

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 inż. Agnieszka Kańka
Punktacja ECTS*	3	

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>	K_W06
	<p>W06: zna mechanizmy obsługi wyjątków, podstawowe metody diagnozowania i analizowania błędów w programach w języku Python, w tym wykorzystanie narzędzi debugowania, oraz rozumie zasady pracy z plikami i serializacji danych.</p>	K_W06 K_W11
Umiejętności	Efekt uczenia się	Odniesienie do efektów kierunkowych
	Po zakończeniu kursu student:	
	U01: potrafi pisać i uruchamiać proste programy w Pythonie, stosując poprawną składnię i konwencje języka.	K_U05
	U02: umie korzystać z podstawowych struktur danych i instrukcji sterujących do rozwiązywania prostych problemów.	K_U04 K_U05
	U03: potrafi definiować i wykorzystywać funkcje, importować moduły oraz korzystać z wybranych bibliotek standardowych.	K_U04 K_U05
	U04: umie stosować techniki programistyczne charakterystyczne dla Pythona, takie jak listy składane, generatory czy funkcje lambda.	K_U05
	U05: potrafi diagnozować i usuwać błędy w programach w języku Python, interpretować komunikaty błędów, stosować mechanizm wyjątków do obsługi sytuacji błędnych oraz wykorzystywać podstawowe narzędzia debugowania.	K_U05 K_U12
	U06: umie zapisywać i odczytywać dane z plików tekstowych oraz stosować podstawowe mechanizmy serializacji, takie jak JSON czy pickle.	K_U04 K_U05
	U07: potrafi samodzielnie doskonalić swoje kompetencje w środowisku e-learningowym.	K_U17

Kompetencje społeczne	Efekt uczenia się	Odniesienie do efektów kierunkowych
	Po zakończeniu kursu student:	
	K01: potrafi współpracować w zespole przy realizacji prostych projektów programistycznych.	K_K01
	K02: dostrzega znaczenie czytelności, przejrzystości i jakości kodu źródłowego.	K_K04
	K03: rozumie konieczność rozwijania swoich umiejętności programistycznych i poszerzania wiedzy o bardziej zaawansowane elementy języka Python.	K_K01

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	6					20					

Opis metod prowadzenia zajęć

Wykłady mają charakter teoretyczno-praktyczny 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. Część wykładów ma formę demonstracyjną - omawiane zagadnienia są ilustrowane analizą kodu, prezentacją działania programów oraz pokazem technik debugowania i diagnostyki błędów.

Ćwiczenia mają charakter praktyczny i laboratoryjny. 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. Istotnym elementem zajęć jest rozwijanie umiejętności analizy błędów, pracy z debuggerem oraz świadomego testowania i weryfikacji poprawności programów.

W trakcie semestru studenci przystępują do dwóch kolokwii obejmujących zagadnienia praktyczne i teoretyczne realizowane na ćwiczeniach.

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					
W06	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					
U06	x				x	x		x					
U07	x					x							
K01								x					
K02								x					
K03	x							x					

Kryteria oceny	<p>Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna. Warunkiem jej przyznania jest spełnienie minimalnych wymagań określonych dla poszczególnych form zajęć.</p> <p>Ocena dobra (4.0) przysługuje studentowi, który wykazuje się dobrą znajomością języka Python, potrafi pisać czytelny i poprawny kod, stosuje właściwe struktury danych oraz potrafi diagnozować i usuwać typowe błędy w programach.</p> <p>Ocena bardzo dobra (5.0) przysługuje studentowi, który swobodnie posługuje się językiem Python, potrafi pisać przejrzysty, dobrze zorganizowany i efektywny kod, świadomie wykorzystuje zaawansowane mechanizmy języka oraz sprawnie analizuje i debuguje złożone problemy programistyczne.</p> <p>Warunkiem zaliczenia ćwiczeń jest uzyskanie co najmniej 50% punktów z każdego z dwóch kolokwii przeprowadzanych w trakcie semestru. Ocena z ćwiczeń stanowi średnią arytmetyczną ocen uzyskanych z obu kolokwii.</p> <p>Pod warunkiem zaliczenia obu kolokwii prowadzący może uwzględnić aktywność studenta na zajęciach, w szczególności systematyczną pracę, rozwiązywanie zadań oraz udział w dyskusji, jako czynnik wpływający na podwyższenie oceny końcowej z ćwiczeń.</p> <p>Ocena końcowa z ćwiczeń ustalana jest na podstawie średniej ocen z kolokwii, z możliwością uwzględnienia aktywności studenta.</p> <p>Warunkiem zaliczenia wykładu jest obowiązkowa obecność na zajęciach wykładowych oraz ukończenie kursu Python Essentials 1 w ramach Cisco Networking Academy i uzyskanie certyfikatu potwierdzającego jego zaliczenie.</p>
----------------	---

Uwagi	
-------	--

Treści merytoryczne (wykaz tematów)

- 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).
- 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.
- Instrukcje sterujące i pętle
 - instrukcje warunkowe i operator warunkowy,
 - pętle `for`, `while` - różnice w stosunku do języka C,
 - instrukcja `match case`.
- 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
- rodzaje błędów w języku Python: składniowe, wykonania i logiczne,
 - interpretacja komunikatów błędów oraz analiza traceback,
 - konstrukcje `try`, `except`, `else`, `finally` oraz zgłaszanie wyjątków (`raise`),
 - podstawowe zasady walidacji danych i dobre praktyki obsługi błędów,
 - wprowadzenie do debugowania programów oraz wykorzystanie debuggera w środowisku programistycznym.
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. M. Lutz, *Python. Wprowadzenie. Wydanie VI*, Helion Gliwice 2025.
2. B. Slatkin, *Efektywny Python. 125 sposobów na lepszy kod. Wydanie III*, Helion, Gliwice 2025;
3. E. Matthes, *Python. Instrukcje dla programisty. Wydanie III*, Helion, Gliwice 2023;
4. A. Martelli, A. Martelli Ravenscroft, S. Holden, P. McGuire, *Python w pigułce. Podręczny przewodnik po wersjach 3.10 i 3.11*, Promise, Warszawa 2023;
5. D. Beazley, *Python. Zwięzłe kompendium dla programisty*, Helion, Gliwice 2023;
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. A. Sweigart, *Wielka księga małych projektów w Pythonie. 81 łatwych praktycznych programów*, Helion, Gliwice 2022;
3. P. Wróblewski, *Python dla testera*, Helion, Gliwice 2021;
4. L. Vaughan, *Python mniej poważnie*, Wydawnictwo Naukowe PWN, Warszawa 2020;
5. P. J. Deitel, H. Deitel, *Python dla programistów. Big Data i AI. Studia przypadków*, Helion, 2020;
6. D. Kopec, *Klasyczne problemy informatyki w Pythonie*, Wydawnictwo Naukowe PWN, Warszawa 2020;
7. M. Lutz, *Python. Leksykon kieszonkowy. Wydanie V*, Helion Gliwice 2014

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	1
	Lektura w ramach przygotowania do zajęć	14

Liczba godzin pracy studenta bez kontaktu z prowadzącymi	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	15
Ogółem bilans czasu pracy		75
Liczba punktów ECTS w zależności od przyjętego przelicznika		3

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	1
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	23
	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	25
Ogółem bilans czasu pracy		75
Liczba punktów ECTS w zależności od przyjętego przelicznika		3