

Plan 5-dniowy (intensywny): OOP w Pythonie + Arcade

Dla ucznia znającego proceduralnego Pythona na poziomie Olimpiady Informatycznej Juniorów • ok. 6–8 h dziennie

Zasada: teoria OOP w małych dawkach rano, resztę dnia kodowanie w Arcade. Arcade jest obiektowe od podstaw, więc każdy dzień praktyki utrwala klasy, dziedziczenie i metody w naturalnym kontekście. Przed startem: `pip install arcade` (Python 3.9+).

Dzień 1 — Podstawy OOP + pierwsze okno

Rano (2–3 h): teoria

- Corey Schafer — Python OOP Tutorials, odcinki 1–3 (klasy i instancje, zmienne klasowe, `classmethod/staticmethod`). Oglądać z pauzami i przepisywać kod samodzielnie.

<https://www.youtube.com/playlist?list=PL-osiE80TeTsghluOqKhwlXsIBldSeYtc>

Po południu (4–5 h): praktyka

- Real Python — „Arcade: A Primer on the Python Game Framework”: okno, rysowanie, `sprite'y`, pętla gry, klawiatura. Przejść cały tutorial, potem zmodyfikować grę po swojemu (inne prędkości, nowy typ obiektu).

<https://realpython.com/arcade-python-game-framework/>

Wieczorem, do poduszki: PyCon 2018, Paul Craven — „Easy 2D Game Creation With Arcade” (~30 min) — przegląd możliwości biblioteki od jej autora.

<https://www.youtube.com/watch?v=DAWHMHMPVHU>

Dzień 2 — Dziedziczenie + `sprite'y`, kolizje, zdarzenia

Rano (1,5–2 h): teoria

- Corey Schafer, odcinki 4–5 (dziedziczenie i klasy pochodne, `dunder methods`).

Po południu (5–6 h): praktyka

- Wybrane rozdziały kursu autora biblioteki Arcade Academy — Learn Python: `sprite'y`, kolizje, poruszanie się, dźwięki. Rozdziały czysto wstępne (zmienne, pętle) pominąć — to poziom dawno opanowany.

<https://learn.arcade.academy/>

- Ćwiczenie na koniec dnia: własna mini-gra typu „zbieraj monety, unikaj wrogów” z klasą bazową `Enemy` i dwoma typami przeciwników dziedziczącymi po niej.

Dzień 3 — Platformówka, część 1

- Oficjalny tutorial platformówki (dokumentacja Arcade, zgodny z aktualnym API): gracz, fizyka, grawitacja, skoki, kamera, mapa poziomu. Przejść mniej więcej pierwszą połowę kroków.

https://api.arcade.academy/en/latest/tutorials/platform_tutorial/index.html

- Równoległe jako wideo-wsparcie: PyCon 2019, Paul Craven — „Build Your Own 2D Platformer Game”.

<https://www.youtube.com/watch?v=Djtm1DzWSvo>

- Darmowe assety (grafiki, dźwięki): kenney.nl/assets

Dzień 4 — Platformówka, część 2 + szlify OOP

- Dokończyć tutorial platformówki: wrogowie, punkty, wiele poziomów, ekran końca gry.
- Corey Schafer, odcinek 6 (property, `getter/setter`) — 20 min teorii i zastosować w kodzie platformówki (np. property „`health`” z ograniczeniem 0–100).

- Alternatywne ujęcie platformówki (tekst): Real Python — „Build a Platform Game in Python With Arcade” — uwaga: pisany pod starsze API, przy rozbieżnościach wygrywa oficjalna dokumentacja.

<https://realpython.com/platformer-python-arcade/>

Dzień 5 — Własny projekt

- Cały dzień na grę własnego pomysłu od zera: remake Tetrisa / Snake'a / Flappy Birda albo coś autorskiego. Wymóg projektowy: minimum 3 własne klasy, w tym jedna hierarchia dziedziczenia — tu OOP naprawdę „klika”.
 - Kopalnia gotowych przykładów do podglądania (kod źródłowy dziesiątek mini-gier):
https://api.arcade.academy/en/latest/example_code/index.html
 - Na koniec: wrzucić projekt na GitHuba — dobre domknięcie tygodnia.
-

Uwagi

- *Arcade 3.x zmieniło część API względem starszych tutoriali (np. `arcade.start_render()` → `self.clear()`). Przy rozbieżnościach zawsze wygrywa aktualna dokumentacja: `api.arcade.academy`.*
- *Materiały są po angielsku — przy poziomie olimpijskim to dobra okazja do angielskiego technicznego.*
- *Jeśli tempo okaże się za szybkie, dzień 5 można poświęcić na dokończenie platformówki, a własny projekt przenieść na później.*
- *Skill Tree w dokumentacji (`api.arcade.academy`) pokazuje, co zgłębiać po tych 5 dniach: shadery, GUI, sieć itd.*