

KARTA KURSU (realizowanego w module specjalności)**Administracja systemami informatycznymi**

Nazwa	Programowanie obiektowe 2
Nazwa w j. ang.	<i>Object Oriented Programming 2</i>

Koordynator	dr Roman Czapla	Zespół dydaktyczny
		dr Roman Czapla mgr inż. Katarzyna Marczak
Punktacja ECTS*	5	

Opis kursu (cele kształcenia)

Celem kursu jest nauka programowania obiektowego oraz poznanie technik zaawansowanego programowania z zastosowaniem języka Python 3.x. Kurs prowadzony jest w języku polskim.

Warunki wstępne

Wiedza	Student zna podstawy analizy, projektowania i programowania obiektowego w dowolnym języku programowania. Student zna podstawy składni języka Python 3.x.
Umiejętności	Potrafi zapisywać podstawowe algorytmy i struktury danych w wybranym języku, projektuje oraz tworzy z wykorzystaniem podstaw metodologii obiektowej programy w języku obiektowym. Student potrafi pisać proste programy użytkowe w języku Python 3.x.
Kursy	<u>Wymagane zaliczenie kursu:</u> Języki skryptowe

Efekty uczenia się

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Wiedza	Po zakończeniu kursu student:	
	W01: zna podstawowe pojęcia związane z programowaniem obiektowym: klasa, obiekt, dziedziczenie, polimorfizm, itp.	S1_W01
	W02: zna zaawansowane techniki programowania zorientowanego obiektowego	S1_W01
	W03: zna elementy metaprogramowania oraz innych technik pozwalających na tworzenie zaawansowanych i zupełnych programów	S1_W01
	W04: wie jak dbać o jakość pisanych programów i jak je testować	S1_W01
	W05: zna wybrane biblioteki pozwalające tworzyć interfejsy graficzne i/lub proste gry	S1_W01

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Umiejętności	Po zakończeniu kursu student:	
	U01: potrafi projektować oraz implementować własne klasy różnorodnych obiektów i tworzyć ich instancje	S1_ U05
	U02: projektuje i tworzy kompletne programy o znacznej złożoności z wykorzystaniem metodologii obiektowej, implementuje wybrane wzorce projektowe	S1_ U05 S1_ U06
	U03: potrafi korzystać z zaawansowanych technik programowania zorientowanego obiektowo w celu udoskonalenia tworzonych programów m.in. potrafi tworzyć programy z użyciem wyjątków i rozumie potrzebę ich obsługi, stosuje elementy metaprogramowania	S1_ U05 S1_ U06
	U04: potrafi pisać odpowiednie testy i testować swoje programy	S1_ U05 S1_ U08
	U05: potrafi wykorzystać zdobyte umiejętności do projektowania interfejsów graficznych i/lub prostych gier 2D z wykorzystaniem odpowiednich bibliotek	S1_ U05 S1_ U06 S1_ U08
	Efekt uczenia się	Odniesienie do efektów kierunkowych
Kompetencje społeczne	Po zakończeniu kursu student:	
	K01: potrafi korzystać z różnych źródeł informacji (w tym zasobów sieciowych) do poszerzania własnej wiedzy i zdobywania nowych umiejętności	S1_K01
	K02: wykazuje umiejętność stosowania w praktyce zdobytej wiedzy przedmiotowej i potrafi działać kreatywnie w celu rozwiązywania postawionych mu zadań.	S1_K02
	K03: potrafi przekazywać wiedzę informatyczną w sposób zrozumiały dla innych i współpracować w zespole	S1_K03

Studia stacjonarne

Organizacja												
Forma zajęć	Wykład (W)	Ćwiczenia w grupach										
		A		K		L		S		P		E
Liczba godzin	15					30						

Studia niestacjonarne

Organizacja													
Forma zajęć	Wykład (W)	Ćwiczenia w grupach											
		A		K		L		S		P		E	
Liczba godzin	10					20							

Opis metod prowadzenia zajęć

Kurs składa się z wykładu i ćwiczeń prowadzonych w formie laboratoriów. W ramach laboratoriów studenci projektują i tworzą zadane programy w języku Python 3.x, które następnie są wspólni omawiane. Studenci w ramach pracy indywidualnej realizują zadane projekty programistyczne, które są omawiane z prowadzącym zajęcia.

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	Inne
W01					x	x		x				x	
W02					x	x		x				x	
W03					x	x		x				x	
W04					x	x		x					
W05					x	x		x					
U01					x	x		x				x	
U02					x	x		x				x	
U02					x	x		x				x	
U03					x	x		x					
U04					x	x		x					
U05					x	x		x					
K01					x	x		x					
K02					x	x		x					
K03					x	x		x					

Kryteria oceny

Ocena jest wystawiana na podstawie wykonanych przez studentów projektów programistycznych i/lub testu sprawdzającego wiedzę z zakresu omawianych treści merytorycznych.

Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna.

Ocenę dobrą i bardzo dobrą może uzyskać student, który:

- który wykazuje się dobrą lub bardzo dobrą znajomością składni i narzędzi języka Python 3.x pozwalających programować z wykorzystaniem metodologii obiektowej
- potrafi wykorzystywać zaawansowane techniki programowania obiektowego do tworzenia wyspecjalizowanych klas i budowy odpowiednich ich hierarchii
- zna i potrafi wykorzystać w swoich programach aspekty metaprogramowania
- potrafi implementować wybrane wzorce projektowe
- potrafi dbać o jakość pisanych programów, zna podstawowe sposoby refaktoryzacji kodu źródłowego i metody testowania programów
- potrafi projektować i tworzyć interfejsy graficzne oraz/lub proste gry 2D z wykorzystaniem metodologii obiektowej

Uwagi

Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do projektowania obiektowego w języku Python
 - własne klasy, atrybuty i metody, tworzenie instancji obiektów
 - pojęcie konstruktora/inicjalizatora i metody specjalne
 - atrybuty i metody klasy – atrybuty statyczne

- prywatność atrybutów w języku Python
- kontrola dostępu do atrybutów – właściwości
- dziedziczenie i polimorfizm
- wyjątki jako obiekty, sposób przechwytywania wyjątków, tworzenie i zgłaszanie wyjątków
- 2. Programowanie zorientowane obiektowo w języku Python
 - iteratory i generatory, obiekty iterowalne
 - przeciążanie operatorów
 - tworzenie w pełni zintegrowanych typów danych, rozszerzanie typów wbudowanych
 - wielokrotne dziedziczenie i klasy domieszkowe
 - menadżery kontekstu i klasy danych
 - protokoły, interfejsy i abstrakcyjne klasy bazowe
- 3. Metaprogramowanie
 - dekoratory funkcji i klas
 - deskryptory, ich rodzaje i zastosowania
 - metoda specjalna `__new__()`
 - metaklasy
- 4. Wybrane elementy programowania w języku Python
 - Python jako język wspierający paradygmat funkcyjny
 - wybrane wzorce projektowe
 - testowanie kodu, dezasemblacja, refaktoryzacja
 - podstawy programowania sterowanego testami
 - nowe elementy języka Python wprowadzane w kolejnych wersjach
 - wskazówki do typów i współprogramy
- 5. Zastosowanie technik programowania obiektowego (jedna wybrana koncepcja)
 - tworzenie prostych gier 2D z wykorzystaniem biblioteki pygame
 - projektowanie interfejsów graficznych z użyciem np. biblioteki PyQt6

Wykaz literatury podstawowej

Wskazane przez prowadzącego rozdziały:

1. A. Martelli, A. Martelli Ravenscroft, S. Holden, P. McGuire, Python w pigułce. Podręczny przewodnik po wersjach 3.10 i 3.11, Promise, 2023;
2. L. Ramalho, Zaawansowany Python, wyd. 2. Przejrzyste, zwarte i efektywne programowanie, Promise, 2022;
3. I. Kalb, Python zorientowany obiektowo. Programowanie gier i graficznych interfejsów użytkownika, Helion, 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. Al Sweigart, Programowanie w Pythonie dla średnio zaawansowanych. Najlepsze praktyki tworzenia czystego kodu, Helion, 2021;
6. P. J. Deitel, H. Deitel, Python dla programistów. Big Data i AI. Studia przypadków, Helion, 2020;
7. S. F. Lott, Modern Python Cookbook: 133 recipes to develop flawless and expressive programs in Python 3.8, 2nd Edition, Packt Publishing, 2020;
8. D. Beazley, B.K. Jones, Python. Receptury. Wydanie III, Helion, 2014;
9. M. Summerfield, Python 3. Kompletnie wprowadzenie do programowania. Wydanie II", Helion, 2010.

Wykaz literatury uzupełniającej

1. Ch. Mayer, Kod Pythona w jednym wierszu. Jak profesjonaliści piszą programy doskonałe, Helion 2021;
2. M. Gorelick, I. Ozsvald, Wysoko wydajny Python. Efektywne programowanie w praktyce. Wydanie II, Helion 2021;
3. B. Slatkin, Efektywny Python. 90 sposobów na lepszy kod. Wydanie II, Helion, 2020;
4. D. Kopec, Klasyczne problemy informatyki w Pythonie, PWN, 2020;
5. S. F. Lott, Python. Programowanie funkcyjne, Helion, 2019;
6. M. Lutz, Python. Wprowadzenie. Wydanie IV, Helion 2010.

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

Liczba godzin w kontakcie z prowadzącymi	Wykład	15
	Konwersatorium (ćwiczenia, laboratorium itd.)	30
	Pozostałe godziny kontaktu studenta z prowadzącym	5
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	30
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	25
	Przygotowanie do egzaminu	20
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	10
	Konwersatorium (ćwiczenia, laboratorium itd.)	20
	Pozostałe godziny kontaktu studenta z prowadzącym	5
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	40
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	25
	Przygotowanie do egzaminu	25
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5