Wymagania edukacyjne Informatyka – semestr II (Liceum wieczorowe / stacjonarne)

Temat	Wymagania konieczne (ocena dopuszczająca)	Wymagania podstawowe (ocena dostateczna)	Wymagania rozszerzające (ocena dobra)	Wymagania dopełniające (ocena bardzo dobra)	Wymagania wykraczające (ocena celująca)
	Słuchacz:	**Słuchacz:**	**Słuchacz:**	**Słuchacz:**	**Słuchacz:**
1. Lekcja organizacyjna BHP pracy z komputerem, zasady promowania i oceniania	Zapoznanie z Przedmiotowym Systemem Oceniania (PSO) w semestrze II (algorytmika i programowanie), zasadami BHP, kryteriami sprawdzianu dopuszczeniowego, warunkami egzaminu semestralnego oraz konfiguracją środowiska programistycznego Python w domu.				
Moduł C Wokół algorytmiki i programowania					
2. Temat C1. Wprowadzenie do programowania i rozwiązywania problemów z wykorzystaniem komputera	– wyjaśnia pojęcie algorytmu na prostym przykładzie życiowym (np. przepis kulinarny) – odróżnia dane wejściowe od wyniku działania algorytmu – wymienia przynajmniej jeden sposób opisu algorytmu	– omawia etapy rozwiązywania problemu (sformułowanie zadania, specyfikacja, projekt, programowanie, testy) – formułuje prostą specyfikację problemu obliczeniowego (dane, wyniki, warunki) – przedstawia algorytm liniowy w postaci listy kroków i prostego schematu blokowego	– poprawnie stosuje znormalizowane symbole schematów blokowych (start/stop, operacyjny, I/O, warunkowy) – analizuje poprawność algorytmu i przeprowadza ręczne śledzenie wykonania krok po kroku – dobiera reprezentatywne dane testowe sprawdzające poprawność algorytmu	– samodzielnie dekomponuje złożony problem obliczeniowy na podproblemy algorytmiczne – projektuje optymalny algorytm i zapisuje go w pseudokodzie oraz schemacie blokowym – identyfikuje i eliminuje błędy logiczne na etapie projektu algorytmu	– analizuje złożoność obliczeniową i poprawność formalną zaprojektowanych algorytmów – tworzy precyzyjne specyfikacje i modele matematyczne dla nietypowych problemów decyzyjnych – proponuje alternatywne metody rozwiązania tego samego zadania i uzasadnia wybór najefektywniejszej
3. Temat C1. Na czym polega programowanie?	– wyjaśnia, do czego służy program komputerowy – rozpoznaje kod źródłowy programu zapisanego w edytorze tekstu – wymienia przynajmniej jeden język programowania wysokiego poziomu	– wyjaśnia różnicę między językiem niskiego poziomu (maszynowym) a językiem wysokiego poziomu – charakteryzuje pojęcia: kod źródłowy, kompilacja, interpretacja, kod wynikowy – podaje cechy charakterystyczne języka Python jako języka skryptowego	– porównuje proces translacji kodu w językach kompilowanych (np. C++) i interpretowanych (Python) – omawia zalety i wady programowania w językach interpretowanych (przenośność, szybkość prototypowania) – charakteryzuje rolę środowiska uruchomieniowego i bibliotek standardowych	– szczegółowo objaśnia architekturę maszyny wirtualnej Pythona (kod bajtowy .pyc, PVM) – analizuje strukturę procesu budowania aplikacji i rolę konsolidatora (linkera) w językach kompilowanych – potrafi dobrać odpowiedni paradygmat i język do zadanego problemu inżynierskiego	– wykazuje pogłębioną wiedzę na temat kompilatorów JIT (Just-In-Time) i optymalizacji kodu maszynowego – analizuje trendy w ewolucji współczesnych języków programowania i systemów typowania – potrafi dyskutować o architekturze interpreterów i ich wpływie na wydajność obliczeniową

4. Temat C2. Tworzenie programów w języku Python	– uruchamia środowisko IDLE lub powłokę Pythona – przepisuje i wykonuje polecenie `print('Witaj')` – wykonuje proste działanie arytmetyczne w konsoli interaktywnej	– posługuje się funkcją `print()` do wyświetlania pojedynczych i wielu argumentów – stosuje znaki ucieczki (`\n`, `\t`) w tekście wyjściowym – odróżnia tekst w cudzysłowie od zmiennych i liczb – usuwa podstawowe błędy składniowe (SyntaxError)	– konfiguruje parametry funkcji `print()` za pomocą argumentów `sep` i `end` – swobodnie pracuje w powłoce interaktywnej testując fragmenty kodu – analizuje i koryguje błędy literowe i składniowe w złożonych poleceniach wypisywania	– tworzy estetyczne komunikaty konsolowe z wykorzystaniem formatowania wielowierszowego (potrójny cudzysłów) – tworzy złożone napisy łączone i powielane za pomocą operatorów `+` i `*` na łańcuchach – sprawnie posługuje się dokumentacją podręczną języka (funkcja `help()`)	– projektuje zaawansowany interfejs tekstowy w konsoli (ramki, tabele znakowe, kolorowanie wyjścia) – tworzy sparametryzowane szablony komunikatów z dynamicznym podstawianiem danych – wykazuje biegłość w posługiwaniu się konsolą i powłokami zaawansowanymi (IPython)
5. Temat C2. Stosowanie zmiennych	– tworzy zmienną i przypisuje jej wartość całkowitą lub tekstową – stosuje podstawowe nazewnictwo zmiennych (bez spacji i znaków polskich) – wykonuje proste dodawanie i odejmowanie zmiennych	– wyjaśnia ideę dynamicznego typowania w Pythonie – rozróżnia podstawowe typy danych: `int`, `float`, `str`, `bool` – stosuje operatory arytmetyczne: `+`, `-`, `*`, `/`, `//` (dzielenie całkowite), `%` (modulo), `**` (potęgowanie) – zachowuje prawidłową kolejność wykonywania działań z użyciem nawiasów	– sprawdza typ zmiennej za pomocą funkcji `type()` – stosuje operatory przypisania ze skrótem (`+=`, `-=`, `*=`, `/=`) – pisze programy realizujące wieloetapowe kalkulacje geometryczne i fizyczne – dba o stosowanie znaczących, czytelnych nazw zmiennych (zgodnie z konwencją snake_case)	– precyzyjnie wyjaśnia specyfikę reprezentacji liczb zmiennoprzecinkowych w pamięci komputera – bezbłędnie posługuje się dzieleniem całkowitym i resztą z dzielenia w problemach cyklicznych – optymalizuje wyrażenia arytmetyczne pod kątem precyzji obliczeń	– demonstruje zrozumienie modelu pamięci Pythona (obiekty, referencje, tożsamość `id()`) – analizuje zjawiska przepełnienia i precyzji zmiennoprzecinkowej w obliczeniach naukowych – tworzy zaawansowane formuły matematyczne z wykorzystaniem biblioteki precyzyjnych ułamków (`fractions`, `decimal`)
6. Temat C2. Wyprowadzanie komunikatów i wyników na ekran monitora	– pobiera tekst od użytkownika za pomocą funkcji `input()` – wyświetla pobrany tekst w powitaniu na ekranie – uruchamia gotowy program pobierający dane	– wyjaśnia, dlaczego dane pobrane przez `input()` są zawsze typu tekstowego (`str`) – dokonuje jawnej konwersji (rzutowania) typów za pomocą `int()` oraz `float()` – pobiera dwie liczby, wykonuje na nich obliczenia i wyświetla wynik – formatuje wyjście łącząc tekst z liczbami	– biegle posługuje się nowoczesnym formatowaniem f-string (`f'Wynik: {zmienna}'`) – formatuje precyzję wyświetlania liczb rzeczywistych (np. `:.2f` dla dwóch miejsc po przecinku) – rozumie i diagnozuje błędy niepoprawnego formatu danych wejściowych (`ValueError`)	– projektuje ergonomiczny, czytelny dialog programu z użytkownikiem (jasne instrukcje wprowadzania) – stosuje zaawansowane specyfikatory formatowania w f-stringach (wyrównanie do lewej/prawej, dopełnianie zerami) – przygotowuje program kalkulatora walutowego lub podatkowego o profesjonalnym wyglądzie wyników	– implementuje zaawansowane parsowanie i weryfikację poprawności danych wprowadzanych przez użytkownika – tworzy uniwersalne funkcje wejścia/wyjścia zabezpieczające program przed wprowadzeniem błędnych znaków – demonstruje wzorcowy styl projektowania interakcji w aplikacjach konsolowych
7. Temat C3. Tworzenie programów w języku Python	– otwiera nowe okno edytora kodu w IDLE – przepisuje prosty program	– organizuje strukturę programu konsolowego (pobranie danych, obliczenia, prezentacja wyników) – importuje moduł matematyczny `math`	– pisze programy obliczeniowe z wykorzystaniem wzorów geometrycznych i fizycznych – dba o przejrzysty styl kodu, stosując logiczne odstępy i strukturę	– tworzy modularne programy konsolowe realizujące złożone symulacje zjawisk fizycznych/ekonomicznych – stosuje moduł `random` do	– projektuje zaawansowane aplikacje konsolowe integrujące moduły biblioteki standardowej – optymalizuje strukturę programu pod kątem ponownego

	składający się z kilku instrukcji – uruchamia program klawiszem F5	– stosuje funkcje z modułu `math` (np. `math.sqrt()`, `math.pow()`, stała `math.pi`) w programie	blokową – porównuje różne sposoby importowania modułów (`import math`, `from math import sqrt`)	generowania losowych danych testowych w programie – przestrzega standardów czytelności kodu zgodnych z przewodnikiem PEP 8	wykorzystania kodu – tworzy własne mini-biblioteki funkcji narzędziowych
8. Temat C3. Pisanie programu w trybie skryptowym	– zapisuje utworzony kod w pliku z rozszerzeniem `.py` – otwiera wcześniej zapisany plik skryptu – wstawia komentarz jednowierszowy ze znakiem `#`	– wyjaśnia różnicę między pracą w powłoce interaktywnej a wykonywaniem pliku skryptowego – stosuje komentarze dokumentujące działanie poszczególnych partii kodu – nadaje plikom skryptów znaczące nazwy bez spacji i znaków diakrytycznych – odczytuje komunikaty o błędach w oknie konsoli	– uruchamia skrypt Pythona bezpośrednio z wiersza poleceń / terminala systemu operacyjnego – analizuje ślad stosu (Traceback) i szybko lokalizuje linię, w której wystąpił błąd – modyfikuje i rozbudowuje istniejący skrypt o nowe funkcjonalności	– tworzy skrypty narzędziowe automatyzujące codzienne zadania obliczeniowe – stosuje wielowierszowe docstringi do dokumentowania przeznaczenia skryptu – sprawnie posługuje się zewnętrznym edytorem kodu (np. VS Code, Thonny, PyCharm)	– tworzy zaawansowane skrypty wsadowe uruchamiane z parametrami wiersza poleceń (`sys.argv`) – implementuje środowiska wirtualne Pythona (venv) do izolacji projektów – demonstruje pełną biegłość w posługiwaniu się narzędziami developerskimi ekosystemu Pythona
9. Temat C3. Zapisywanie rozwiązania problemów w języku Python	– uruchamia program rozwiązujący zadany problem i wprowadza dane – sprawdza działanie programu dla podanych w treści zadania przykładów – dokonuje drobnej zmiany wartości stałych w kodzie	– samodzielnie przekłada treść prostego zadania rachunkowego na działający skrypt Python – dobiera dane testowe sprawdzające poprawność obliczeń dla typowych wartości – weryfikuje zgodność wyników programu z obliczeniami ręcznymi	– tworzy pełną specyfikację zadania i realizuje kompletny proces implementacji problemu z fizyki lub bankowości – testuje program dla danych granicznych i nietypowych (np. zero, wartości ujemne) – formatuje raport końcowy programu w czytelny sposób dla odbiorcy	– projektuje i programuje rozwiązanie problemu o wieloetapowej logice biznesowej (np. taryfikator opłat, kalkulator kredytowy) – samodzielnie wykrywa i usuwa błędy semantyczne w algorytmie obliczeniowym – dba o pełną odporność programu na typowe pomyłki użytkownika	– modeluje matematycznie i programuje symulacje złożonych procesów ekonomicznych lub przyrodniczych – tworzy kompletny projekt programistyczny z dokumentacją techniczną i zestawem testów jednostkowych – analizuje efektywność zaproponowanego algorytmu numerycznego
10. Sprawdzian dopuszczeniowy	Praktyczny sprawdzian programistyczny przy komputerze weryfikujący umiejętności tworzenia skryptów w Pythonie (pobieranie danych, rzutowanie typów, wyrażenia arytmetyczne, formatowanie wyjścia, specyfikacja zadania) – warunek konieczny dopuszczenia do egzaminu semestralnego.				
11. Temat C4. Stosowanie instrukcji warunkowych w językach C++ i Python	– rozpoznaje instrukcję warunkową `if` w kodzie programu – podaje przykład sytuacji decyzyjnej w algorytmie – zapisuje prosty warunek sprawdzający znak liczby lub równość	– zna składnię instrukcji warunkowej w Pythonie (`if...else`) oraz w C++ – rozumie kluczowe znaczenie wcięć blokowych (indentation) w Pythonie i klamer w C++ – stosuje operatory relacyjne: `==`, `!=`, `<`, `>`, `<=`, `>=` – buduje program podejmujący prostą decyzję dwuwariantową	– implementuje rozgałęzienia wielowariantowe z użyciem `elif` (Python) oraz `else if` (C++) – łączy warunki za pomocą operatorów logicznych: koniunkcji (`and` / `&&`), alternatywy (`or` / `||`), negacji (`not` / `!`) – konstruuje poprawne warunki złożone (np. przynależność do przedziału liczbowego)	– bezbłędnie przekłada rozbudowany schemat blokowy z wieloma blokami decyzyjnymi na kod źródłowy – unika typowych błędów (np. mylenie `=` z `==` w C++, brak dwukropka w Pythonie) – optymalizuje kolejność sprawdzania warunków pod kątem szybkości działania programu	– projektuje bezbłędne, wielopoziomowe systemy reguł decyzyjnych o złożonej strukturze warunków – precyzyjnie stosuje prawa De Morgana przy przekształcaniu i upraszczaniu skomplikowanych wyrażeń logicznych – sprawnie implementuje logikę

			– testuje program dla wszystkich gałęzi decyzyjnych		warunkową równolegle w Pythonie i C++
12. Temat C4. Algorytmy z warunkami w językach C++ i Python	– uruchamia program znajdujący większą z dwóch liczb – modyfikuje dane wejściowe i analizuje zmianę ścieżki programu – wskazuje blok warunkowy w schemacie blokowym	– projektuje i programuje algorytm wyznaczania maksimum i minimum z dwóch liczb – implementuje sprawdzanie parzystości liczby za pomocą operatora modulo `%` – programuje algorytm badania podzielności liczby przez zadaną wartość – dobiera dane testowe sprawdzające każdą ścieżkę wykonania programu	– programuje algorytm wyznaczania największej z trzech liczb (wariant z zagnieżdżeniem lub operatorami logicznymi) – implementuje algorytm badania warunku trójkąta dla trzech podanych długości odcinków – programuje rozwiązywanie równania liniowego $ax + b = 0$ z uwzględnieniem przypadków szczególnych ($a=0$) – analizuje przypadki brzegowe w testowaniu kodu	– implementuje pełny algorytm rozwiązywania równania kwadratowego $ax^2 + bx + c = 0$ z dyskusją liczby rozwiązań ($\Delta > 0, = 0, < 0$) – tworzy program klasyfikacji wielostopniowej (np. taryfikator opłat, system oceniania słuchaczy) – stosuje zagnieżdżone instrukcje warunkowe z zachowaniem pełnej czytelności kodu	– projektuje i implementuje zaawansowane algorytmy decyzyjne z zakresu geometrii obliczeniowej (np. położenie punktu względem prostej/odcinka) – przeprowadza formalny dowód poprawności algorytmu dla wszystkich klas danych wejściowych – porównuje elegancję i zwięzłość zapisu struktur warunkowych w Pythonie i C++
13. Temat C5. Stosowanie instrukcji iteracyjnych w językach C++ i Python	– wyjaśnia pojęcie iteracji jako wielokrotnego powtórzenia operacji – rozpoznaje pętlę `for` w kodzie źródłowym – uruchamia program wypisujący kolejne liczby całkowite	– zna składnię pętli o znanej liczbie powtórzeń: `for i in range(...)` w Pythonie oraz `for` w C++ – zna składnię pętli warunkowej `while` w wybranym języku – pisze program wykonujący powtórzenie operacji zadaną liczbę razy – stosuje pętlę do obliczenia sumy kilku wprowadzonych liczb	– biegle posługuje się funkcją `range(start, stop, step)` w Pythonie (w tym odliczanie w dół z krokiem ujemnym) – poprawnie operuje zmiennymi pełniącymi rolę licznika i sumatora (akumulatora) – implementuje algorytm obliczania średniej arytmetycznej n liczb podanych przez użytkownika – rozumie mechanizm powstawania pętli nieskończonej i potrafi jej zapobiec	– implementuje klasyczne algorytmy iteracyjne: obliczanie silni liczby ($n!$), potęgowanie liczby o wykładniku naturalnym – programuje algorytmy wyszukiwania elementów spełniających warunek w ciągu liczb – stosuje pętle zagnieżdżone (np. generowanie tabliczki mnożenia lub wzorów znakowych)	– projektuje i programuje wydajne algorytmy iteracyjne rozwiązujące zagadnienia teorii liczb (np. badanie liczb pierwszych, liczby doskonałe) – optymalizuje liczbę iteracji w pętlach obliczeniowych – tworzy rozbudowane programy konsolowe łączące instrukcje iteracyjne, warunkowe i zaawansowaną logikę przetwarzania danych
14. Egzamin semestralny	Praktyczny egzamin semestralny przy stanowisku komputerowym z zakresu algorytmiki i programowania w języku Python (lub C++): implementacja programu z instrukcjami warunkowymi i iteracyjnymi według zadanej specyfikacji.				
15. Omówienie wyników egzaminu	Przedstawienie wyników egzaminu semestralnego, omówienie wzorcowych rozwiązań kodu, analiza błędów składniowych i logicznych, wystawienie ocen semestralnych i zatwierdzenie promocji na semestr III.				