**Nazwa przedmiotu:**

Podstawy informatyki i programowania

**Koordynator przedmiotu:**

Rajmund Kożuszek

**Status przedmiotu:**

Obowiązkowy

**Poziom kształcenia:**

Studia I stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Przedmioty techniczne

**Kod przedmiotu:**

PIPR

**Semestr nominalny:**

1 / rok ak. 2021/2022

**Liczba punktów ECTS:**

6

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

1. liczba godzin kontaktowych – 61 godz., w tym
 obecność na wykładach: 30 godz.,
 obecność na zajęciach laboratoryjnych wraz z konsultacjami projektu: 30 godz.,
 udział w konsultacjach związanych z realizacją przedmiotu: 1 godz.
2. praca własna studenta – 89 godz., w tym
 analiza literatury i materiałów wykładowych związana z przygotowaniem do kolejnych wykładów, wskazanie przykładów do wykonania: 20 godz.
 przygotowanie do laboratorium i realizacji projektu: 14 godz.
 praca na laboratorium i wykonanie zadania projektowego: 45 godz.
 przygotowanie do kolokwiów: 10 godz.

Łączny nakład pracy studenta wynosi 150 godz., co odpowiada 6 pkt. ECTS.

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich: 2.25 pkt. ECTS, co odpowiada 61 godz. kontaktowym.

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym: 3 pkt. ECTS, co odpowiada 75 godz. realizacji ćwiczeń

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład:  | 30h |
| Ćwiczenia:  | 0h |
| Laboratorium:  | 450h |
| Projekt:  | 225h |
| Lekcje komputerowe:  | 0h |

**Wymagania wstępne:**

Bez wymagań wstępnych

**Limit liczby studentów:**

210

**Cel przedmiotu:**

Celem przedmiotu jest przekazanie podstawowej wiedzy dotyczącej podstaw informatyki programowania. Obejmuje ona: podstawy architektury systemów komputerowych, reprezentację bitową liczb i znaków, algorytmizację problemów, pojęcie złożoności obliczeniowej, zasady programowania strukturalnego, podstawowe struktury danych, wyjątki. Wprowadzone zostanie pojęcie niezawodności wraz z objaśnieniem stosowania testów jednostkowych i systemowych. Przedstawione zostaną podstawowe narzędzia programistyczne: środowisko zintegrowane, debugger, system kontroli wersji, generator dokumentacji. Praktyczna nauka podstaw programowania prowadzona będzie w oparciu o język Python. Studenci będą utrwalać wiedzę teoretyczną i zdobywać umiejętności praktyczne dzięki realizacji ciekawych zadań i projektów programistycznych.

**Treści kształcenia:**

WYKŁADY:

1. Wprowadzenie (2 godz.)
Informacje o przedmiocie. Regulamin przedmiotu. Podstawowe pojęcia: program, algorytm. Pseudokod i schemat blokowy. Stworzenie pierwszego programu komputerowego. Debugger. Zintegrowane środowisko uruchomieniowe.

2. Podstawy (2 godz.)
Podstawy architektury systemów komputerowych. Języki maszynowe, assemblery, języki wysokopoziomowe, kompilatory i interpretery. Podstawowe typy danych i ich reprezentacje bitowe, w tym reprezentacja znaków i łańcuchów znakowych. Algorytmizacja problemów, pojęcie złożoności obliczeniowej. Specyfikacja wymagań, projektowanie, tworzenie zadań.

3. Podstawowe elementy programu (2 godz.)
Typy wbudowane, operatory. Podejmowanie decyzji, powtarzanie. Funkcje. Moduły i pakiety.

4. Paradygmaty programowania (2 godz.)
Paradygmaty programowania wspierane przez Pythona: zasady programowania strukturalnego, podstawowe struktury danych; wstęp do programowania obiektowego: klasy i instancje.

5. Struktury danych i algorytmy (4 godz.)
Tworzenie własnych struktur danych takich jak listy i drzewa. Podstawowe algorytmy: sortowanie, przeszukiwanie binarne.

6. Pobieranie i zwracanie danych (4 godz.)
Pobieranie i wypisywanie danych na strumienie standardowe. Praca z plikami (samodzielnie i z użyciem bibliotek: csv, json, xml). Wyrażenia regularne. Sprawdzanie danych wejściowych i formatu plików. Argumenty wywołania programu.

7. Narzędzia (2 godz.)
Środowisko zintegrowane, debugger, profiler, linter, system kontroli wersji, generator dokumentacji. Styl kodowania, PEP.

8. Niezawodność oprogramowania (2 godz.)
Pojęcie niezawodności oprogramowania, testy jednostkowe i systemowe. Sprawdzanie pokrycia testami. Wyjątki.

9. Reużywalność kodu (2 godz.)
O potrzebie tworzenia reużywalnego kodu oraz o wyższości dobrze przetestowanych modułów nad własnymi implementacjami. Przegląd możliwości biblioteki standardowej i wybranych bibliotek zewnętrznych. Generowanie liczb pseudolosowych.

10. Rekurencja (2 godz.)
Rekurencyjne typy danych. Rekurencyjne algorytmy: problem wież Hanoi. Przeszukiwanie z nawrotami: problem konika szachowego.

11. Graficzny interfejs użytkownika (2 godz.)
Prosty program z GUI.

PROJEKT:
Projekt to większe zadanie do samodzielnego zrealizowania. Zadnie powinno być ciekawe, np. gra. Konsultowanie, dyskutowanie postępów i ocenianie projektu odbywa się na spotkaniach w laboratorium.

LABORATORIA:
Na kilku początkowych spotkaniach studenci realizują oddzielne zadanie na każdym spotkaniu. Na pozostałych pracują nad małym zadaniem projektowym oraz konsultują duże zadanie, które realizowane jest samodzielnie w ramach Projektu.

1. Organizacja laboratoriów. Podstawowe polecenia systemu operacyjnego Linux. Przesyłanie plików: ssh. Uruchomienie pierwszego programu. Rozdanie tematów małego zadania projektowego.
2. Narzędzia: zintegrowane środowisko programistyczne. Debugger. Linter. Kontrola wersji. Rozdanie tematów dużego zadania projektowego, które będzie realizowane w ramach Projektu.
3. Tworzenie prostych programów: typy danych, operatory, sterowanie przebiegiem wykonania programu, pętle.
4. Tworzenie funkcji, tworzenie własnych typów danych.
5. Tworzenie własnych kontenerów: listy, drzewa
6. Implementacja prostych algorytmów: sortowanie, przeszukiwanie binarne.
7. Komunikacja obiektów ze światem zewnętrznym: konsola, pliki.
8. Obsługa sytuacji wyjątkowych. Testy jednostkowe.
9. Praca nad małym zadaniem projektowym, omówienie postępów prac nad dużym projektem. Omówienie dokumentacji.
10. Praca nad małym zadaniem projektowym, omówienie postępów prac nad dużym projektem. Ocena przyjętych założeń, ocena szkieletu programu. Przypomnienie dobrych praktyk programistycznych.
11. Praca nad małym zadaniem projektowym, omówienie postępów prac nad dużym projektem.
12. Praca nad małym zadaniem projektowym, omówienie postępów prac nad dużym projektem.
13. Praca nad małym zadaniem projektowym, omówienie postępów prac nad dużym projektem.
14. Omówienie i ocena obu projektów.

**Metody oceny:**

Realizacja przedmiotu obejmuje następujące formy zajęć:

- wykład prowadzony w wymiarze 2 godz. tygodniowo; pierwsze kolokwium po zrealizowaniu siódmego tematu, drugie kolokwium kończące przedmiot. Na wykładzie występowały będą elementy interakcji z prowadzącym, co pozwoli na dostosowanie tematyki i tempa wykładu do rzeczywistych potrzeb. Wykłady prowadzone będą w oparciu o slajdy, których treść będzie udostępniana studentom przed wykładem. Oprócz tego przygotowane i udostępnione zostaną przykładowe kody programów, ilustrujące tematykę omawianą na wykładach.
- zajęcia laboratoryjne w wymiarze 2 godz. tygodniowo; w ramach tych zajęć student, korzystając z oprogramowania i sprzętu komputerowego, będąc pod opieką prowadzącego zajęcia, będzie realizował wskazane ćwiczenia dotyczące podstaw programowania. Na laboratorium będzie odbywało się konsultowanie i ocenianie zadań projektowych. Na laboratorium oraz podczas realizacji zadań projektowych wykorzystywana będzie idea odwróconego uczenia (ang. flipped classroom). Oznacza to, że znaczną część wiedzy potrzebnej do realizacji zadań studenci zdobywają samodzielnie, a spotkania z prowadzącym są okazją do weryfikacji jakości zdobytej wiedzy oraz ukierunkowania dalszych poszukiwań.

Sprawdzanie założonych efektów kształcenia realizowane jest przez:
 ocenę wiedzy i umiejętności związanych z realizacją zadań laboratoryjnych – ocena z wybranych ćwiczeń laboratoryjnych oraz zadania projektowego;
 ocena projektu;
 ocenę wiedzy wykazanej na dwóch kolokwiach pisemnych.

**Egzamin:**

nie

**Literatura:**

Mark Summerfield: Python 3. Kompletne wprowadzenie do programowania. Wydanie II. Helion, 2010
Niklaus Wirth: Algorytmy + struktury danych = programy. WNT, 2004
David Beazley, Brian K. Jones Python. Receptury. Wydanie III. Helion, 2014

**Witryna www przedmiotu:**

https://usosweb.usos.pw.edu.pl/kontroler.php?\_action=katalog2/przedmioty/pokazPrzedmiot&prz\_kod=103A-INxxx-ISP-PIPR&callback=g\_00801e58

**Uwagi:**

(-)

## Charakterystyki przedmiotowe

### Profil ogólnoakademicki - wiedza

**Charakterystyka W01:**

ma wiedzę w zakresie tworzenia oprogramowania obejmującą techniki programowania strukturalnego. Umie tworzyć użyteczne aplikacje. Zna środowiska i narzędzia ułatwiające programowanie. Zna podstawy projektowania i tworzenia oprogramowania

Weryfikacja:

kolokwium, laboratorium, ocena projektu

**Powiązane charakterystyki kierunkowe:** W08

**Powiązane charakterystyki obszarowe:** P6U\_W, I.P6S\_WG.o, III.P6S\_WG

### Profil ogólnoakademicki - umiejętności

**Charakterystyka U01:**

potrafi zaprojektować, zrealizować i przetestować program, używając właściwie dobranych metod i narzędzi

Weryfikacja:

laboratorium, ocena projektu

**Powiązane charakterystyki kierunkowe:** U07

**Powiązane charakterystyki obszarowe:** P6U\_U, I.P6S\_UW.o, III.P6S\_UW.o

### Profil ogólnoakademicki - kompetencje społeczne

**Charakterystyka K01:**

rozumie znaczenie wiedzy w rozwiązywaniu problemów poznawczych i praktycznych oraz potrzebę zasięgania opinii ekspertów w przypadku trudności w samodzielnym rozwiązywaniu problemu

Weryfikacja:

laboratorium, projekt

**Powiązane charakterystyki kierunkowe:** K03

**Powiązane charakterystyki obszarowe:** P6U\_K, I.P6S\_KK, I.P6S\_KR