Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

INFORMATYKA

Lekcja pokazowa/otwarta – SP

Scenariusz lekcji

Temat:
Przejście od Godziny Kodowania do Pythona

Maciej M. Sysło
syslo@ii.uni.wroc.pl
http://mmsyslo.pl

Wrocław miasto spotkań
DOLNY ŚLĄSK
Dolnośląska Biblioteka Pedagogiczna we Wrocławiu
Uniwersytet Wrocławski
GMINA DŁUGOŁĘKA
Wrocławskie Centrum Doskonalenia Nauczycieli
Dolnośląska Szkoła Ćwiczeń

1. Obszar informatyki i klasa, w której będą prowadzone zajęcia

Obszar informatyki: Algorytmika i programowanie – szkoła podstawowa

Klasa: wybór klasy zależy od nauczyciela, który powinien wziąć pod uwagę, czy uczniowie są przygotowani do zajęć na zaproponowany temat.

Przygotowanie uczniów do lekcji: Uczniowie powinni mieć wcześniej kontakt z programowaniem wizualno-blokowym oraz znać środowisko programowania w języku Python. W szczególności powinni umieć napisać prosty program w tym środowisku z wykorzystaniem modułu `turtle`.

Kontakt uczniów ze środowiskiem wizualno-blokowym mógł mieć miejsce podczas zajęć w środowisku Scratcha lub przy programowaniu robotów. Na tych zajęciach uczniowie będą jednak korzystali z łamigłówek **Godziny Kodowania** – code.org – zatem wcześniej powinni mieć styczność z tą inicjatywą, która zawiera wiele zestawów łamigłówek tematycznych i tworzących całe zintegrowane kursy. Łamigłówki tematyczne są bardzo ciekawe dla uczniów i inspirujące do pracy, gdyż występują w nich postacie znane uczniom z gier i zabaw rozgrywanych na ogół poza komputerem. Kursy z kolei, przewidziane na dany poziom kształcenia, są kolekcją zestawów łamigłówek, a poszczególne zestawy odpowiadają wybranym pojęciom i konstrukcjom programistycznym. Przed tymi zajęciami uczniowie powinni spędzić przynajmniej jedną lekcję, na której wykonają z powodzeniem pewną liczbę łamigłówek.

Głównym celem tych zajęć jest łagodne przeprowadzenie uczniów ze środowiska programowania wizualno-blokowego do środowiska programowania tekstowego.

Terminem **programowanie tekstowe** określa się programowanie w środowiskach, w których program jest pisany, ma postać tekstu. W szkole uczniowie spotykają się najpierw z **programowaniem wizualno-blokowym**, w którym konstrukcje programistyczne (instrukcje) są tworzone z gotowych bloków, jednego lub wielu. Ponadto programowane są na ogół postacie, zwierzęta, inne istoty lub obiekty. Ten rodzaj programowania ma tę zaletę, że kieruje uwagę ucznia na układanie bloków w celu osiągnięcia określonego celu – interaktywnej historyjki, nie zajmując jego uwagi stroną składniową programu – często tylko niektóre bloki pasują do siebie.

W blokach w środowisku programowania wizualno-blokowego na ogół znajdują się teksty, bliskie znaczeniom tych bloków w językach tekstowych. Te zajęcia mają na celu zwrócenie uwagi uczniów na te powiązania, które mogą się przyczynić do spiralnego rozwoju ich umiejętności programowania i myślenia komputacyjnego.

Proponowana lekcja może być również wkładem do kształcenia matematycznego w zakresie wyobraźni geometrycznej, wspartej odpowiednimi obliczeniami kątów i odległości, związanymi z figurami geometrycznymi i ich aranżacją. Pojawić się może również rekurencja w realizacji wybranych konstrukcji.

2. Zagadnienia metodyczne, stanowiące podstawę przygotowania lekcji w zakresie rozwijania kompetencji metodycznych

Metodyczną podstawę realizacji lekcji na zaplanowany temat, a faktycznie każdej lekcji realizującej obowiązującą podstawę programową informatyki, stanowią fundamenty kształcenia informatycznego w polskich szkołach, wynikające z konstrukcji podstawy programowej, miejsca i roli programowania, powiązań informatyki z innymi dziedzinami oraz metodyki kształcenia. Tymi fundamentami są:

1. **kolejność** celów ogólnych kształcenia – kształcenie w zakresie abstrakcyjnego i algorytmicznego myślenia powinno poprzedzać naukę programowania i korzystanie z aplikacji komputerowych
2. **spiralność** kształcenia wokół tych samych informatycznych pojęć, metod i celów przez wszystkie lata w szkole od pierwszej po ostatnią klasę
3. **myślenie komputacyjne** – jednym z głównych celów kształcenia informatycznego jest rozwój sposobów myślenia angażowanego w formułowanie problemu i przedstawianie rozwiązań w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać; jest to sedno podejścia informatycznego do rozwiązywania problemów
4. **programowanie** – etap kreatywnego rozwiązywania problemów z wykorzystaniem komputera – *learning by doing,* konstrukcjonizm
5. **informatyka w swoich zastosowaniach** – nauczanie przez rozwiązywanie problemów z różnych dziedzin

6. **metoda projektów** – zalecana w podstawach wszystkich przedmiotów praca w zespołach – informatyk nie pracuje dzisiaj sam, samodzielnie, tylko w zespole

Te ogólne wskazania zostały uwzględnione w tej propozycji scenariusza i powinny zostać uwzględnione i zrealizowane przez nauczyciela, który ma poprowadzić pokazową/otwartą lekcję według tego scenariusza.

Zajęcia mają na celu „łagodne" przeprowadzenie uczniów ze środowiska Godziny Kodowania do środowiska programowania w języku Python, czyli przejście ze środowiska programowania wizualno-blokowego do środowiska programowania tekstowego. W tym pierwszym środowisku uczniowie tworzą programy dla gotowych łamigłówek, a w tym drugim – mają nieograniczoną swobodę. Jednak znajomość bloków z tego pierwszego środowiska pozwala im łatwo przenieść programy w postaci bloków do czysto tekstowego języka, gdyż faktycznie w każdym bloku jest również zapisany tekst tego, do czego służy blok.

Zajęcia na tej lekcji mają na celu wykształcenie u uczniów abstrakcyjnego myślenia o programowaniu – program to obiekt abstrakcyjny sytuacji, którą chcemy zaprogramować. W tym sensie postać zapisu programu ma drugorzędne znaczenie. Właściwe miejsce dla programowania w kształceniu informatycznym wskazuje niniejsze zalecenie metodyczne, które nauczyciel stara się przekazać uczniom, odpowiednio przygotowując i prowadząc zajęcia:

Zanim zaczniesz programować,
wpierw pomyśl komputacyjnie.

Zajęcia na tej lekcji są też okazją do wykorzystania wiedzy z matematyki na temat kątów, długości odcinków i tworzenia figur oraz ich kombinacji.

3. Temat lekcji: Przejście od Godziny Kodowania do Pythona

4. Treści nauczania

Z podstawy dla klas IV–VI:

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i zapisuje w wizualnym języku programowania:
      a) pomysły historyjek i rozwiązania problemów, w tym proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych oraz zdarzeń,
   2) testuje na komputerze swoje programy pod względem zgodności z przyjętymi założeniami i ewentualnie je poprawia, objaśnia przebieg działania programów;

Z podstawy dla klas VII–VIII:

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne […].

Uwaga! W podstawie programowej dla szkół podstawowych jest zalecenie, by rozpocząć programowanie w języku tekstowym. Takie środowisko jest niezbędne dla realizacji algorytmów podanych w podstawie w części I.

Pojęcia i metody informatyczne, którym są poświęcone zajęcia:

- programowanie wizualno-blokowe – przeciągnij i upuść blok
- programowanie tekstowe – wpisz tekst programu
- podstawowe bloki i instrukcje do budowy programów
- instrukcje iteracyjne i warunkowe
- algorytm dla danego celu
- program – algorytm dla komputera
- wykonanie programu
- poprawianie (debagowanie) programu
- programowanie wizualno-blokowe a tekstowe
- konstrukcje figur geometrycznych i ich aranżacji

5. Cele/osiągnięcia ucznia

Uczeń:

- orientuje się w strukturze programu napisanego w środowisku wizualno-blokowym, np. w środowisku Godziny Kodowania
- orientuje się w strukturze programu napisanego w języku Python
- stosuje w obu środowiskach instrukcje iteracyjne (pętle) i instrukcje warunkowe
- potrafi wyabstrahować tekst programu z bloków Godziny Kodowania
- stosuje instrukcje z modułu `turtle` w Pythonie
- transformuje program ze środowiska Godziny Kodowania do programu w Pythonie
- uruchamia tak otrzymane programy i ewentualnie poprawia je, gdy źle działają
- koncepcyjnie operuje konstrukcjami programistycznymi takimi jak: instrukcje iteracyjne (pętle) i instrukcje warunkowe, niezależnie od języka programowania

6. Metody pracy z uczniem, formy pracy uczniów

- **Słowna i oglądowa**: nauczyciel ilustruje na tablicy interaktywnej przykładowe powiązanie bloku z instrukcją tekstową, posługując się przy tym tekstem, który jest w bloku; takie ilustracje powinny dotyczyć wszystkich bloków, które wystąpią w zadaniach dla uczniów.
- **Czynna**: uczniowie otrzymują od nauczyciela wykazy bloków do wypełnienia obok nich odpowiednimi instrukcjami w Pythonie; najpierw rozwiązują na komputerze wybraną łamigłówkę w Godzinie Kodowania, a następnie piszą program w Pythonie, który tworzy identyczną lub podobną konstrukcję geometryczną; testują i debagują swoje programy w Pythonie.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, aktywizując się nawzajem; ponadto uczniowie mogą modyfikować swoje programy w Pythonie, by otrzymywać bardziej złożone konstrukcje geometryczne według własnych pomysłów.

7. Środki dydaktyczne wykorzystywane przez nauczyciela i przez uczniów

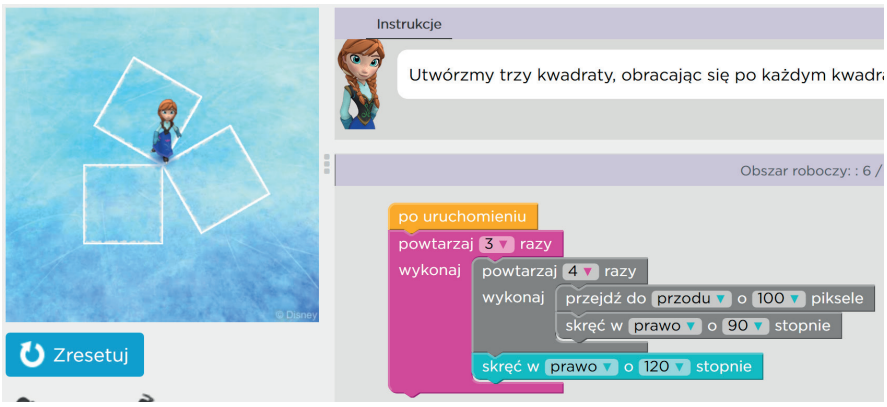
Dobrze jest przygotować listę bloków, które pojawiają się w łamigłówkach *Krainy lodu* z Godziny Kodowania, do wpisania obok nich przez uczniów podczas lekcji odpowiednich instrukcji w Pythonie.

Nauczyciel powinien wcześniej zapoznać się ze środowiskiem Godziny Kodowania https://code.org/. Aktualne informacje o tym środowisku są publikowane w blogu na stronie autora http://mmsyslo.pl.

W przypadku tej lekcji zaleca się skorzystanie z zestawu łamigłówek *Kraina lodu* z Anną i Elzą: https://studio.code.org/s/frozen/stage/1/puzzle/1.

Nauczyciel powinien mieć przygotowany przykład przejścia od instrukcji w blokach do tekstowego zapisu instrukcji z wykorzystaniem modułu `turtle` w Pythonie, by zademonstrować uczniom, na czym będzie polegało ich zadanie. Przykład takiego przejścia poniżej:

Ukończona łamigłówka nr 5 https://studio.code.org/s/frozen/stage/1/puzzle/5 dla 3 kwadratów, a w dalszej części – nr 6 dla 10 kwadratów, ale program napisany w postaci funkcji, by dobierać długość boku i liczbę kwadratów.



Teksty wypisane z bloków:

- powtarzaj 3 razy
- powtarzaj 4 razy
- przejdź do przodu 100
- skręć w prawo 90
- skręć w prawo 120

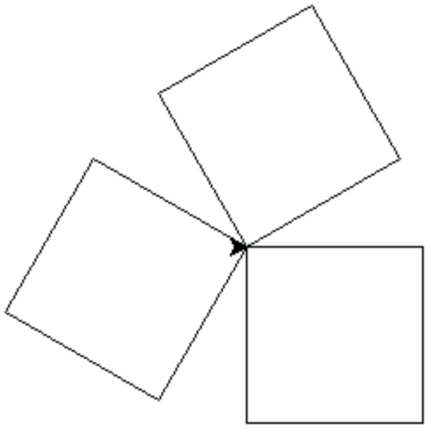
Odpowiedni program w Pythonie i efekt jego działania:

Kwadratowy płatek.py - D:\Dc

File Edit Format Run Opt

```
from turtle import *

for _ in range(3):
    for _ in range(4):
        forward(100)
        right(90)
    right(120)
```



Modyfikacja do funkcji i jej efekt:

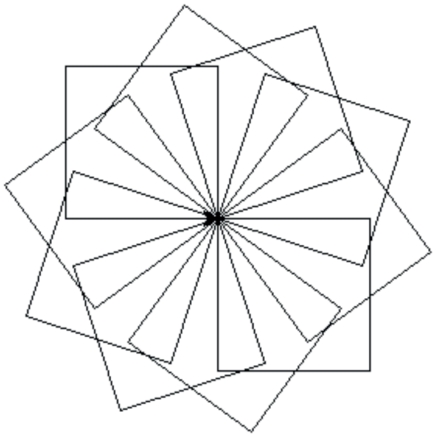
Kwadratowy płatek funkcja.py - D:/Documents/

File Edit Format Run Options Window

```
from turtle import *

def KwadratowyPłatek(bok,liczba):
    for _ in range(liczba):
        for _ in range(4):
            forward(bok)
            right(90)
        right(360/liczba)

KwadratowyPłatek(100,10)
```



8. Przebieg lekcji

1. *Wprowadzenie do lekcji*. Uczniowie zapewne dobrze znają film *Kraina lodu* (*Frozen*), nauczyciel może wprowadzić ich w atmosferę lekcji, uruchamiając na aktywnej tablicy piosenkę z tego filmu: https://www.youtube.com/watch?v=5BSY_bsflvk – pod koniec wyświetlania widać pojawiające się różne geometryczne konstrukcje powstające na skutek niskich temperatur. Łamigłówki w zestawie *Kraina lodu* https://studio.code.org/s/frozen/stage/1/puzzle/1 umożliwiają uczniom programowanie takich efektów.
2. *Rozpoczęcie zajęć przy komputerach*. Uczniowie rozwiązują łamigłówki z zestawu *Kraina lodu* https://studio.code.org/s/frozen/stage/1/puzzle/1. Nauczyciel zachęca, by obejrzeli pojawiające się filmy z napisami po polsku.
   Nie powinni mieć kłopotu z kolejnymi łamigłówkami. Łamigłówki 4 i 5 są wprowadzeniem do rysowania płatków z figur geometrycznych, rozwinięciem są łamigłówki nr 11 i 12, w których występują romby.
   To są dwa zestawy, które mogą być polecone uczniom, aby dalej się nimi zajmowali. Nauczyciel nie powinien zabraniać uczniom zajęcia się bardziej złożonymi płatkami śniegu, także według własnego pomysłu.

3. *Wprowadzenie nauczyciela*. Nauczyciel wyjaśnia, na czym będzie polegała lekcja – pokazuje, jak „wyciągnąć" teksty z bloków i utworzyć z nich program w Pythonie. Przykład jest w poprzedniej sekcji.
4. *Indywidualna prac uczniów.* Z programu blokowego ukończonej łamigłówki z Godziny Kodowania uczniowie wypisują na papierze teksty z kolejnych bloków. Następnie z tych tekstów tworzą program w Pythonie z wykorzystaniem modułu `turtle`.
   Uruchamiają programy w Pythonie i ewentualnie debagują je.
5. *Kontynuacja zajęć*. Podobnie, jak na przykładzie w poprzedniej sekcji, nauczyciel zachęca uczniów, by napisali program w takiej postaci, która łatwo umożliwia zmianę wyglądu płatka – do tego jest potrzebne pojęcie funkcji. Nauczyciel objaśnia na swoim przykładzie, jak to zrobić, również – jakie jest znaczenie funkcji w programowaniu.
6. *Bardziej zaawansowane zajęcia*. Uczniom, którzy szybko wykonali zadanie, nauczyciel proponuje modyfikację według własnego uznania, na przykład uwzględniając inne figury i ich ilość do tworzenia płatków śniegu.
7. *Uwaga metodyczna.* KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie odpowiadającym wybranemu programowi blokowemu z łamigłówek 5+6 lub 11+12.
8. *Dyskusja podsumowująca.* Zajęcia kończą się dyskusją na temat znaczenia języka programowania w programowaniu. Uczniowie znają już środowisko programów blokowych (Godzina Kodowania, Scratch) i tekstowych (Python), powinni więc po tych zajęciach umieć myśleć konstrukcjami programistycznymi (jak przypisanie, pętla, instrukcja warunkowa) w oderwaniu od ich konkretnej realizacji i zapisu w danym środowisku programowania. Będzie to oznaką przejścia na wyższy poziom abstrakcyjnego (komputacyjnego) myślenia, które jest ważne przy rozwijaniu umiejętności programowania, gdyż każdy program jest tworem abstrakcyjnym.
9. *Indywidualna praca uczniów i ich ocena*. W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie dostępnych jest bardzo wiele w sieci.

Ocena pracy uczniów – ogólne uwagi

Ocenianie powinno być motywujące dla uczniów kolejno przydzielanymi zadaniami do wykonania. Zadania są przydzielane wszystkim uczniom, ale zakres ich wykonania może być różny, może wymagać pomocy innego ucznia lub wsparcia przez nauczyciela.

Ocena postępów w programowaniu powinna uwzględniać zakres samodzielności, ale o tym uczniowie powinni być informowani na każdych zajęciach informatycznych. Taki system oceniania może dopingować uczniów do podejmowania coraz odważniejszych prób samodzielnego programowania, bez pomocy ucznia w parze, bez wsparcia nauczyciela i bez korzystania z pomocy „z sieci", czyli korzystania z gotowych rozwiązań, których obecnie dostępnych jest bardzo wiele w sieci.

Uwaga! Niestety, zmorą zadań programistycznych (np. domowych) jest obecnie nagminne korzystanie przez uczniów z kodów dostępnych w sieci (niezliczona liczba kodów w każdym języku programowania dla każdego zadania). Nauczyciel powinien wziąć to pod uwagę i wypracować własny system oceniania zadań programistycznych.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Poszerzenie zakresu tych zajęć może polegać na modyfikacji programów w języku Python, by zastosować w nich inne figury geometryczne do tworzenia płatków śniegu.

Innym poszerzeniem może być zastąpienie w rozważaniach języka Python przez C++, jeśli ten drugi język jest przedmiotem zajęć. Odradza się jednak zajmowanie uczniów zbyt wieloma środowiskami programowania. Wystarczy wprowadzenie do programowania wizualno-blokowego w Godzinie Kodowania, następnie programowanie w Scratchu i na koniec jeden z języków, ale nie oba, Python lub C++. To nie wyklucza, że niektórzy uczniowie mogą pisać programy w obu tych środowiskach tekstowego programowania. Językiem C++ mogą być zainteresowani uczniowie startujący w konkursach i olimpiadach informatycznych.