Politechnika Krakowska im. Tadeusza Kościuszki

Karta przedmiotu

obowiązuje studentów rozpoczynających studia w roku akademickim 2017/2018

Wydział Fizyki, Matematyki i Informatyki

Kierunek studiów: Informatyka

Profil: Ogólnoakademicki

Forma sudiów: stacjonarne

Kod kierunku: I

Stopień studiów: I

Specjalności: Brak specjalności

1 Informacje o przedmiocie

Nazwa przedmiotu	Języki symboliczne
Nazwa przedmiotu w języku angielskim	
Kod przedmiotu	WFMiI I oIS D14 17/18
Kategoria przedmiotu	Przedmioty specjalnościowe
Liczba punktów ECTS	4.00
Semestry	4

2 Rodzaj zajęć, liczba godzin w planie studiów

Semestr	Wykład	Ćwiczenia	Laboratorium	Laboratorium komputerowe	Seminarium	Projekt
4	30	0	30	0	0	0

3 Cele przedmiotu

Cel 1 Poznanie paradygmatu programowania imperatywnego i jego różnicy w stosunku do deklaratywnego podejścia matematycznego. Poznanie składni języka LISP jako przykładu języka symbolicznego pozwalającego na implementowania przetwarzania symbolicznego. Językiem ten jest zarazem językiem ogólnego przeznaczenia umożliwiającym programowanie w stylu proceduralnym, funkcyjnym i obiektowym.

Cel 2 Nabycie umiejętności projektowania i implementacji programów napisanych w dialekcie Scheme. Projektowanie i implementacja programów demonstrujących koncepcje i użycie mechanizmów rekurencji, procesów iteracyjnych, procedur prostych i złożonych oraz abstrakcji danych.

Cel 3 Poznanie składni języka Python. Poznanie podstawowych typów danych i instrukcje sterujących używanych w języku Python.

Cel 4 Nabycie umiejętności projektowania i implementacji programów w języku Python. Umiejętność wykorzystania zaawansowanych typów danych języka Python. Umiejętność wykorzystania języka Python w zakresie programowania w modelu obiektowym.

4 Wymagania wstępne w zakresie wiedzy, umiejętności i innych kompetencji

1 Zaliczone przedmioty: Wstęp do programowania oraz Algorytmy i struktury danych.

5 Efekty kształcenia

EK1 Wiedza Student rozumie pojęcie programowania imperatywnego i jego różnicę w stosunku do deklaratywnego podejścia matematycznego. Zna podstawy składni języka LISP. Student rozumie na czym polega zastosowanie mechanizmów typowych dla języka w celu budowy abstrakcji na poziomie procesu obliczeniowego, złożonych procedur, danych.

EK2 Wiedza Student zna składnię języka Python oraz typowych dla niego typów danych. Umie wykorzystać możliwości języka Python w zakresie programowania w modelu proceduralnym i obiektowym.

EK3 Umiejętności Student umie skonfigurować środowisko programistyczne w dialekcie Scheme oraz potrafi napisać i uruchomić własny program w języku LISP rozwiązujący postawiony przed nim problem. Student potrafi zaprojektować i zaimplementować program wykorzystujący mechanizmy rekurencji i procesów iteracyjnych oraz potrafi budować programy na odpowiednim poziomie abstrakcji.

EK4 Umiejętności Student potrafi pisać programy użytkowe w języku Python oraz odpowiednio dobrać model programowania i struktury danych. Umie skorzystać z dokumentacji języka Python.

6 Treści programowe

Wykład		
Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
W1	Informacje o przedmiocie. Podstawowe informacje o języku Lisp, jego historii i dialektach. Przedstawienie idei programowania imperatywnego w stosunku do podejścia deklaratywnego. Elementy programowania. Nazwy i środowisko programistyczne.	2
W2	Proste wyrażenia i operacje na prostych typach danych. Podstawy składni. Procedury złożone. Podstawieniowy model stosowania procedur. Stosowana a normalna kolejność obliczania. Wyrażenia warunkowe i predykaty. Procedury jako abstrakcyjne, czarne skrzynki.	2
W3	Formułowanie abstrakcji za pomocą procedur wyższych rzędów. Tworzenie procedur za pomocą lambda-abstrakcji. Tworzenie zmiennych lokalnych za pomocą let. Procedury jako metody ogólne. Procedury jako wyniki.	2

Wykład		
Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
W4	Budowanie abstrakcji za pomocą danych. Bariery abstrakcji. Pojęcie danych. Dane hierarchiczne i własności domknięcia. Reprezentowanie ciągów, list, struktur listowych. Operacje na listach, odwzorowywanie list. Wyrażenia symboliczne i ich wartościowanie.	2
W5	Hierarchiczne struktury danych, odwzorowywanie drzew. Ciągi jako interfejsy konwencjonalne. Operacje na ciągach. Odwzorowania zagnieżdżone. Zbiory jako listy nieuporządkowane i uporządkowane. Zbiory jako drzewa binarne. Wyszukiwanie informacji w zbiorach. Systemy z operacjami ogólnymi. Ogólne operacje arytmetyczne. Łączenie danych różnych typów. Algebra symboliczna.	2
W6	Wprowadzenie do języka Python. Wersje języka. Opis interpretera. Tryby pracy interpretera. Raportowanie błędów. Automatyczne uruchamianie skryptów języka Python. Kodowanie znaków, Konwencje dotyczące komentarzy.	2
W7	Podstawowe typy danych języka Python. Podstawowe operacje na łańcuchach znakowych, listach i słownikach.	3
W8	Definicje funkcji i argumentów, wyrażenia warunkowe, wyrażenia lambda, pętle. Iterowanie po indeksach sekwencji. Domyślne argumenty funkcji. Napisy dokumentujące. Konwencje kodowania i stylu.	3
W9	Rozbudowane struktury danych stos, kolejki. Listy zagnieżdżone, krotki, zbiory, słowniki. Mutowalne i niemutowalne typy danych. Moduły i pakiety oraz ich import. Kompilacja w języku Python. Odwołania relatywne i absolutne.	2
W10	Operacje wejścia i wyjścia języka Python. Konwersja danych do stringów. Formatowanie napisów. Czytanie i pisanie do plików. Tworzenia obiektów typu: persistent.	2
W11	Zgłaszanie i obsługa błędów i wyjątków. Wyjątki definiowane przez użytkownika. Predefiniowane akcje typu clean up.	2
W12	Przestrzenie nazw. Instrukcje nonlocal i global. Zasięg zmiennych. Programowanie obiektowe w Pythonie. Definicja klas. Obiekty klas i metod. Zmienne obiektów (instancji) i zmienne klas. Atrybuty klas i obiektów.	2
W13	Dziedziczenie klas. Zmienne prywatne. Wyjątki jako klasy. Iteratory i generatory.	2
W14	Biblioteka standardowa. Operacje na wzorcach. Obliczenia matematyczne. Kontrola jakości pisanego oprogramowania, testy.	2

Laboratorium		
Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin

Laboratorium		
Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
L1	Konfiguracja środowiska programistycznego dla dialektu języka LISP Schame. Obliczanie wartości prostych wyrażeń. Kombinacje wyrażeń. Operatory i argumenty, notacja prefiksowa. Zagnieżdżanie kombinacji. Formatowanie kodu źródłowego. Zmienne i ich wartości. Nazwy i środowisko. Definiowanie procedur, formy specjalne. Parametry formalne w definicji procedury.	2
L2	Podstawieniowy model stosowania procedur. Stosowana a normalna kolejność obliczeń. Wyrażenia warunkowe i predykaty, analiza przypadków. Predykaty, klauzule i następniki. Wyrażenia cond, if, else, relacje pierwotne oraz operacje logiczne takie jak: and, or, not. Implementacja pierwiastkowania metodą Newtona. Obliczanie pierwiastków sześciennych. Implementacje programów z wykorzystaniem zmiennych lokalnych. Parametry formalne jako zmienne związane. Zakres zmiennych związanych deklarowanych jako parametry formalne procedury. Wiązanie składniowe.	2
L3	Obliczanie n! i przykłady rekursji. Implementacja n! jako liniowy proces rekurencyjny oraz liniowy proces iteracyjny. Rekursja ogonowa. Obliczanie liczb Fibonacciego jako przykład rekursji rozgałęziającej się (drzewiastej). Implementacja operacji potęgowania, poszukiwanie największych wspólnych dzielników oraz licz pierwszych.	2
L4	Procedury operujące na procedurach, procedury wyższego rzędu. Użycie procedur jako argumentów na przykładzie implementacji sumowania szeregów. Lambda-abstrakcje. Tworzenie zmiennych lokalnych za pomocą let. Implementacja zastosowania procedur jako metod ogólnych poprzez znajdowanie pierwiastków równań przez bisekcję. Tworzenie procedur których wyniki same są procedurami. Abstrakcja danych łączenie obiektów danych w dane złożone. Implementacja przykładowych operacji arytmetycznych na liczbach wymiernych. Pary jako - przykład struktur złożonych.	2
L5	Tworzenie struktur hierarchicznych - struktur złożonych z części, które same są złożone z części. Budowanie ciągów i drzew za pomocą par. Implementacja list oraz podstawowych na nich operacji. Implementacja drzew. Operowanie na danych symbolicznych. Implementacja zbiorów, tworzenie drzew Huffmana.	2
L6	Podstawy składni języka Python. Podstawowe struktury danych.	2
L7	Podstawowe operacje na łańcuchach znakowych, listach i słownikach.	2
L8	. Definicje funkcji i argumentów, podstawowe funkcje wbudowane, wyrażenia warunkowe, wyrażenia lambda, pętle. Iterowanie po indeksach sekwencji. Domyślne argumenty funkcji.	2
L9	Tworzenie własnych modułów i pakietów. Zasady zasięgu zmiennych. Implementacja rozbudowanych struktur danych. Listy zagnieżdżone, krotki, zbiory, słowniki.	2
L10	Operacje na plikach. Formatowanie napisów. Tworzenie i wykorzystanie iteratorów i generatorów.	2

Laboratorium		
Lp	Tematyka zajęć Opis szczegółowy bloków tematycznych	Liczba godzin
L11	Obsługa błędów i wyjątków. Wyjątki definiowane przez użytkownika. Predefiniowane akcje typu clean up.	3
L12	Programowanie obiektowe, hierarchia klas.	3
L13	Obiekty klas i metod. Zmienne obiektów (instancji) i zmienne klas. Atrybuty klas i obiektów. Dziedziczenie klas. Zmienne prywatne. Wyjątki jako klasy.	4

7 Narzędzia dydaktyczne

N1 Wykłady

N2 Ćwiczenia laboratoryjne

8 Obciążenie pracą studenta

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe z nauczycielem akademickim, w tym:	
Godziny wynikające z planu studiów	60
Konsultacje przedmiotowe	0
Egzaminy i zaliczenia w sesji	0
Godziny bez udziału nauczyciela akademickiego wynikające z nakładu pracy studenta, w tym:	
Przygotowanie się do zajęć, w tym studiowanie zalecanej literatury	40
Opracowanie wyników	10
Przygotowanie raportu, projektu, prezentacji, dyskusji	10
Sumaryczna liczba godzin dla przedmiotu wynikająca z całego nakładu pracy studenta	**120**
Sumaryczna liczba punktów ECTS dla przedmiotu	4.00

9 Sposoby oceny

Ocena formująca

F1 Kolokwium

Ocena podsumowująca

P1 Średnia ważona ocen formujących

Kryteria oceny

Efekt kształcenia 1	
Na ocenę 3.0	Student zna pojęcie programowania imperatywnego i jego różnicę w stosunku do deklaratywnego podejścia matematycznego. Student wie jak definiować procedury złożone. Umie przedstawić stosowaną i normalną kolejność obliczania danej procedury.
Na ocenę 3.5	Student rozumie pojęcie programowania imperatywnego. Zna podstawy składni języka LISP. Student wie jak definiować procedury złożone. Umie przedstawić stosowaną i normalną kolejność obliczania danej procedury. Potrafi formułować abstrakcje za pomocą procedur wyższego rzędu.
Na ocenę 4.0	Student rozumie pojęcie programowania imperatywnego. Zna na dobrym poziomie składnię języka LISP. Student wie jak definiować procedury złożone. Umie przedstawić stosowaną i normalną kolejność obliczania danej procedury. Potrafi formułować abstrakcje za pomocą procedur wyższego rzędu z wykorzystaniem lambda-abstrakcji.
Na ocenę 4.5	Student rozumie pojęcie programowania imperatywnego. Zna na dobrym poziomie składnię języka LISP. Wie jakie konstrukcje językowe są adekwatne do rozważanych problemów. Oprócz wymagań takich jak na ocenę 4.0 student potrafi dobrać odpowiednie struktury danych do rozważnego problemu.
Na ocenę 5.0	Student rozumie pojęcie programowania imperatywnego. Zna bardzo dobrze składnię języka LISP. Wie jakie konstrukcje językowe są adekwatne do rozważanych problemów i potrafi projektować systemy na nich bazujące. Student wie jakie mechanizmy abstrakcji są adekwatne do rozważanego problemu oraz potrafi wskazać mechanizmy i odpowiednie struktury danych do ich realizacji typowe dla języka LISP.
Efekt kształcenia 2	
Na ocenę 3.0	Student zna podstawowe konstrukcje języka Python
Na ocenę 3.5	Student wie jak definiować procedury złożone. Umie przedstawić stosowaną i normalną kolejność obliczania danej procedury. Potrafi formułować abstrakcje za pomocą procedur wyższego rzędu.
Na ocenę 4.0	Student umiejętnie wykorzystuje poznane elementy języka w programowaniu proceduralnym
Na ocenę 4.5	Student umiejętnie wykorzystuje poznane elementy języka w programowaniu obiektowym
Na ocenę 5.0	Student umie tworzyć skomplikowane struktury danych oraz odpowiednio wykorzystuje je w programowaniu proceduralnym i obiektowym
Efekt kształcenia 3	

Na ocenę 3.0	Student potrafi skonfigurować środowisko programistyczne dialektu Scheme. Potrafi zrozumieć i zapisać program na podstawie lektury kodu źródeł. Student potrafi rozwiązać problem z wykorzystaniem procedur rekurencyjnych.
Na ocenę 3.5	Potrafi napisać program w języku LISP realizujący przyjęte założenia. Potrafi przedstawić rozwiązanie zadanego problemu przy pomocy rekursji liniowej i iteracyjnej.
Na ocenę 4.0	Potra przeprowadzić analizę pozwalającą ustalić wymagania jakie stawiamy programowi. Potrafi zaimplementować procedury wyższego rzędu z użyciem lambda-abstrakcji. Potrafi wykorzystywać procedury jako wyniki.
Na ocenę 4.5	Potrafi uogólnić rozwiązanie w celu dostosowania go do innych typów danych. Potrafi wykorzystywać struktury listowe, drzewiaste oraz zbiory oraz dokonywać na nich odpowiednich operacji.
Na ocenę 5.0	Potrafi przeanalizować zaproponowane rozwiązanie w kontekście jego zalet w stosunku do innych języków programowania. Potrafi łączyć różne typy danych oraz modelować z użyciem danych modyfikowalnych.
Efekt kształcenia 4	
Na ocenę 3.0	Student umiejętnie korzysta z dokumentacji języka Python oraz manuala.
Na ocenę 3.5	Student umie wykorzystać natywne struktury danych języka Python
Na ocenę 4.0	Student wie w jaki sposób parametry są przekazywane do funkcji. Wie jak zwrócić wynik działania funkcji. Potrafi tworzyć i wywoływać funkcję o zmiennej liczbie parametrów. Zna zasady zasięgu zmiennych
Na ocenę 4.5	Student umiejętnie tworzy własne moduły oraz pakiety.
Na ocenę 5.0	Student potrafi umiejętnie tworzyć hierarchie klas.

10 Macierz realizacji przedmiotu

Efekt kształcenia	Odniesienie danego efektu do szczegółowych efektów zdefiniowanych dla programu	Cele przedmiotu	Treści programowe	Narzędzia dydaktyczne	Sposoby oceny
EK1		Cel 1	W1 W2 W3 L1 L2	N1 N2	F1
EK2		Cel 3 Cel 4	W1 W2 W3 W4 W5 L1 L2 L3 L4 L5	N1 N2	F1

Efekt kształcenia	Odniesienie danego efektu do szczegółowych efektów zdefiniowanych dla programu	Cele przedmiotu	Treści programowe	Narzędzia dydaktyczne	Sposoby oceny
EK3		Cel 2 Cel 3 Cel 4	W1 W2 W3 W4 W5 W6 L1 L2 L3 L4 L5	N1 N2	F1 P1
EK4		Cel 4	W1 W2 W3 W4 W5 W6 L1 L2 L3 L4 L5 L6	N1 N2	F1 P1

11 Wykaz literatury

Literatura podstawowa

[1] Harold Abelson and Gerald Jay Sussman Sussman — *Struktura i interpretacja programów komputerowych*, Warszawa, 2002, WNT

[2] Mark Pilgrim — *Dive into Python*, , 2004,

Literatura uzupełniająca

[1] Matthias Felleisen Robert Bruce Findler Matthew Flatt Shriram Krishnamurthi — *How to Design Programs*, London, 2001, The MIT Press

[2] Brian Harvey Matthew Wright — *Simply Scheme*, London, 1999, The MIT Press

[3] [Daniel P. Friedman, Matthias Felleisen — *The Little Schemer*, London, 1995, The MIT Press

[4] Mark Lutz — *Learning Python*, Beijing, 2013, OReilly Media, Inc.

12 Informacje o nauczycielach akademickich

Osoba odpowiedzialna za kartę

dr inż. Tomasz Gąciarz (kontakt: tga@pk.edu.pl)

Osoby prowadzące przedmiot

1 dr inż. Tomasz Gąciarz (kontakt: tga@pk.edu.pl)

13 Zatwierdzenie karty przedmiotu do realizacji

(miejscowość, data) (odpowiedzialny za przedmiot) (dziekan)

Przyjmuję do realizacji (data i podpisy osób prowadzących przedmiot)

. . . . . .