# KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Wprowadzenie do programowania [S1SI1E>WdP]

## Przedmiot

Kierunek studiów
Sztuczna inteligencja/Artificial Intelligence

Rok/Semestr
1/1

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

## Liczba godzin

Wykład
30

Laboratorium
30

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

## Liczba punktów ECTS

5,00

## Koordynatorzy

dr hab. inż. Maciej Antczak prof. PP
maciej.antczak@put.poznan.pl

dr hab. inż. Tomasz Żok prof. PP
tomasz.zok@put.poznan.pl

## Wykładowcy

## Wymagania wstępne

Student rozpoczynający ten moduł powinien posiadać podstawową wiedzę i umiejętności z matematyki i informatyki przy czym doświadczenie w programowaniu mile widziane aczkolwiek nie jest wymagane. Powinien potrafić edytować plik tekstowy z wykorzystaniem ulubionego edytora w systemie Linux/Windows oraz uruchamiać plik skryptu z linii poleceń. Ponadto powinien prezentować takie postawy jak uczciwość, wytrwałość, kreatywność i szacunek dla innych ludzi. W końcu powinien posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł często w języku angielskim.

## Cel przedmiotu

Celem przedmiotu jest zapoznanie studenta z podstawami programowania w języku Python. W szczególności obejmuje to: 1. Przekazanie studentom podstawowej wiedzy o mechanizmach języka Python pozwalających programować w nim zarówno strukturalnie jak i obiektowo. 2. Przekazanie studentom podstawowej wiedzy o specjalizowanych strukturach danych dostępnych w języku Python oraz wykształcenie w nich umiejętności ich praktycznego wykorzystania. 3. Rozwijanie u studentów umiejętności algorytmizacji stosunkowo prostych problemów i ich implementacji. 4. Wykształcenie u studentów umiejętności tworzenia własnych skryptów w języku Python na podstawie zadanej specyfikacji. 5. Nabycie przez studentów umiejętności przewidywania możliwych błędów wykonania oraz zabezpieczania programów w taki sposób, aby tego typu błędów unikać. 6. Przekazanie studentom podstawowej wiedzy o testowaniu jednostkowym programów w języku Python.

## Przedmiotowe efekty uczenia się

Wiedza:
W wyniku przeprowadzonych zajęć student:
1. Posiada praktyczną wiedzę z zakresu programowania strukturalnego i obiektowego w języku Python.
2. Zna i potrafi w praktyce wykorzystywać specjalizowane struktury danych dostępne w języku Python.
3. Zna i rozumie podstawowe techniki, metody, algorytmy oraz narzędzia wykorzystywane w procesie rozwiązywania zadań informatycznych.
4. Ma uporządkowaną, podbudowaną teoretycznie podstawową wiedzę dotyczącą kluczowych obszarów informatyki takich jak m.in. algorytmika, języki i paradygmaty programowania, systemy operacyjne oraz inżynieria oprogramowania.

Umiejętności:
W wyniku przeprowadzonych zajęć student:
1. Posiada podstawowe umiejętności informatyczne w zakresie analizy złożoności obliczeniowej algorytmów, programowania z użyciem języka Python oraz użytkowania systemów operacyjnych.
2. Potrafi formułować i rozwiązywać stosunkowo proste problemy z zakresu informatyki ze szczególnym uwzględnieniem sztucznej inteligencji, stosując odpowiednio dobrane metody.
3. Potrafi łatwo adaptować istniejące oraz formułować i implementować nowe algorytmy z użyciem przynajmniej jednego z dostępnych narzędzi.
4. Potrafi planować i organizować pracę przy realizacji zadań inżynierskich zarówno indywidualnie jak i w zespole.

Kompetencje społeczne:
W wyniku przeprowadzonych zajęć student:
1. Rozumie, że w informatyce ze szczególnym uwzględnieniem sztucznej inteligencji wiedza i umiejętności bardzo szybko stają się przestarzałe, dostrzegając przy tym potrzebę ciągłego dokształcania się oraz podnoszenia własnych kompetencji.
2. Ma świadomość istotności wiedzy i badań naukowych związanych z informatyką i sztuczną inteligencją w rozwiązywaniu praktycznych problemów o kluczowym znaczeniu dla funkcjonowania jednostek, firm, organizacji oraz całego społeczeństwa.
3. Potrafi funkcjonować i współdziałać w grupie, przyjmując w niej różne role oraz potrafi odpowiednio określiać priorytety służące realizacji określonego przez siebie lub innych zadania.

## Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Wiedza nabyta w ramach wykładu jest weryfikowana za pomocą 60-minutowego kolokwium realizowanego na 15 wykładzie, które studenci rozwiązują samodzielnie. Kolokwium składa się z około 20 pytań (testowych, zamkniętych, wielokrotnego wyboru, równo punktowanych). Próg zaliczeniowy: 50% punktów.
Umiejętności nabyte w ramach ćwiczeń laboratoryjnych weryfikowane są na podstawie dwóch 60-minutowych kolokwiów rozwiązywanych indywidualnie przy komputerze. Podczas każdego kolokwium implementowany jest dokładnie jeden program w języku Python transformujący zadane wejście na oczekiwane wyjście, oba zapisane w formacie tekstowym. Ponadto studenci udoskonalają swoje umiejętności programistyczne poprzez: (a) samodzielne rozwiązywanie różnorodnego zestawu predefiniowanych zadań udostępnionych na platformie HackerRank oraz (b) samodzielną implementację projektu na platformie CodinGame.

Wszystkie zgłoszone przez studentów programy zostaną poddane procesowi automatycznego porównywania w celu wyszukiwania plagiatów, które z założenia są niedozwolone. Wszystkie osoby biorące udział w tym procederze zostaną ukarane niezaliczeniem każdego zadania, którego ten problem będzie dotyczył.
Na podstawie powyższych aktywności każdy student uzyskuje jedną ważoną ocenę końcową znormalizowaną z użyciem skali procentowej. Próg zaliczeniowy: 50% punktów.
Studenci, którzy uzyskali bardzo dobre rezultaty (>90%) z ćwiczeń laboratoryjnych są zwalniani z kolokwium zaliczeniowego przeprowadzanego na ostatnim wykładzie.

## Treści programowe

Program wykładu obejmuje następujące zagadnienia:
1. podstawy języka: typy, zmienne, wyrażenia, pętle, sekwencyjne struktury danych, operatory, wyrażenia warunkowe, funkcje.
2. pakowanie/rozpakowywanie krotek, zaawansowane mechanizmy przekazywania parametrów i zwracania wyników, enumerowanie, wyrażenia lambda, funkcje sort vs. sorted, płytkie vs. głębokie kopiowanie struktur danych, deklarowanie klas/pól/metod.
3. obiektowość: dziedziczenie, przesłanianie metod, złożone konstruktory, polimorfizm, abstrakcja, hermetyzacja.
4. transformacja sekwencyjnych struktur danych z wykorzystaniem Map, Filter, List Comprehension, Zip, obsługa wyjątków, asercje, iteratory, dekoratory, generatory, wyrażenia regularne.

## Tematyka zajęć

Program wykładu obejmuje następujące zagadnienia:
Wykład 1: Przedstawienie podstawowych informacji o realizowanym kursie, omówienie podstawowych terminów i pojęć skojarzonych z programowaniem, charakterystyka języka Python, jak rozpocząć przygodę z programowaniem w języku Python?
Wykład 2: Deklarowanie zmiennych, słowa kluczowe, podstawowe instrukcje (np. print, instrukcja przypisania), definiowanie prostych wyrażeń (wykorzystując operatory, operandy, literały, wartości) i ich ewaluacja, weryfikowanie typów wartości, konwersja wybranych typów, wczytywanie danych wprowadzonych przez użytkownika, wywoływanie funkcji, identyfikacja oraz usuwanie błędów w tworzonym oprogramowaniu.
Wykład 3: Moduły (importowanie), przestrzenie nazw, generowanie liczb losowych, wstęp w tematykę programowania obiektowego, wstępna charakterystyka pętli for, podstawowe sekwencyjne struktury danych (np. łańcuchy, listy, krotki), operatory indeksowania i wycinania, podstawowe funkcje operujące na analizowanych strukturach danych (np. len, count, index, split, join), łączenie i powtarzanie list.
Wykład 4: Omówienie wzorca wykorzystywanego w celu akumulowania wartości podczas odwiedzania kolejnych elementów sekwencyjnej struktury danych, iterowanie po wartościach lub z wykorzystaniem indeksów elementów, proste zagnieżdżanie pętli for oraz śledzenie zmian wartości wykorzystywanych w nich zmiennych, obiekt iterowalny vs. zmienna iterowalna, wartości i wyrażenia logiczne, operatory arytmetyczne, logiczne, in, not in, proste, zagnieżdżone i łańcuchowe instrukcje warunkowe, obiekty zmienne vs. niezmienne, usuwanie elementów listy.
Wykład 5: Transformowanie sekwencyjnych struktur danych: obiekty vs. referencje, aliasowanie obiektów, klonowanie list, omówienie podstawowych metod wbudowanych dla list i łańcuchów z przykładami, append vs. concatenate, przykłady formatowania łańcuchów, modyfikowanie zawartości listy podczas jej przeglądania. Przetwarzanie plików tekstowych: podstawowe metody wykorzystywane podczas manipulowania nimi, sposoby identyfikowania plików w systemie plików, czytanie pliku linia po linii oraz zapisywanie nowych informacji do pliku, zastosowanie instrukcji with podczas przetwarzania plików, sposoby przetwarzania plików CSV.
Wykład 6: Słowniki: podstawowe własności tej struktury danych, podstawowe sposoby ich wykorzystywania, charakterystyka dostępnych metod wraz z przykładami, kopiowanie vs. aliasowanie, różne scenariusze iteracji operujących na słownikach. Funkcje: co to są funkcje i dlaczego są w praktyce bardzo użyteczne?, omówienie szczegółowych aspektów kluczowych podczas deklarowania funkcji, podstawowe sposoby przekazywania parametrów do funkcji, zwracanie rezultatu(ów) działania funkcji.
Wykład 7: Lokalność parametrów oraz zmiennych deklarowanych w funkcji, przykłady wykorzystywania zmiennych globalnych w funkcji, reguły rządzące funkcjonalną dekompozycją, instrukcje print vs. return stosowane w ciele funkcji, obiekty zmienne jako parametry przekazywane do funkcji, omówienie

podstawowego przebiegu uruchamiania zbioru instrukcji/wywołań funkcji. Pakowanie/rozpakowywanie krotek: różne warianty przypisywania, enumerowanie elementów w sekwencyjnej strukturze danych, zwracanie krotek lub przekazywanie ich jako argumentów wywołania funkcji.
Wykład 8: Omówienie instrukcji while wraz z przykładami (np. pętla nieskończona), zaprezentowanie przykładu pętli nasłuchującej (oczekującej na dane wprowadzane przez użytkownika) oraz wartości wartowniczej, położenie nacisku na potrzebę walidowania danych wprowadzanych przez użytkownika, przykłady zastosowania instrukcji break i continue w pętli, omówienie podstawowych sposobów wykorzystywania parametrów opcjonalnych podczas wywoływania funkcji wraz ze skojarzonymi z nimi niebezpieczeństwami, zastosowanie nazwy zmiennej wraz z przypisaną do niej wartością podczas wywoływania funkcji, omówienie funkcji anonimowych wykorzystujących wyrażenia lambda, porównanie metody sort z funkcją sorted, przykładowe scenariusze sortowania kluczy słownika lub krotek, wykorzystanie kilku wyrażeń warunkowych w ramach pojedynczego sortowania.
Wykład 9: Przechowywanie złożonych obiektów (np. list, krotek, słowników, funkcji) w listach, zagnieżdżanie słowników, przetwarzanie plików JSON, przykłady stosowania zagnieżdżonych pętli, płytkie vs. głębokie kopiowanie list, wskazówki dotyczące bezpiecznego przetwarzania zagnieżdżonych danych, omówienie mechanizmów służących transformacji sekwencyjnych struktur danych (np. map, filter, zip), wprowadzenie pojęcia wyjątku oraz omówienie zbioru standardowych wyjątków, rozwinięcie przebiegu uruchamiania kodu o sytuacje wyjątkowe, omówienie podstawowej instrukcji try/except wraz z przykładami jej zastosowania.
Wykład 10: Paradygmat programowania obiektowego: przypomnienie wcześniej omawianych terminów np. obiekty, omówienie sposobu definiowania niestandardowych klas, rozpatrywanie parametrów w ramach konstruktorów, definiowanie metod składowych w klasach, przekazywanie obiektów do metod jak i funkcji, konwersja obiektu do postaci tekstowej, zwracanie nowo utworzonych obiektów przez składowe metody klas, sortowanie listy obiektów niestandardowego typu, pola obiektu vs. pola klasy, charakterystyka prostego, wielokrotnego i wielopoziomowego dziedziczenia oraz polimorfizmu.
Wykład 11: Strategie wytwarzania programów działających zgodnie z oczekiwaniami, asercje: weryfikacja (a) spodziewanego typu rezultatu wyrażenia, (b) uruchomienia określonej ścieżki w ramach złożonej instrukcji warunkowej, (c) zaistnienia wartości brzegowych podczas iterowania w pętli, (d) wszystkich możliwych scenariuszy wykonania określonej funkcji, (e) prawidłowości modyfikowania obiektu zmiennego w ramach wywołanej funkcji, (f) określenia wartości parametrów opcjonalnych podczas wywołania funkcji, (g) oczekiwanych stanów obiektów. Testowanie jednostkowe: wprowadzenie pojęcia przypadku testowego oraz pakietu testów, dlaczego warto pisać testy jednostkowe i jak powinno się to robić?, wprowadzenie narzędzi wykorzystywanych do tego celu np. unittest i pytest, prezentacja szeroko wykorzystywanego zbioru asercji, testowanie sytuacji wyjątkowych. Omówienie koncepcji przyrostowego wytwarzania oprogramowania opartego na testach (test-driven development - TDD), testowanie ręczne vs. automatyczne oraz jednostkowe vs. integracyjne, przykłady wykorzystanie instrukcji __import__, statyczna analiza kodu.
Wykład 12: Wykorzystywanie alternatywnych formaterów łańcuchów np. % lub f-string, tworzenie i wypisywanie komentarzy typu docstring, inicjalizacja ziarna generatora pseudolosowego, omówienie wybranych funkcji pozwalających losowo wybierać elementy lub zbiory elementów z wejściowej listy oraz tasujących listę w miejscu, walidacja prawidłowości identyfikatorów np. słów kluczowych lub nazw zmiennych, omówienie zarezerwowanych klas identyfikatorów, przykłady zastosowania isinstance oraz wybranych funkcji modułu math, prawidłowe przetwarzanie liczb dziesiętnych (zapewnianie oczekiwanej precyzji) oraz wymiernych, omówienie szeroko wykorzystywanych funkcji do manipulowania łańcuchami, operatory bitowe, operator trójskładnikowy, przeciążanie operatorów, sposoby naśladowania nieistniejącej instrukcji switch, przykład przypisania zmiennej wewnątrz wyrażenia (z wykorzystaniem tzw. walrus operatora), bardziej złożone przykłady operowania przestrzeniami nazw, wykorzystanie instrukcji else lub pass po lub wewnątrz pętli, udoskonalenie mechanizmu przekazywania parametrów z wykorzystaniem operatorów "/" oraz "*", omówienie instrukcji try/except/finally oraz sposobu zgłaszania wyjątków z wykorzystaniem instrukcji raise.
Wykład 13: Lista wybranych dotąd nieomawianych funkcji wbudowanych, omówienie następujących pojęć, a mianowicie iteratora, generatora, dekoratora, zbioru wraz z przykładami ich zastosowania, przedstawienie specjalizowanych struktur danych oraz iteratorów udostępnianych w module odpowiednio collections i itertools, omówienie sposobu wykorzystywania wyrażeń regularnych oraz właściwości, określenie zbioru praktyk zapewniających wysoką jakość i czytelność kodu, charakterystyka Python Virtual Environment oraz jupyter notebooks.
Wykład 14: Wprowadzenie mechanizmów obsługi daty i czasu, serializacji, wielowątkowości oraz wieloprogramowości.
Wykład 15: Kolowium zaliczeniowe.
Ćwiczenia laboratoryjne prowadzone są w formie piętnastu dwugodzinnych zajęć odbywających się w

laboratorium komputerowym. Pierwsze zajęcia przeznaczone są na zapoznanie studentów z zasadami użytkowania laboratorium i zaliczania ćwiczeń. Ćwiczenia realizowane są przez studentów indywidualnie. Program zajęć laboratoryjnych obejmuje następujące zagadnienia:
1. Wprowadzenie do pracy w systemie Linux – podstawowe instrukcje, przetwarzanie potokowe, przekierowywanie strumieni wejściowych i wyjściowych.
2. Zapoznanie się z interpreterem Pythona oraz systemem pomocy oraz dokumentacją stanowiącymi część środowiska programistycznego.
3. Implementacja prostych programów mających na celu poznanie typów danych i struktur kontrolnych języka w oparciu o interaktywny kurs podstaw programowania w języku Python udostępniony przez Runestone Academy [5].
4. Udoskonalanie umiejętności programowania w języku Python poprzez samodzielne rozwiązywanie jak największej liczby zadań charakteryzujących się różnym poziomem trudności udostępnionych na platformie HackerRank.
5. Samodzielne opracowanie oraz implementacja programu rozwiązującego problem opublikowany na platformie CodinGame.
6. Testowanie implementowanych programów oraz przygotowanie zbioru testów jednostkowych dla oprogramowania wytwarzanego na platformie CodinGame.

## Metody dydaktyczne

1. Wykład: prezentacja multimedialna wedle potrzeby ilustrowana dodatkowymi przykładami prezentowanymi na tablicy.
2. Ćwiczenia laboratoryjne: ćwiczenia praktyczne przy komputerze realizowane według określonego scenariusza, implementacja fragmentów kodu oraz skryptów rozwiązujących stosunkowo proste problemy algorytmiczne, dyskusja zastosowanych rozwiązań oraz konstrukcji programistycznych.

## Literatura

Podstawowa
1. Automatyzacja nudnych zadań z Pythonem : nauka programowania. Autor: Sweigart, Albert. Górczyński, Robert (tłumacz). Wydawnictwo Helion, 2017.
2. Python : instrukcje dla programisty. Autor: Matthes, Eric. Górczyński, Robert. (tłumacz). Wydawnictwo Helion, 2020.
3. Python : zacznij programować. Autor: Miles, Rob S. Meryk, Radosław (tłumacz). Wydawnictwo Helion, 2019.
4. Python : szybko i prosto. Autor: Ceder, Naomi R. Bogusławska, Katarzyna (tłumacz). Wydawnictwo Helion 2019.
5. Kurs podstaw programowania w języku Python udostępniony przez Runestone Academy (https://runestone.academy/runestone/books/published/fopp/index.html).
Uzupełniająca
6. Python : uczymy się programowania. Autor: Bell, Ana. Gola, Przemysław (tłumacz). Wydawnictwo Helion, 2019.
7. Python 3 : proste wprowadzenie do fascynującego świata programowania. Autor: Shaw, Zed. Lachowski, Lech (tłumacz). Wydawnictwo Helion, 2018.
8. Wstęp do języka Python dla początkujących. Jak szybko nauczyć się języka Python? Kurs udostępniony przez Data Flair (https://data-flair.training/blogs/python-tutorial/)

## Bilans nakładu pracy przeciętnego studenta

| | Godzin | ECTS |
|---|---|---|
| Łączny nakład pracy | 125 | 5,00 |
| Zajęcia wymagające bezpośredniego kontaktu z nauczycielem | 60 | 2,50 |
| Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu) | 65 | 2,50 |