KARTA KURSU (realizowanego w module specjalności)

Administracja systemami informatycznymi (ASI)
(nazwa specjalności)

Nazwa	**Programowanie obiektowe 2**
Nazwa w j. ang.	Object Oriented Programming 2

Koordynator	dr Roman Czapla	Zespół dydaktyczny
		dr Roman Czapla dr Wojciech Nawalaniec
Punktacja ECTS*	st. stacjonarne: 5 st. niestacjonarne:5	

Opis kursu (cele kształcenia)

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

Warunki wstępne

Wiedza	Student zna podstawy analizy, projektowania i programowani obiektowego oraz podstawy języka C++. Student zna podstawy składni języka Python 3.x.
Umiejętności	Potrafi zapisywać podstawowe algorytmy i struktury danych w języku C++, projektuje oraz tworzy z wykorzystaniem podstaw metodologii obiektowej programy w języku C++. Student potrafi pisać proste programy użytkowe w języku Python 3.x.
Kursy	Programowanie obiektowe, Języki skryptowe

Efekty uczenia się

	Efekt uczenia się dla kursu	Odniesienie do efektów dla specjalności (określonych w karcie programu studiów dla modułu specjalnościowego)
Wiedza	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 proste gry	S1_W01

	Efekt uczenia się dla kursu	Odniesienie do efektów dla specjalności (określonych w karcie programu studiów dla modułu specjalnościowego)
Umiejętności	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
	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
	U04: potrafi pisać odpowiednie testy i testować swoje programy	S1_ U05 S1_ U08 S1_ U12
	U05: potrafi wykorzystać zdobyte umiejętności do projektowania interfejsów graficznych i prostych gier 2D z wykorzystaniem odpowiednich bibliotek w ramach projektu programistycznego	S1_ U05 S1_ U06 S1_ U12

	Efekt kształcenia dla kursu	Odniesienie do efektów dla specjalności (określonych w karcie programu studiów dla modułu specjalnościowego)
Kompetencje społeczne	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 wiedzą 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	X					
W05					X	X	X	X					
U01					X	X	X	X					
U02					X	X	X	X					
U03					X	X	X	X					
U04					X	X	X	X					
U05					X	X	X	X					
K01					X	X							
K02					X	X	X	X					
K03					X	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. 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 metaprogramownia • 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 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
    - prywatne i chronione atrybuty w języku Python
    - kontrola dostępu do atrybutów – właściwości
    - dziedziczenie i polimorfizm
    - rodzaje wyjątków, zastosowanie i podstawy działania
    - 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
    - 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
    - obsługa plików tekstowych i binarnych, serializacja obiektów
    - Python jako język wspierający paradygmat funkcyjny
    - moduły i pakiety
    - wybrane wzorce projektowe
    - testowanie kodu, dezasemblacja, refaktoryzacja
    - podstawy programowania sterowanego testami
5. Zastosowanie technik programowania obiektowego
    - tworzenie prostych gier 2D z wykorzystaniem biblioteki pygame
    - projektowanie interfejsów graficznych z użyciem biblioteki PyQt5

Wykaz literatury podstawowej

Wybrane fragmenty z książek:

1. D. Phillips, *„Python 3 Object-Oriented Programming, Third Edition”,* Packt Publishing 2018
2. S. F. Lott, *„Mastering Object-oriented Python, Second Edition”*, Packt Publishing 2019
3. D. Beazley, B.K. Jones, *„Python. Receptury. Wydanie III”*, Helion 2014
4. S. F. Lott, „*Python. Programowanie funkcyjne”*, Helion 2019
5. C. Jackson, „*Python ninja”*, Helion, 2019
6. J. Danjou, „*Python na poważnie”*, PWN, 2019

Wykaz literatury uzupełniającej

1. M. Lutz, „*Python. Leksykon kieszonkowy. Wydanie V*”, Helion 2014
2. M. Lutz, „*Python. Wprowadzenie. Wydanie IV*”, Helion 2010
3. S. F. Lott, „*Modern Python Cookbook”*, Packt Publishing 2016
4. M. Jaworski, T. Ziade, *„Expert Python Programming, Third Edition”*, Packt Publishing 2019

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	15
liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	20
	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/zaliczenia	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	10
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)	30
	Przygotowanie do egzaminu/zaliczenia	25
Ogółem bilans czasu pracy		125
Liczba punktów ECTS w zależności od przyjętego przelicznika		5