni języka. Uczniowie uczą się na podstawie przykładu, dzięki czemu ich wiedza będzie trwalsza i użyteczniejsza.

W dalszej części zajęć z uczniami można rozbudowywać program. Na przykład korzystając z *Google Blockly*, budujemy funkcję znajdującą największy wspólny dzielnik, a sam algorytm Euklidesa modyfikujemy tak, by zamiast odejmowania używał reszty z dzielenia (rysunek 7). Budowanie funkcji jest możliwe w *Google Blockly*, natomiast w *Scratchu* nie ma możliwości zwrotu wartości funkcji.

Kod otrzymany w *Pythonie* będzie zawierał nowe informacje – o tym, jak wygląda struktura funkcji i jak zapisywać resztę z dzielenia.

Wywołanie funkcji wymaga użycia nazewnika funkcji i podania parametrów wywołania. Efekt jest identyczny, jak poprzednio.

Warto dać uczniom do wykonania inne zadanie polegające na zapisie pewnego algorytmu w wybranym przez siebie języku, np. algorytmu wyznaczenia dzielników zadanej liczby naturalnej bądź wyodrębnienia cyfr danej liczby.

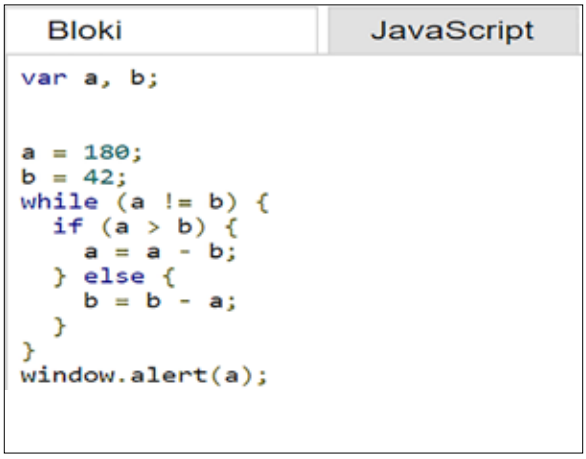


Rysunek 8. Kod funkcji automatycznie wygenerowany i przeniesiony do Pythona.

Rysunek 9. Wywołanie funkcji.

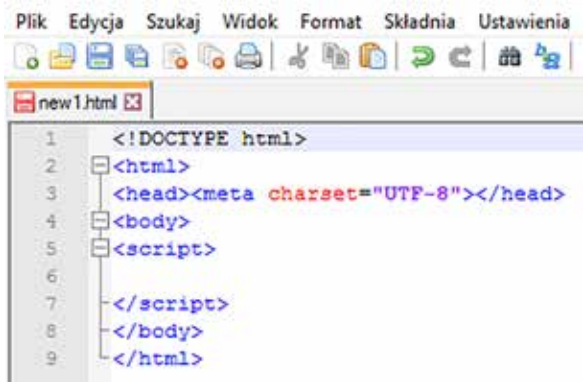
JavaScript

Spróbujmy teraz skorzystać z kodu *JavaScript*, który został wygenerowany z użyciem *Google Blockly* dla algorytmu Euklidesa:



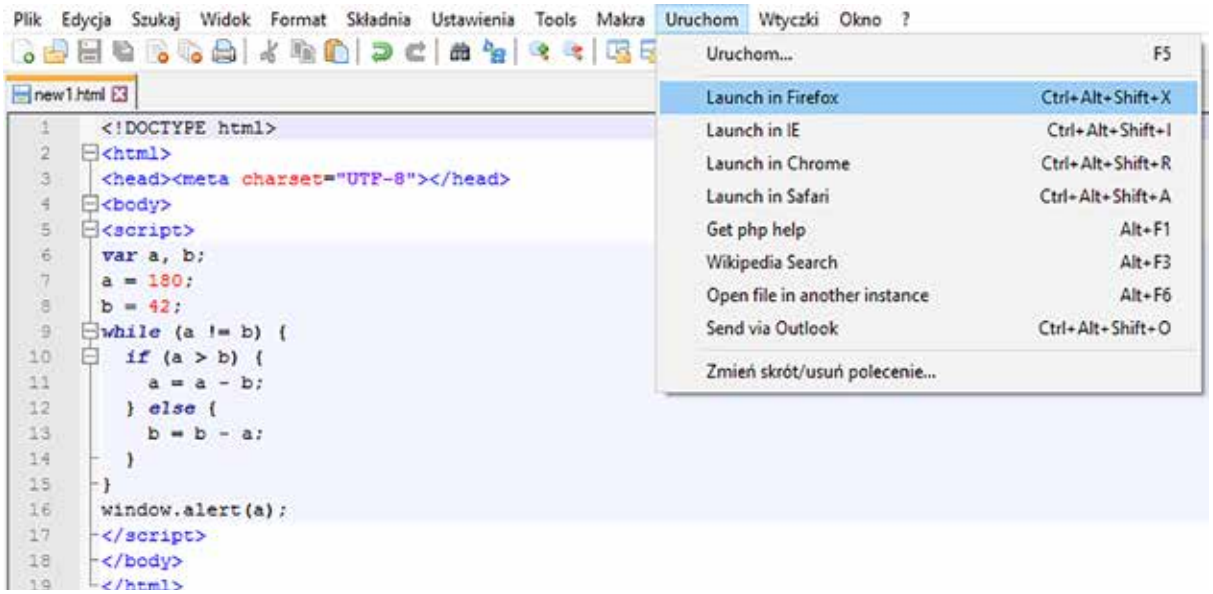
Rysunek 10. Kod w JavaScript wygenerowany automatycznie.

Najprościej będzie kod *JavaScript* zanurzyć w dokumencie *html*, a następnie tak utworzony



Rysunek 11. Szablon opisu strony internetowej.

dokument przekazać przeglądarce internetowej do zinterpretowania. Dla utworzenia dokumentu html możemy użyć programu *Notatnik++* z możliwością bezpośredniego przejścia – uruchomienia kodu *html* w przeglądarce internetowej. Można też użyć systemowego notatnika, ale wówczas niektóre czynności będą bardziej złożone.

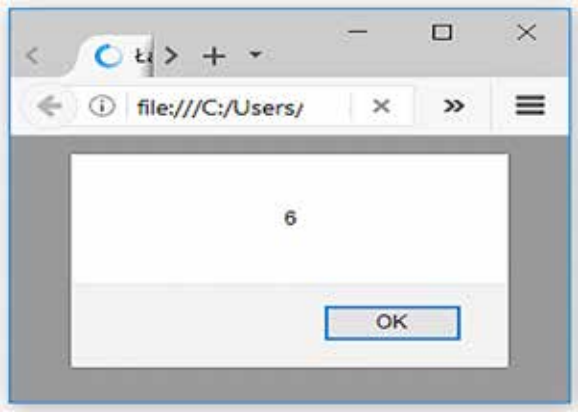


Rysunek 12. Opis strony internetowej po umieszczeniu w nim kodu

W pierwszym kroku tworzymy szablon opisu strony internetowej. Utworzony skrypt będzie działać także wówczas, gdy ograniczymy się do piątego i siódmego wiersza, ale nie polecamy takiego rozwiązania, ponieważ tworzy to złe nawyki dotyczące budowy plików opisujących strony internetowe.

W miejscu pustego szóstego wiersza umieszczamy kod z *Google Blockly*. Dokument zapisujemy (to ważne, bo program edycyjny komunikuje się z przeglądarką, używając wersji pliku *html* zapisanej na dysku) i przekazujemy przeglądarce do zinterpretowania, wybierając *Uruchom* i *Launch in* (tu-nazwa- przeglądarki).

Przeglądarka zinterpretuje kod i wypisze wynik.



Rysunek 13. Wynik działania JavaScript.

Podsumowanie

Przedstawiliśmy rozwiązanie dotyczące automatycznej zamiany kodu zapisanego w środowisku programowania wizualnego na kod w języku programowania tekstowego. Pokazuje ono, że dość łatwo można zachęcić uczniów, nawet tych, którzy przez długi czas programowali wizualnie, do podjęcia trudu tworzenia własnych programów z użyciem narzędzi programowania tekstowego. Koncentrujemy się na rozwiązywanym problemie, algorytmie. Składnia języka pozostaje w tle, wprowadzamy tylko te instrukcje, które są potrzebne.

Bibliografia

1. Borowiecki M., Chechłacz K. *Od programowania wizualnego do tekstowego*, Informatyka w Edukacji, Toruń 2017.
2. Borowiecki M. *Python na lekcjach informatyki w szkole ponadgimnazjalnej*, Informatyka w Edukacji, Toruń 2013.
3. Borowiecki M. *Python w szkole od podstawówki do liceum*, EduFakty – Uczę Nowocześnie nr 23, styczeń-luty 2013.
4. Podstawa programowa kształcenia ogólnego dla szkoły podstawowej, http://www.dziennikustaw.gov.pl/DU/2017/356, dostęp 3.09.2017.
5. Polewczyński A. *Nauka kodowania z Google Blockly*, Informatyka w Edukacji, Toruń 2014.
6. Zając K. *Python jako alternatywa dla Pascala*, Informatyka w Edukacji, Toruń 2012.
7. https://scratch.mit.edu, dostęp 3.09.2017.
8. https://blockly-demo.appspot.com/static/demos/code/index.html?lang=pl, dostęp 3.09.2017.
9. https://www.python.org, dostęp 3.09.2017.

Maciej BOROWIECKI jest wicedyrektorem ds. edukacyjnych, **Krzysztof CHECHŁACZ** jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.