

KARTA KURSU

Nazwa	Podstawy programowania w języku Python
Nazwa w j. ang.	Python Programming Fundamentals

Koordynator	dr Roman Czapla	Zespół dydaktyczny
		dr Iryna Artyshchuk dr Roman Czapla mgr Katarzyna Marczak mgr Patryk Mazurek dr Anna Wojciechowska
Punktacja ECTS*	4	

Opis kursu (cele kształcenia)

Celem kursu jest zapoznanie studentów z podstawami programowania w języku Python, przy jednoczesnym wykorzystaniu posiadanych już umiejętności programistycznych w języku C. Kurs koncentruje się na specyfice języka Python, jego składni oraz różnicach w stosunku do języka C, a także na prezentacji dobrych praktyk programistycznych.

Studenci uczą się rozwiązywać problemy algorytmiczne i inżynierskie przy użyciu Pythona, poznają podstawowe techniki i struktury danych oraz zdobywają umiejętność pracy z biblioteką standardową i wybranymi pakietami zewnętrznymi. Kurs kładzie nacisk zarówno na pisanie poprawnego i czytelnego kodu, jak i na wykorzystywanie charakterystycznych mechanizmów Pythona, takich jak obsługa wyjątków, praca z plikami czy prosta serializacja danych.

Warunki wstępne

Wiedza	Student zna podstawowe pojęcia programowania strukturalnego, takie jak zmienne, typy danych, instrukcje sterujące, funkcje oraz proste struktury danych (tablice, wskaźniki). Posiada elementarną wiedzę z zakresu algorytmiki i rozwiązywania prostych problemów obliczeniowych.
Umiejętności	Student potrafi pisać i uruchamiać proste programy w języku C, wykorzystując zmienne, operatory, instrukcje warunkowe i pętle. Potrafi korzystać z funkcji, pracować z tablicami i stosować podstawowe techniki modularnego programowania. Umie korzystać z dokumentacji i materiałów dydaktycznych w języku polskim i angielskim.
Kursy	<u>Wymagane zaliczenie kursu:</u> Programowanie

Efekty uczenia się

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Wiedza	Po zakończeniu kursu student: W01: zna podstawowe różnice i podobieństwa między językiem Python a językiem C, rozumie ideę języka interpretowanego i automatycznego zarządzania pamięcią.	K_W06
	W02: zna typy danych i operatory w Pythonie oraz podstawowe struktury danych, takie jak listy, krotki, słowniki i zbiory.	K_W06
	W03: zna składnię instrukcji sterujących i pętli w Pythonie oraz rozumie różnice względem języka C.	K_W06
	W04: posiada wiedzę na temat definiowania funkcji, typów parametrów, modułów i pracy z biblioteką standardową.	K_W06

	<p>W05: zna podstawowe techniki programistyczne charakterystyczne dla Pythona, takie jak listy składane, generatory czy funkcje wyższego rzędu, oraz ich zastosowania.</p> <p>W06: zna podstawowe mechanizmy obsługi wyjątków i pracy z plikami oraz rozumie zasady serializacji danych w Pythonie.</p>	<p>K_W06</p> <p>K_W06</p>
	Efekt uczenia się	Odniesienie do efektów kierunkowych
Umiejętności	<p>Po zakończeniu kursu student:</p> <p>U01: potrafi pisać i uruchamiać proste programy w Pythonie, stosując poprawną składnię i konwencje języka.</p> <p>U02: umie korzystać z podstawowych struktur danych i instrukcji sterujących do rozwiązywania prostych problemów.</p> <p>U03: potrafi definiować i wykorzystywać funkcje, importować moduły oraz korzystać z wybranych bibliotek standardowych.</p> <p>U04: umie stosować techniki programistyczne charakterystyczne dla Pythona, takie jak listy składane, generatory czy funkcje lambda.</p> <p>U05: potrafi obsługiwać błędy wykonania przy użyciu mechanizmu wyjątków.</p> <p>U06: umie zapisywać i odczytywać dane z plików tekstowych oraz stosować podstawowe mechanizmy serializacji, takie jak JSON czy pickle.</p>	<p>K_U05</p> <p>K_U05</p> <p>K_U05</p> <p>K_U05</p> <p>K_U05</p> <p>K_U05</p>

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Kompetencje społeczne	<p>Po zakończeniu kursu student:</p> <p>K01: potrafi współpracować w zespole przy realizacji prostych projektów programistycznych.</p> <p>K02: dostrzega znaczenie czytelności, przejrzystości i jakości kodu źródłowego.</p> <p>K03: rozumie konieczność rozwijania swoich umiejętności programistycznych i poszerzania wiedzy o bardziej zaawansowane elementy języka Python.</p>	<p>K_K02</p> <p>K_K01</p> <p>K_K01</p>

Studia stacjonarne

Organizacja											
Forma zajęć	Wykład (W)	Ćwiczenia w grupach									
		A		K		L		S		P	E
Liczba godzin	10					30					

Studia niestacjonarne

Organizacja											
Forma zajęć	Wykład (W)	Ćwiczenia w grupach									
		A		K		L		S		P	E
Liczba godzin	6					20					

Opis metod prowadzenia zajęć

Wykłady mają charakter teoretyczny i obejmują omówienie materiału z naciskiem na różnice między językiem Python a językiem C oraz na charakterystyczne cechy Pythona, takie jak interpretacja, automatyczne zarządzanie pamięcią czy specyficzne struktury danych. Omawiane zagadnienia są ilustrowane przykładami kodu.

Ćwiczenia mają charakter praktyczny. Studenci implementują programy w Pythonie, ucząc się poprawnej składni języka, korzystania z podstawowych i złożonych struktur danych, stosowania instrukcji sterujących oraz wykorzystywania charakterystycznych technik, takich jak listy składane, funkcje lambda, generatory czy obsługa wyjątków. Ważnym elementem zajęć jest analiza i omawianie różnych podejść do rozwiązania problemów programistycznych.

Na zakończenie kursu uczestnicy przystępują do testu sprawdzającego poziom opanowania materiału. Test obejmuje zarówno pytania teoretyczne, jak i praktyczne zadania programistyczne, pozwalając zweryfikować zdobytą wiedzę i umiejętności w zakresie podstaw programowania w języku Python.

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					
W02					x	x		x					
W03					x	x		x					
W04					x	x		x					
W05					x	x		x					
W06					x	x		x					
U01					x	x		x					
U02					x	x		x					
U02					x	x		x					
U03					x	x		x					
U04					x	x		x					
U05					x	x		x					
U06					x	x		x					
K01								x					
K02								x					
K03								x					

Kryteria oceny	<p>Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania</p> <p>Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna. Warunkiem jej przyznania jest zdobycie minimalnej wymaganej liczby punktów na teście końcowym.</p> <p>Ocena dobra (4.0) może zostać przyznana studentowi, który wykazuje się dobrą znajomością języka Python, potrafi pisać efektywny i czytelny kod, a także stosuje zaawansowane funkcje i struktury danych. W bardziej złożonych zadaniach mogą</p>
----------------	---

pojawiać się drobne błędy, jednak ogólna poprawność rozwiązania nie budzi zastrzeżeń. Potwierdzeniem tego poziomu opanowania materiału jest uzyskanie odpowiedniej liczby punktów na teście końcowym.

Ocena bardzo dobra (5.0) przysługuje studentowi, który posiada pełną kontrolę nad językiem Python, potrafi pisać optymalny, dobrze zorganizowany i przejrzysty kod. Efektywnie stosuje zaawansowane techniki programistyczne i potrafi rozwiązywać złożone problemy algorytmiczne. Potwierdzeniem tego poziomu wiedzy i umiejętności jest zdobycie wysokiej liczby punktów na teście końcowym.

Obecność na wykładach jest warunkiem koniecznym zaliczenia tej części kursu.

Uwagi

Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do języka Python i jego różnice względem języka C
 - wstęp do Pythona: podobieństwa i różnice w porównaniu do języka C,
 - środowisko programistyczne,
 - interpretacja a kompilacja, zarządzanie pamięcią (brak wskaźników, automatyczne zarządzanie pamięcią).
2. Typy danych, operatory i podstawowe struktury
 - typy danych: liczby, napisy,
 - operatory arytmetyczne, logiczne i porównania,
 - różnice w zarządzaniu typami danych w Pythonie i C,
 - podstawowe struktury danych: listy, krotki, słowniki, zbiory.
3. Instrukcje sterujące i pętle
 - instrukcje warunkowe i operator warunkowy,
 - pętle for, while – różnice w stosunku do języka C,
 - instrukcja match case.
4. Funkcje i moduły
 - tworzenie funkcji, argumenty i wartości zwracane,
 - typy parametrów: domyślne, opcjonalne, *args, **kwargs,
 - funkcje lambda i ich zastosowania,
 - rekurencja,
 - importowanie i tworzenie modułów,
 - praca z biblioteką standardową (np. os, sys, random, math, collections),
 - przykłady użycia wybranych pakietów zewnętrznych.
5. Struktury danych i techniki programowania w Pythonie
 - listy składane i ich zastosowanie,
 - słowniki i zbiory składane,
 - segmentowanie sekwencji (listy, krotki, napisy),
 - funkcje map(), filter(), reduce() i ich zastosowania,
 - generatory i wyrażenia generatorowe,
 - wydajność struktur danych w Pythonie (porównanie list, zbiorów i słowników).
6. Obsługa wyjątków i błędów
 - podstawowe konstrukcje try, except, finally,
 - proste przykłady obsługi błędów wykonania,
 - różnice między obsługą błędów w Pythonie i w języku C.
7. Podstawy pracy z plikami i serializacja danych
 - otwieranie, odczyt i zapis plików tekstowych,
 - różne tryby pracy z plikami,
 - serializacja danych w Pythonie (np. moduł pickle, format JSON),
 - porównanie z obsługą plików w języku C.

Wykaz literatury podstawowej

Wskazane przez prowadzącego rozdziały:

1. B. Slatkin, *Efektywny Python. 125 sposobów na lepszy kod*. Wydanie III, Helion, Gliwice 2025;
2. E. Matthes, *Python. Instrukcje dla programisty. Wydanie III*, Helion, Gliwice 2023;
3. A. Martelli, A. Martelli Ravenscroft, S. Holden, P. McGuire, *Python w pigułce. Podręczny przewodnik po wersjach 3.10 i 3.11*, Promise, Warszawa 2023;
4. D. Beazley, *Python. Zwięzłe kompendium dla programisty*, Helion, Gliwice 2023;
5. A. Sweigart, *Wielka księga małych projektów w Pythonie. 81 łatwych praktycznych programów*, Helion, Gliwice 2022;
6. Ch. Mayer, *Kod Pythona w jednym wierszu. Jak profesjonaliści piszą programy doskonałe*, Helion Gliwice 2021.

Wykaz literatury uzupełniającej

1. A. Sweigart, *Rekurencyjna książka o rekurencji. Zostań mistrzem rozmów kwalifikacyjnych poświęconych językom Python i JavaScript*, Helion, Gliwice 2023;
2. P. Wróblewski, *Python dla testera*, Helion, Gliwice 2021;
3. L. Vaughan, *Python mniej poważnie*, Wydawnictwo Naukowe PWN, Warszawa 2020;
4. P. J. Deitel, H. Deitel, *Python dla programistów. Big Data i AI. Studia przypadków*, Helion, 2020;
5. D. Kopec, *Klasyczne problemy informatyki w Pythonie*, Wydawnictwo Naukowe PWN, Warszawa 2020;
6. M. Lutz, *Python. Leksykon kieszonkowy. Wydanie V*, Helion Gliwice 2014;
7. 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	10
	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	55
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	0
	Przygotowanie do egzaminu/zaliczenia	0
Ogółem bilans czasu pracy		100
Liczba punktów ECTS w zależności od przyjętego przelicznika		4

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

Liczba godzin w kontakcie z prowadzącymi	Wykład	6
	Konwersatorium (ćwiczenia, laboratorium itd.)	20
	Pozostałe godziny kontaktu studenta z prowadzącym	4
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	70
	Przygotowanie krótkiej pracy pisemnej lub referatu po zapoznaniu się z niezbędną literaturą przedmiotu	0
	Przygotowanie projektu lub prezentacji na podany temat (praca w grupie)	0
	Przygotowanie do egzaminu/zaliczenia	0
Ogółem bilans czasu pracy		100
Liczba punktów ECTS w zależności od przyjętego przelicznika		4