

KARTA KURSU (realizowanego w specjalności)**Inżynieria oprogramowania**

(nazwa specjalności)

Nazwa	Programowanie Obiektowe w Języku Python
Nazwa w j. ang.	<i>Object-Oriented Programming in Python</i>

Koordinator	dr Roman Czapla	Zespół dydaktyczny
		dr Roman Czapla mgr inż. Agnieszka Kańska mgr Katarzyna Marczał
Punktacja ECTS*	5	

Opis kursu (cele kształcenia)

Celem kursu jest przedstawienie programowania obiektowego z perspektywy języka Python, z uwzględnieniem różnic i specyfiki w porównaniu do języków takich jak C++. Studenci pogłębiają znajomość paradygmatu obiektowego, poznając idiomatyczne rozwiązania Pythona, takie jak dynamiczna struktura klas, brak jawnej deklaracji typów, właściwości, protokoły, dekoratory i metaprogramowanie.

Zajęcia skupiają się na praktycznym zastosowaniu mechanizmów obiektowych w Pythonie, rozszerzonych o współczesne elementy języka, jak klasy danych, typowanie z wykorzystaniem adnotacji, menadżery kontekstu czy programowanie asynchroniczne. Omawiane są także wzorce projektowe w kontekście dynamicznego typowania i elastyczności składni.

W ramach kursu studenci realizują miniaplikację (np. w Pygame lub PyQt), która integruje poznane techniki programowania obiektowego w Pythonie.

Warunki wstępne

Wiedza	Student zna podstawy programowania obiektowego w języku C++, w szczególności pojęcia klasy, obiektu, dziedziczenia, polimorfizmu oraz enkapsulacji. Posiada podstawową znajomość składni i semantyki języka Python w zakresie programowania proceduralnego.
Umiejętności	Student potrafi napisać proste programy w języku Python, korzystając z podstawowych konstrukcji (zmienne, funkcje, instrukcje sterujące) oraz potrafi zastosować zasady programowania obiektowego w praktyce (na przykładzie C++).
Kursy	<u>Wymagane zaliczenie kursu</u> : Podstawy Programowania w języku Python, Programowanie Obiektowe

Efekty uczenia się

	Efekt uczenia się dla kursu	Odniesienie do efektów kierunkowych
Wiedza	Po zakończeniu kursu student: W01: zna specyfikę programowania obiektowego w języku Python i potrafi porównać ją z innymi językami, takimi jak C++.	S1_W02
	W02: zna składnię i semantykę Pythona w kontekście programowania obiektowego, w tym konstrukcje klas, metod i atrybutów.	S1_W02
	W03: rozumie mechanizmy dziedziczenia, polimorfizmu, enkapsulacji oraz ich idiomatyczne zastosowania w Pythonie.	S1_W02
	W04: zna zaawansowane elementy języka, takie jak dekoratory, deskryptory, klasy danych, metaprogramowanie i protokoły.	S1_W02
	W05: Student rozumie znaczenie wzorców projektowych, zasad SOLID, testowania, refaktoryzacji i statycznego typowania w Pythonie.	S1_W02
	Efekt uczenia się dla kursu	Odniesienie do efektów kierunkowych
Umiejętności	Po zakończeniu kursu student: U01: potrafi projektować klasy i ich relacje (dziedziczenie, kompozycja) z uwzględnieniem dobrych praktyk obiektowych.	S1_U02 S1_U06
	U02: potrafi wykorzystywać specyficzne mechanizmy języka Python, takie jak metody specjalne, menadżery kontekstu czy adnotacje typów.	S1_U02 S1_U06
	U03: potrafi implementować wybrane wzorce projektowe z użyciem języka Python.	S1_U02 S1_U06
	U04: potrafi tworzyć kod obiektowy zgodny z zasadami SOLID oraz stosować testy jednostkowe i refaktoryzację.	S1_U02 S1_U06
	U05: potrafi zbudować niedużą aplikację z wykorzystaniem biblioteki (np. Pygame, PyQt), integrując różne techniki obiektowe.	S1_U02 S1_U06

	Efekt uczenia się dla kursu	Odniesienie do efektów kierunkowych
Kompetencje społeczne	Po zakończeniu kursu student:	
	K01: potrafi efektywnie współpracować z innymi przy realizacji zadań programistycznych.	S1_ K03
	K02: potrafi korzystać z dokumentacji i źródeł zewnętrznych w celu rozwiązywania problemów technicznych.	S1_ K04
	K03: wykazuje odpowiedzialność za jakość własnej pracy programistycznej i przestrzega zasad etyki zawodowej.	S1_ K04

Studia stacjonarne

Organizacja							
Forma zajęć	Wykład (W)	Ćwiczenia w grupach					
		A	K	L	S	P	E
Liczba godzin	30			30			

Studia niestacjonarne

Organizacja							
Forma zajęć	Wykład (W)	Ćwiczenia w grupach					
		A	K	L	S	P	E
Liczba godzin	15			20			

Opis metod prowadzenia zajęć

Wykład ma na celu wprowadzenie zagadnień teoretycznych związanych z programowaniem obiektowym w języku Python, prezentację idiomatycznych konstrukcji językowych oraz omówienie różnic między Pythonem a językami statycznie typowanymi (takimi jak C++). Omawiane są mechanizmy klas, dziedziczenia, metaprogramowania, wzorce projektowe oraz dobre praktyki w projektowaniu kodu.

Ćwiczenia laboratoryjne mają charakter praktyczny i służą utrwaleniu zagadnień poruszanych na wykładzie. Studenci rozwiązują zadania programistyczne oraz przygotowują projekt zaliczeniowy, który integruje poznane techniki programowania obiektowego. Praca odbywa się indywidualnie oraz w małych zespołach, z bieżącym wsparciem prowadzącego.

Formy sprawdzania efektów uczenia się

	E – learning	Gry dydaktyczne	Ćwiczenia w szkole	Zajęcia terenowe	Praca laboratoryjna	Projekt indywidualny	Projekt grupowy	Udział w dyskusji	Referat	Praca pisemna (esej)	Egzamin ustny	Egzamin pisemny	Zadania problemowe
W01					x		x	x				x	
W02					x		x	x				x	
W03					x		x	x				x	
W04					x		x	x				x	
W05					x		x	x				x	
U01					x	x	x	x				x	x
U02					x	x	x	x				x	x
U03					x	x	x	x				x	x
U04					x	x	x	x				x	x
U05					x	x	x	x				x	x
K01								x					
K02								x					
K03								x					

Kryteria oceny	Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna.
	Zaliczenie przedmiotu odbywa się na podstawie dwóch kolokwii sprawdzających znajomość zagadnień teoretycznych i praktycznych z zakresu programowania obiektowego oraz projektu zaliczeniowego realizowanego indywidualnie lub w dwuosobowych zespołach.
	Warunkiem uzyskania zaliczenia jest: • zaliczenie obu kolokwii (wymagana ocena pozytywna z każdego), • złożenie i obrona projektu zaliczeniowego, • aktywne uczestnictwo w zajęciach.
	Ocena końcowa wyliczana jest jako średnia z ocen cząstkowych (kolokwia i projekt), z możliwością uwzględnienia aktywności studenta na zajęciach. Zaliczenie na ocenę dobrą lub bardzo dobrą otrzymuje student, który spełnia warunki oceny dostatecznej, a ponadto: • projektuje rozwiązania programistyczne zgodne z dobrymi praktykami i zasadami SOLID, • stosuje zaawansowane techniki języka Python (np. dekoratory, wzorce projektowe, typowanie), • tworzy kod przejrzysty, dobrze udokumentowany i przetestowany, • potrafi wyjaśnić zastosowane rozwiązania oraz świadomie dobrać odpowiednie mechanizmy obiektowe do danego problemu. Obecność na wykładach jest warunkiem zaliczenia tej części kursu.

Uwagi	
-------	--

Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do programowania obiektowego w języku Python: klasy i obiekty, atrybuty, metody instancyjne i klasowe, konstruktor i metody specjalne.
2. Mechanizmy obiektowe: dziedziczenie, polimorfizm, kompozycja, właściwości i kontrola dostępu do atrybutów.

3. Iterowalność i protokoły: tworzenie obiektów iterowalnych i generatorów, przeciążanie operatorów, menadżery kontekstu.
4. Rozszerzone techniki obiektowe: klasy danych, wielokrotne dziedziczenie, klasy abstrakcyjne i protokoły typów.
5. Wzorce projektowe: wybrane wzorce projektowe (kreacyjne, strukturalne i behawioralne) oraz ich zastosowanie w Pythonie.
6. Wprowadzenie do współbieżności i asynchroniczności: podstawowe koncepcje, proste zastosowania wątków i korutyn.
7. Metaprogramowanie: dekoratory, deskryptory, podstawy metaklas, introspekcja i manipulacja strukturą obiektów.
8. Testowanie i dobre praktyki: testy jednostkowe, podstawy TDD, refaktoryzacja kodu, zasady SOLID i stosowanie adnotacji typów.
9. Przegląd nowych elementów języka Python oraz porównanie z innymi językami obiektowymi (np. C++): różnice w podejściu do typowania, dziedziczenia, enkapsulacji i metod specjalnych; idiomatyczne cechy Pythona w kontekście programowania obiektowego.
10. Projekt zaliczeniowy - realizacja miniaplikacji z użyciem wybranej biblioteki (np. Pygame, PyQt, Tkinter), integrującej poznane koncepcje programowania obiektowego.

Wykaz literatury podstawowej

Wskazane przez prowadzącego rozdziały:

1. B. Slatkin, *Efektywny Python. 125 sposobów na lepszy kod. Wydanie III*, Helion, Gliwice 2025;
2. A. Martelli, A. Martelli Ravenscroft, S. Holden, P. McGuire, *Python w pigułce. Podręczny przewodnik po wersjach 3.10 i 3.11*, Promise, Warszawa 2023;
3. L. Ramalho, *Zaawansowany Python, wyd. 2. Przejrzyste, zwarte i efektywne programowanie*, Promise, Warszawa 2022;
4. S. F. Lott, D. Phillips, *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries*, 4th Edition, Packt Publishing 2021;
5. A. Sweigart, *Programowanie w Pythonie dla średnio zaawansowanych. Najlepsze praktyki tworzenia czystego kodu*, Helion, Gliwice 2021;
6. P. J. Deitel, H. Deitel, *Python dla programistów. Big Data i AI. Studia przypadków*, Helion, 2020;
7. D. Beazley, B.K. Jones, *Python. Receptury. Wydanie III*, Helion, Gliwice 2014;
8. M. Summerfield, *Python 3. Kompletnie wprowadzenie do programowania. Wydanie II*, Helion, Gliwice 2010.

Wykaz literatury uzupełniającej

1. I. Kalb, *Python zorientowany obiektowo. Programowanie gier i graficznych interfejsów użytkownika*, Helion, Gliwice 2022;
2. Ch. Mayer, *Kod Pythona w jednym wierszu. Jak profesjonaliści piszą programy doskonałe*, Helion Gliwice 2021;
3. M. Gorelick, I. Ozsvald, *Wysoko wydajny Python. Efektywne programowanie w praktyce. Wydanie II*, Helion, Gliwice 2021;
4. B. Slatkin, *Efektywny Python. 90 sposobów na lepszy kod. Wydanie II*, Helion, Gliwice 2020;
5. D. Kopec, *Klasyczne problemy informatyki w Pythonie*, PWN, Warszawa 2020;
6. S. F. Lott, *Modern Python Cookbook: 133 recipes to develop flawless and expressive programs in Python 3.8*, 2nd Edition, Packt Publishing, 2020;
7. S. F. Lott, *Python. Programowanie funkcyjne*, Helion, Gliwice 2019;
8. M. Lutz, *Python. Wprowadzenie. Wydanie IV*, Helion, Gliwice 2010.

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) – **studia stacjonarne**

Liczba godzin w kontakcie z prowadzącymi	Wykład	30
	Konwersatorium (ćwiczenia, laboratorium itd.)	30
	Pozostałe godziny kontaktu studenta z prowadzącym	2
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	18
	Realizacja zadań domowych (problemowych) po zapoznaniu się z niezbędną literaturą przedmiotu	5
	Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna lub w grupie)	25
	Przygotowanie do egzaminu/zaliczenia	25
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) – **studia niestacjonarne**

Liczba godzin w kontakcie z prowadzącymi	Wykład	15
	Konwersatorium (ćwiczenia, laboratorium itd.)	20
	Pozostałe godziny kontaktu studenta z prowadzącym	2
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	23
	Realizacja zadań domowych (problemowych) po zapoznaniu się z niezbędną literaturą przedmiotu	10
	Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna lub w grupie)	25
	Przygotowanie do egzaminu/zaliczenia	35
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5