

## KARTA KURSU

Nazwa	<b>Programowanie Funkcyjne (Python)</b>
Nazwa w j. ang.	Functional programming (Python)

Koordynator	dr Roman Czapla	Zespół dydaktyczny
		dr Roman Czapla dr Iryna Artyschchuk
Punktacja ECTS*	3	

### Opis kursu (cele kształcenia)

Celem kursu jest wprowadzenie studentów do programowania w języku Python ze szczególnym uwzględnieniem paradygmatu programowania funkcyjnego. Studenci poznają podstawy składni i semantyki języka Python, a następnie nauczą się stosować podejście funkcyjne do rozwiązywania problemów obliczeniowych. W trakcie zajęć omawiane będą m.in. czyste funkcje, funkcje wyższego rzędu, funkcje anonimowe (lambda), rekurencja, operacje na kolekcjach przy użyciu funkcji map, filter i reduce, a także elementy programowania deklaratywnego i pracy z modułem functools. Kurs kładzie nacisk na praktyczne zastosowania: od prostych algorytmów po przetwarzanie danych, rozwijając zarówno umiejętności programistyczne, jak i myślenie abstrakcyjne.

### Warunki wstępne

Wiedza	Student powinien posiadać elementarną wiedzę matematyczną, obejmującą podstawowe pojęcia logiki, funkcji oraz prostych struktur zbiorów. Wymagana jest również ogólna orientacja w zakresie korzystania z technologii informatycznych.
Umiejętności	Oczekuje się umiejętności sprawnego obsługi komputera i systemu operacyjnego, a także zdolności analitycznego myślenia i rozwiązywania prostych problemów. Student powinien potrafić korzystać z literatury oraz materiałów dydaktycznych w języku polskim i angielskim.
Kursy	Nie są wymagane wcześniejsze kursy z programowania. Kurs nie zakłada znajomości języka Python ani innych języków programowania i ma charakter wprowadzający.

### Efekty uczenia się

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Wiedza	Po zakończeniu kursu student: <b>W01:</b> zna podstawowe elementy składni i semantyki języka Python oraz rozumie strukturę prostych programów.	K_W03
	<b>W02:</b> zna i rozumie podstawowe typy danych i kolekcje wbudowane w Pythonie oraz zasady ich stosowania.	K_W03
	<b>W03:</b> zna podstawowe koncepcje programowania funkcyjnego (funkcje wyższego rzędu, funkcje anonimowe, rekurencja, listy składane, wyrażenia generatorowe).	K_W03
	<b>W04:</b> rozumie różnice między paradygmatem imperatywnym a funkcyjnym oraz zna przykłady bibliotek wspierających styl funkcyjny w Pythonie.	K_W03

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Umiejętności	Po zakończeniu kursu student:	
	<b>U01:</b> potrafi tworzyć proste programy w Pythonie, stosując poprawną składnię i dobre praktyki.	K_U04
	<b>U02:</b> umie korzystać z podstawowych typów danych, kolekcji oraz konstrukcji sterujących w rozwiązywaniu problemów.	K_U04
	<b>U03:</b> potrafi definiować własne funkcje i stosować funkcje anonimowe oraz wyższego rzędu (map, filter, reduce).	K_U04
	<b>U04:</b> potrafi rozwiązywać problemy z użyciem podejścia funkcyjnego (rekurencja, operacje na kolekcjach, listy składane, wyrażenia generatorowe).	K_U04
	<b>U05:</b> umie łączyć elementy programowania proceduralnego i funkcyjnego w tworzeniu aplikacji w Pythonie.	K_U04
	Efekt uczenia się	Odniesienie do efektów kierunkowych
Kompetencje społeczne	Po zakończeniu kursu student:	
	<b>K01:</b> dostrzega znaczenie dobrych praktyk programistycznych i jakości kodu w pracy indywidualnej i zespołowej.	K_K02
	<b>K02:</b> potrafi współpracować z innymi nad rozwiązaniem problemów programistycznych, dzieląc się zadaniami i odpowiedzialnością.	K_K02
	<b>K03:</b> rozumie potrzebę ciągłego uczenia się i rozwijania umiejętności w obszarze programowania oraz nowych paradygmatów i technologii.	K_K01

### 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

Podczas zajęć wykładowych omawiane są podstawowe zagadnienia paradygmatu funkcyjnego w kontekście języka Python, ilustrowane przykładami kodu. Zajęcia ćwiczeniowe mają charakter praktyczny - studenci samodzielnie rozwiązują zadania programistyczne, analizują przykładowe programy, a następnie prezentują i omawiają własne rozwiązania w grupie. Istotnym elementem zajęć jest także dyskusja nad różnymi podejściami do rozwiązania problemu oraz porównywanie efektywności i czytelności kodu.

## 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					
W02					x			x					
W03					x			x					
W04					x			x					
U01					x			x					
U02					x			x					
U03					x			x					
U04					x			x					
U05					x			x					
K01								x					
K02								x					
K02								x					

Kryteria oceny	<p>Osiągnięcie efektów kształcenia podanych powyżej uprawnia studentów do uzyskania oceny nie wyższej niż dostateczna.</p> <p>Zaliczenie przedmiotu odbywa się na podstawie dwóch kolokwii sprawdzających znajomość zagadnień teoretycznych i praktycznych z zakresu programowania w języku Python, ze szczególnym uwzględnieniem elementów paradygmatu funkcyjnego.</p> <p>Warunkiem uzyskania zaliczenia jest:</p> <ul style="list-style-type: none"> <li>• zaliczenie obu kolokwii (wymagana ocena pozytywna z każdego),</li> <li>• aktywne uczestnictwo w zajęciach ćwiczeniowych.</li> </ul> <p>Ocena końcowa wyliczana jest jako średnia z ocen częściowych, z możliwością podniesienia o pół stopnia w przypadku wyróżniającej się aktywności studenta.</p> <p>Zaliczenie na ocenę dobrą lub bardzo dobrą otrzymuje student, który spełnia warunki oceny dostatecznej, a ponadto:</p> <ul style="list-style-type: none"> <li>• rozwiązuje zadania programistyczne w sposób poprawny, efektywny i zgodny z dobrymi praktykami,</li> <li>• potrafi zastosować funkcje wyższego rzędu, rekurencję, listy składane oraz wyrażenia generatorowe w praktycznych problemach,</li> <li>• wykazuje się umiejętnością krytycznego porównania paradygmatu imperatywnego i funkcyjnego,</li> <li>• potrafi świadomie dobrać narzędzia i techniki programowania funkcyjnego do określonego zagadnienia.</li> </ul> <p><b>Obecność na wykładach jest warunkiem zaliczenia tej części kursu.</b></p>

## Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do pracy z językiem Python
  - a. Środowisko programistyczne PyCharm
  - b. Interpreter języka Python
  - c. Struktura programu w języku Python
    - składnia i konwencje leksykalne
    - komentarze
    - zmienne i obiekty
  - d. Obsługa standardowego wyjścia i wejścia
  - e. Podstawowe funkcje wbudowane
2. Podstawy programowania proceduralnego w języku Python
  - a. Podstawowe typy danych
    - całkowite i zmiennoprzecinkowe rodzaje danych
    - ciągi tekstowe
  - b. Kontenery wbudowane
    - krotki i listy
    - zbiory i słowniki
    - iteracja i kopiowanie kolekcji
  - c. Struktury kontrolne i funkcje
    - konstrukcje rozgałęziające
    - pętle
    - własne funkcje
  - d. Wybrane moduły i pakiety
3. Wprowadzenie do programowania funkcyjnego w języku Python
  - a. Historia i podstawowe koncepcje
  - b. Programowanie imperatywne vs. programowanie funkcyjne
  - c. Funkcje lambda, map(), filter(), reduce()
  - d. Funkcje jako obiekty pierwszej klasy w języku Python
  - e. Rekurencja i przykłady algorytmów rekurencyjnych
  - f. Typy niemutowalne
  - g. Listy składane i wyrażenia generatora
  - h. Biblioteki wspierające programowanie funkcyjne w języku Python – moduł functools

## 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. S. F. Lott, *Programowanie funkcyjne w Pythonie. Jak pisać zwięzły, wydajny i ekspresywny kod*. 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. A. Sweigart, *Rekurencyjna książka o rekurencji. Zostań mistrzem rozmów kwalifikacyjnych poświęconych językom Python i JavaScript*, Helion, Gliwice 2023.

## Wykaz literatury uzupełniającej

1. Ch. Mayer, *Kod Pythona w jednym wierszu. Jak profesjonalisci piszą programy doskonałe*, Helion Gliwice 2021;
2. P. J. Deitel, H. Deitel, *Python dla programistów. Big Data i AI. Studia przypadków*, Helion, Gliwice 2020;
3. D. Kopec, *Klasyczne problemy informatyki w Pythonie*, PWN, Warszawa 2020;
4. M. Lutz, *Python. Leksykon kieszonkowy*. Wydanie V, Helion, Gliwice 2014;
5. 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	10
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	10
	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	14
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	15
	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	20
Ogółem bilans czasu pracy		75
Liczba punktów ECTS w zależności od przyjętego przelicznika		3