МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
На засіданні кафедри
економічної кібернетики і
системного аналізу
Протокол № 1 від 22.08.2023 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

_____ Каріна НЕМАШКАЛО

ПРОГРАМУВАННЯ
робоча програма навчальної дисципліни (РПНД)

Галузь знань	**12 «Інформаційні технології»**
Спеціальність	**124 «Системний аналіз»**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**Управління складними системами**

Статус дисципліни	**обов'язкова**
Мова викладання, навчання та оцінювання	**українська**

Розробники:
к.е.н., доцент _____ Роман ЯЦЕНКО

викладач _____ Антон ЯКОВЛЄВ

Завідувач кафедри
економічної кібернетики
і системного аналізу _____ Лідія ГУР'ЯНОВА

Гарант програми _____ Оксана ПАНАСЕНКО

Харків
2024

ВСТУП

Вивчення навчальної дисципліни "Програмування" для здобувачів вищої освіти галузі знань 12 «Інформаційні технології» є надзвичайно актуальним у сучасному світі. Мова програмування Python, яка володіє простим синтаксисом та великою потужністю, стала популярним інструментом для розв'язання різноманітних завдань у сферах програмування, аналізу даних, штучного інтелекту та веб-розробки. Вміння працювати з цією мовою надає здобувачам освіти конкурентну перевагу на ринку праці, сприяє розвитку аналітичних та алгоритмічних навичок, а також допомагає їм в освоєнні сучасних технологій та стандартів розробки програмного забезпечення. Практичні навички, отримані під час вивчення Python, роблять здобувачів освіти більш підготовленими до викликів сучасної індустрії інформаційних технологій.

Мета навчальної дисципліни є надання здобувачам вищої освіти системи теоретичних і практичних знань з програмування з метою побудови ефективних алгоритмів обробки даних та їх програмної реалізації. У якості головного інструменту обрано сучасну мову програмування Python.

Основним завданням вивчення дисципліни «Програмування» є вивчення теоретичних і практичних засад програмування, основ алгоритмізації та структур даних, формування у здобувачів навичок використання отриманих знань на практиці, створення власного програмних засобів.

Предметом вивчення навчальної дисципліни є концепції та інструменти побудови програмних засобів для підвищення ефективності функціонування економічних систем в сучасних умовах розвитку ринкового середовища.

Об'єктом навчальної дисципліни "Програмування" є вивчення основних принципів розробки програмного забезпечення та навичок написання програм. Здобувачі освоюють базові концепції програмування, включаючи структури даних, алгоритми, управління потоком виконання, тестування і налагодження коду. Вивчення цієї дисципліни визначає фундаментальні компетентності для подальшого успішного застосування в індустрії розробки програмного забезпечення та інших сферах інформаційних технологій.

Результати навчання та компетентності, які формує навчальна дисципліна визначено в табл. 1.

Таблиця 1

Результати навчання та компетентності, які формує навчальна дисципліна

Результати навчання	**Компетентності, якими повинен оволодіти здобувач освіти**
РН8. Володіти сучасними методами розробки програм і програмних комплексів та прийняття оптимальних рішень щодо складу програмного забезпечення, алгоритмів процедур і операцій.	КЗ 2. Здатність застосовувати знання у практичних ситуаціях КЗ 14. Здатність оцінювати та забезпечувати якість виконуваних робіт КФ 7. Здатність використовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей та прогнозування поведінки конкретних систем а саме: об'єктно-орієнтований підхід при проектуванні складних систем різної природи, прикладні математичні пакети, застосування баз даних і знань
РН9. Вміти створювати ефективні алгоритми для обчислювальних задач системного аналізу та систем підтримки прийняття рішень.	КЗ 2. Здатність застосовувати знання у практичних ситуаціях КЗ 13. Здатність працювати в міжнародному контексті КФ 7. Здатність використовувати сучасні інформаційні технології для комп'ютерної реалізації математичних моделей та прогнозування поведінки конкретних систем а саме: об'єктно-орієнтований підхід при проектуванні складних систем різної природи, прикладні математичні пакети, застосування баз даних і знань

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Зміст навчальної дисципліни
Змістовий модуль 1.
Основи програмування на Python

Тема 1. Основи мови програмування Python

1.1. Історія мови програмування Python.

Розглядається походження та етапи розвитку мови програмування Python, визначаючи ключові події та внесок Гвідо ван Россума у її створення.

1.2.Переваги Python.

Розглядаються основні переваги мови Python, такі як читабельний синтаксис, широкі можливості бібліотек та фреймворків, що роблять її привабливою для розробників.

1.3.Основні елементи мови.

Розглядаються базові елементи Python, включаючи змінні, типи даних, операції та умовні конструкції, які формують основу для написання програм.

1.4.Структурні елементи програми.

Розглядаються структурні елементи програмування, такі як цикли, функції та обробка виключень у Python, що дозволяє ефективно організовувати та виконувати код.

1.5.The Zen of Python.

Розкриваються принципи The Zen of Python, які визначають філософію мови: засади, які надихають на створення якісного та зрозумілого коду в середовищі Python.

Тема 2. Типи даних та змінні. Введення-виведення даних.

2.1. Оператори введення та виведення даних.

Розглядається робота з операторами введення та виведення в мові програмування Python, включаючи команди для обміну інформацією з користувачем та зчитування/виведення даних.

2.2. Типи даних.

Вивчаються різні типи даних у Python, такі як цілі числа, дійсні числа, рядки та булеві значення, зокрема їхні особливості та використання в програмуванні.

2.3. Перетворення типів даних.

Розглядається можливість перетворення типів даних в Python, що дозволяє змінювати формат або призначення змінних з одного типу в інший.

2.4. Складені оператори присвоювання.

Розглядаються розширені оператори присвоювання, такі як операції із змінними, що дозволяють коротше записувати операції присвоювання в Python.

Тема 3. Умовний оператор та розгалужений обчислювальний процес.

3.1. Синтаксис умовного оператору.

Розглядається синтаксис та використання умовного оператору в Python для виконання різних дій в залежності від умови.

3.2. Вкладені умовні інструкції.

У цьому пункті вивчається можливість вкладення умовних інструкцій, що дозволяє створювати складніші конструкції для обробки різних випадків.

3.3. Оператори порівняння.

Розглядаються різні оператори порівняння в Python, такі як ==, !=, <, >, <=, >=, які використовуються для порівняння значень.

3.4. Логічні оператори.

Вивчається використання логічних операторів (and, or, not) для об'єднання та перевірки умов в програмах на Python.

3.5. Каскадні умовні конструкції.

Розглядається використання каскадних умовних конструкцій для послідовної перевірки умов та виконання відповідних блоків коду.

3.6. Тернарний умовний оператор.

У цьому пункті досліджується тернарний умовний оператор, який надає коротший синтаксис для визначення значення залежно від умови.

Тема 4. Цикл з визначеною кількістю ітерацій.

4.1. Цикл for.

Розглядається цикл for в мові програмування Python, який дозволяє ітеруватися по послідовності або колекції об'єктів для виконання определених операцій.

4.2. Функція range.

Розглядається функція range, яка створює послідовність чисел для використання у циклах та інших контекстах, що дозволяє керувати кількістю ітерацій.

4.3. Генерація псевдовипадкових чисел

У даному пункті вивчається генерація псевдовипадкових чисел в Python, використовуючи модуль random, що дозволяє створювати випадкові значення для різних варіантів застосувань.

Тема 5. Обробка символьних даних.

5.1. Загальні підходи до обробки символьних даних.

У цьому пункті вивчається загальний підхід до обробки символьних даних в мові програмування Python, включаючи роботу з рядками та символами.

5.2. Створення зрізів.

Розглядається тема створення зрізів (slicing) в Python, яка дозволяє отримувати підрядки рядків або списків, спрощуючи роботу зі символьними даними.

5.3. Методи роботи з рядками.

У цьому пункті вивчаються різноманітні методи роботи з рядками в Python, такі як пошук, заміна, перетворення регістрів та інші, що полегшують обробку символьних даних.

Тема 6. Цикли з перевіркою умов.

6.1. Цикл while.

Розглядається цикл while в мові програмування Python, який виконує блок коду, доки вказана умова є істинною, дозволяючи гнучку реалізацію циклічних структур.

6.2. Оператори управління циклом.

Розглядаються оператори управління циклом (break та continue) в Python, які надають можливість вийти з циклу або перейти до наступної ітерації в залежності від виконання умов, що спрощує управління циклічним виконанням коду.

6.3. Множинне присвоювання.

Досліджується множинне присвоювання в Python, яке дозволяє одночасно присвоювати значення декільком змінним, спрощуючи і структуруючи код.

Тема 7. Структурне програмування за допомогою функцій.

7.1. Принципи структурного програмування.

Розглядаються основні принципи структурного програмування, такі як розкладання програми на логічні блоки, уникання вкладених структур та використання узагальнених структур керування.

7.2. Оголошення функцій.

Розглядається тема оголошення та визначення функцій в Python, яке дозволяє створювати власні блоки коду з можливістю повторного використання та структурує програму.

7.3. Локальні та глобальні змінні.

Вивчаються концепції локальних та глобальних змінних в контексті функцій в Python, що дозволяє керувати областю видимості змінних та ефективно використовувати їх у програмі.

Тема 8. Обробка даних у послідовностях та списках.

8.1. Поняття списку.

Розглядається концепція списків в мові програмування Python, їх створення та основні властивості, які дозволяють зберігати та обробляти послідовності даних.

8.2. Методи split та join.

Розглядаються методи split та join, які дозволяють розділяти рядки на списки та з'єднувати елементи списку в рядок відповідно, розширюючи можливості обробки даних у текстових послідовностях.

8.3. Операції зі списками.

Вивчаються різноманітні операції зі списками в Python, такі як додавання, видалення, зміна та інші, що дозволяють ефективно маніпулювати даними у списках.

8.4. Генератори списків.

Розглядається використання генераторів списків, які дозволяють компактно та ефективно створювати списки за допомогою одного рядка коду на основі виразів та умов.

Тема 9. Двовимірні структури даних.

9.1. Вкладені списки.

Розглядається концепція вкладених списків в мові програмування Python, яка дозволяє створювати структури даних з більшою розгалуженістю та організовувати дані в двовимірній формі.

9.2. Створення та введення вкладених списків.

Розглядається процес створення та введення вкладених списків, включаючи методи ініціалізації та заповнення даними, що допомагає побудувати багатовимірні структури даних.

9.3. Приклади програм з двовимірними структурами даних.

Аналізуються приклади програм, які використовують двовимірні структури даних, щоб демонструвати їх застосування та розвивати навички роботи з такими об'єктами в програмуванні.

Тема 10. Множинні та асоціативні структури даних.

10.1. Призначення та створення словників.

Розглядається призначення та створення словників в Python, які є асоціативними структурами даних, що дозволяють зберігати пари ключ-значення для ефективного доступу до даних.

10.2. Робота з елементами словника.

Розглядаються операції та методи роботи з елементами словника, такі як додавання, видалення та зміна значень, що надає засоби управління та модифікації асоціативних даних.

10.3. Призначення множини як структури даних.

Вивчається призначення множин як структури даних в Python, яка дозволяє зберігати унікальні елементи без визначення порядку, що розширює можливості обробки даних.

10.4. Операції над множинами.

Розглядаються різноманітні операції над множинами, такі як об'єднання, перетин, різниця та інші, що надають засоби для ефективної роботи з множинними структурами даних.

Тема 11. Робота з файлами та файловою системою.

11.1. Загальні підходи до роботи з текстовими файлами.

Вивчаються загальні підходи та стратегії роботи з текстовими файлами в мові програмування Python, зокрема, відкриття, закриття та операції читання/запису.

11.2. Операції відкриття та закриття файлів.

Розглядаються операції відкриття та закриття файлів, що дозволяють програмі взаємодіяти з файловою системою та ефективно керувати потоками даних.

11.3. Запис даних до файлу.

Вивчається процес запису даних до файлу, включаючи використання різних методів та форматування для забезпечення правильного збереження інформації.

11.4. Зчитування даних з файлу.

Розглядається техніка зчитування даних з файлу, а також обробка та аналіз отриманої інформації з метою подальшого використання в програмі.

11.5. Методи роботи зі складеними даними.

В даному пункті розглядається робота зі складеними даними у файлі, такими як JSON або CSV, та методи їх обробки та використання в програмі.

Тема 12. Стандартна бібліотека Python

12.1. Модуль os та робота з операційною системою.

Розглядається модуль os, який надає функції для взаємодії з операційною системою, включаючи роботу з файловою системою та виконання команд.

12.2. Модуль datetime та робота з датами та часом.

Вивчається модуль datetime, який дозволяє працювати з датами та часом, включаючи операції форматування та обчислення інтервалів.

12.3. Модуль re та регулярні вирази.

Вивчається модуль re, який дозволяє використовувати регулярні вирази для обробки та пошуку текстової інформації.

12.4. Модуль urllib та робота з мережею.

Розглядається модуль urllib, який надає інструменти для роботи з мережею, включаючи взаємодію з URL-адресами та завантаження веб-ресурсів.

Змістовий модуль 2.

Структури даних та об’єктно-орієнтоване програмування

Тема 13. Колекції та генератори

13.1. Основні поняття.

Розглядається основна термінологія та поняття, пов'язані з колекціями та генераторами, що є важливим для подальшого вивчення теми.

13.2. Загальні підходи до роботи з будь-якою колекцією.

Вивчається загальний підхід до роботи з колекціями, незалежно від їх типу, з фокусом на спільних операціях та методах.

13.3. Загальні методи для колекцій.

Розглядаються загальні методи, які можна застосовувати до різних типів колекцій для виконання різних завдань.

13.4. Конвертація одного типу колекції в інший.

Вивчається процес конвертації колекцій, що дозволяє змінювати тип або структуру колекції.

13.5. Індексування.

Розглядається можливість доступу до конкретних елементів колекцій за допомогою індексів.

13.6. Зрізи.

Вивчаються зрізи (slicing) як засіб вибору підмножини елементів у колекціях.

13.7. Сортування елементів колекції.

Розглядається метод сортування елементів колекцій для легкого управління порядком елементів.

13.8. Об'єднання послідовностей та словників.

Вивчається спосіб об'єднання та комбінування різних колекцій у єдину структуру.

13.9. Операції над множинами.

Розглядаються операції, які можна виконувати над множинами для здійснення різних логічних операцій.

13.10. Зміна та об'єднання колекцій.

Вивчається можливість модифікації та об'єднання колекцій для динамічної зміни їхнього вмісту.

13.11. Визначення та класифікація генераторів.

Розглядається концепція генераторів, їх визначення та класифікація за різними критеріями.

13.12. Синтаксис генераторів.

Вивчається синтаксис визначення та використання генераторів у мові програмування Python.

13.13. Вирази-генератори.

Розглядаються вирази-генератори як засіб створення компактних та ефективних генераторів у Python.

13.14. Генерація колекцій.

Вивчається процес генерації колекцій за допомогою генераторів, що полегшує створення послідовностей.

13.15. Функція-генератор.

Розглядається концепція функцій-генераторів, які дозволяють створювати ітератори за допомогою функцій у Python.

Тема 14. Аналіз ефективності алгоритмів пошуку та сортування

14.1. Аналіз ефективності алгоритмів.

Розглядається методика та інструменти для оцінки ефективності алгоритмів з метою вибору оптимального рішення для конкретної задачі.

14.2. Алгоритми пошуку.

Вивчаються різні алгоритми пошуку, такі як лінійний, бінарний та інші, та їхні особливості та ефективність в різних умовах.

14.3. Алгоритми сортування.

Розглядається ряд алгоритмів сортування, таких як QuickSort, MergeSort та інші, з фокусом на їхній реалізації, перевагах та недоліках в конкретних сценаріях використання.

Тема 15. Рекурсія та мемоізація

15.1. Рекурсивні функції.

Вивчається концепція рекурсивних функцій та їхній спосіб використання для розв'язання складних завдань шляхом викликів самої себе.

15.2. Приклади рекурсивних алгоритмів.

Досліджуються приклади рекурсивних алгоритмів, таких як факторіал, числа Фібоначчі та інші, для розкриття різноманітних застосувань рекурсії.

15.3. Порівняння рекурсії та мемоізації.

Аналізується використання мемоізації як техніки оптимізації рекурсивних алгоритмів і порівнюється їхня ефективність та способи застосування.

15.4. Перебір з поверненням.

Розглядається метод перебору з поверненням, що базується на рекурсивному підході, та його використання для розв'язання задач, які включають в себе можливі комбінації та варіації.

Тема 16. Вступ до ООП. Об’єкти та класи

16.1. Процедурний та об’єктно-орієнтований стилі програмування.

Розглядається відмінність між процедурним та об’єктно-орієнтованим стилями програмування, визначаючи основні принципи та переваги ООП.

16.2. Поняття про класи та об’єкти.

Вивчаються базові поняття ООП, такі як класи та об’єкти, їх структура, властивості та використання для моделювання реальних об’єктів або процесів.

16.3. Інкапсуляція.

Аналізується концепція інкапсуляції, яка полягає у прихованні внутрішньої реалізації об’єкта та наданні доступу лише до зовнішнього інтерфейсу.

16.4. Статичні атрибути та методи.

Розглядається використання статичних атрибутів та методів у класах, які пов'язані з класом, а не з конкретним екземпляром об’єкта.

16.5. Відношення між класами.

Досліджуються різні типи відношень між класами в ООП, такі як асоціація, композиція та наслідування, з фокусом на їхній семантиці та використанні.

Тема 17. Наслідування та поліморфізм

17.1. Наслідування.

Розглядається концепція наслідування в ООП, яка дозволяє створювати новий клас на основі вже існуючого, успадковуючи його властивості та методи.

17.2. Поліморфізм та віртуальні методи.

Вивчається поліморфізм, який дозволяє використовувати об'єкти різних класів через їхні спільні інтерфейси, а також використання віртуальних методів для реалізації поліморфізму в ООП.

17.3. Множинне наслідування.

Досліджується можливість створення класів, які успадковують властивості та методи від більше ніж одного батьківського класу, і розглядаються випадки множинного наслідування та його обмеження.

Тема 18. Спеціальні методи

18.1. Спеціальні поля та методи.

Вивчається використання спеціальних методів та полів в класах Python для перевизначення стандартної поведінки об'єктів, таких як методи __init__ та __str__, що впливають на конструювання та представлення об'єктів.

18.2. Перевантаження операторів.

Розглядається можливість визначення власного поведінки операторів для об'єктів певного класу, що дозволяє здійснювати операції, такі як додавання чи порівняння, з об'єктами цього класу.

18.3. Ітератори та генератори.

Досліджується використання ітераторів та генераторів в Python, які дозволяють послідовно отримувати значення з колекцій чи генерувати їх на льоту, спрощуючи роботу з об'єктами великого обсягу даних.

18.4. Рекурентні співвідношення.

Розглядається застосування рекурентних співвідношень у програмуванні, які описують відношення між елементами послідовностей чи структур даних, що може використовуватися для оптимізації алгоритмів та обчислень.

Тема 19. Обробка виняткових ситуацій

19.1. Помилки та винятки.

Розглядається концепція помилок в програмуванні та винятків, які можуть виникати в процесі виконання програми.

19.2. Обробка винятків.

Вивчається методика обробки виняткових ситуацій за допомогою блоку try-except, яка дозволяє визначити та виконати код для виняткових ситуацій та уникнути аварійного завершення програми.

19.3. Вкладена обробка.

Розглядається можливість вкладеного використання конструкцій try-except, що полегшує обробку різних типів виняткових ситуацій в різних частинах коду.

19.4. Повний синтаксис оператору try.

Вивчається розширений синтаксис конструкції try, який включає блоки except, else та finally, для більш гнучкого та комплексного управління винятками в програмі.

19.5. Оператор assert.

Розглядається використання оператору assert для визначення та перевірки інваріантів у програмному коді, що допомагає забезпечити коректність та стабільність роботи програми.

Тема 20. Система контролю версій Git

20.1. Розподілені системи керуваннями версіями програмного забезпечення.

Вивчається концепція розподілених систем контролю версій, які дозволяють зберігати та відстежувати історію змін в програмному коді у розподілених репозиторіях.

20.2. Система Git.

Розглядається основи використання системи контролю версій Git, зокрема команди для ініціалізації репозиторію, відстеження змін, та коміту нових версій.

20.3. Команди управління репозиторієм та роботи з гілками.

Вивчаються команди Git для управління репозиторієм, включаючи створення та об'єднання гілок, що дозволяє ефективно працювати з різними гілками розвитку проекту.

20.4. Платформа GitHub.

Розглядається використання платформи GitHub для зберігання та спільної роботи над проектами, а також можливості ведення pull request'ів та спільної роботи в команді.

Перелік лабораторних занять за навчальною дисципліною наведено в табл. 2.

Таблиця 2

Перелік лабораторних занять

Назва теми	Зміст
Тема 1.	Розв’язування завдань щодо використання базового рівня мови Python для введення та виводу даних
Тема 2.	Розв’язування завдань щодо використання умовного оператора у мові Python
Тема 3.	Розв’язування завдань щодо використання циклів з визначеною кількістю операцій у мові Python
Тема 4.	Розв’язування завдань щодо обробки символьних даних у мові Python
Тема 5.	Розв’язування завдань щодо використання циклів з перевіркою умов мовою Python
Тема 6.	Розв’язування завдань щодо використання принципів структурного програмування та функцій у мові Python
Тема 7.	Розв’язування завдань щодо використання послідовностей та списків у мові Python
Тема 8.	Розв’язування завдань щодо використання двовимірних структур даних у мові Python
Тема 9.	Розв’язування завдань щодо використання асоціативних структур даних у мові Python
Тема 10.	Розв’язування завдань щодо використання множин та колекцій у мові Python
Тема 13.	Розв’язування завдань щодо використання генераторів та колекцій у мові Python
Тема 14.	Розв’язування завдань щодо аналізу ефективності алгоритмів пошуку та сортування
Тема 15.	Розв’язування завдань щодо використання рекурсивних функцій у мові Python
Тема 16.	Розв’язування завдань щодо використання об’єктно-орієнтованого програмування
Тема 17.	Розв’язування завдань щодо обробки виняткових ситуацій у мові Python
Тема 18.	Розв’язування завдань щодо використання складних структур даних у мові Python

Перелік самостійної роботи за навчальною дисципліною наведено в табл. 3.

Таблиця 3

Перелік самостійної роботи

Назва теми	Зміст
Тема 1-10	Виконання практичних завдань для закріплення навичок програмування
Тема 12	Створення та виступ з презентацією за стандартною бібліотекою мови програмування Python
Тема 13-19	Виконання практичних завдань для закріплення навичок програмування

Кількість годин лекційних і лабораторних занять та годин самостійної роботи наведено в робочому плані (технологічній карті) з навчальної дисципліни.

МЕТОДИ НАВЧАННЯ

При викладанні навчальної дисципліни “Програмування” для активізації навчального процесу передбачено застосування сучасних навчальних технологій, таких, як: проблемні лекції; міні-лекції; презентації, виконання індивідуальних творчих завдань.

Проблемна лекція “Огляд популярних Python-фреймворків” в темі 12 спрямована на розвиток логічного мислення здобувачів. Коло питань теми обмежується двома-трьома ключовими моментами, увага здобувачів концентрується на матеріалі, що не знайшов відображення в підручниках, використовується досвід закордонних навчальних закладів з роздачею здобувачам під час лекцій друкованого матеріалу та виділенням головних висновків з питань, що розглядаються. При викладанні лекційного матеріалу здобувачам пропонуються питання для самостійного розмірковування. При цьому лектор задає запитання які спонукають здобувача шукати розв’язання проблемної ситуації. Така система примушує здобувачів сконцентруватися і почати активно мислити в пошуках правильної відповіді.

На початку проведення проблемної лекції необхідно чітко сформулювати проблему, яку необхідно вирішити здобувачам. При викладанні лекційного матеріалу слід уникати прямої відповіді на поставлені запитання, а висвітлювати матеріал таким чином, щоб отриману інформацію здобувач міг використовувати при розв’язанні проблеми.

Міні-лекція “Сховище репозиторіїв GitHub” в темі 20 передбачає викладення навчального матеріалу за короткий проміжок часу й характеризуються значною ємністю, складністю логічних побудов, доказів та узагальнень. Міні-лекції проводяться, як правило, як частина заняття-дослідження. На початку проведення міні-лекції за вказаною вище темою лектор акцентує увагу здобувачів на необхідності представити викладений лекційний матеріал у так званому структурно-логічному вигляді. На розгляд виносяться питання, які зафіксовані у плані лекцій, але викладаються вони стисло. Лекційне заняття, проведене у такий спосіб, побуджає у здобувача активність та увагу при сприйнятті матеріалу, а також спрямовує його на використання системного підходу при відтворенні інформації, яку він одержав від викладача.

Презентації результатів виконання самостійного завдання щодо стандартної бібліотеки Python за темою 12 – це виступи перед аудиторією, що використовуються для представлення звіту про виконання індивідуальних завдань. Однією з позитивних рис презентації та її переваг при використанні в навчальному процесі є обмін досвідом, який здобули здобувачі при роботі над індивідуальним завданням або у певній малій групі.

ФОРМИ ТА МЕТОДИ ОЦІНЮВАННЯ

Університет використовує накопичувальну (100-бальну) систему оцінювання.

Система оцінювання сформованих компетентностей у здобувачів враховує види занять, які згідно з програмою навчальної дисципліни передбачають лекційні, лабораторні заняття, а також виконання самостійної роботи. Контрольні заходи включають:

поточний контроль, що здійснюється протягом семестру під час проведення лекційних, лабораторних занять і оцінюється сумою набраних балів (максимальна сума – 60 балів; мінімальна сума, що дозволяє здобувачі складати іспит, – 35 балів);

модульний контроль, що проводиться у формі поточних контрольних робіт за змістові модулі та має на меті *інтегровану* оцінку результатів навчання здобувача після вивчення матеріалу з логічно завершеної частини дисципліни – змістового модуля 1 та 2;

підсумковий контроль, що проводиться у формі екзамену під час семестрової сесії, відповідно до графіку навчального процесу.

Поточний та модульний контроль оцінювання знань протягом змістових модулів включає:

виконання самостійних індивідуальних завдань. Загальна кількість балів – 40;

презентація результатів виконання самостійного завдання щодо бібліотеки Python – 5 балів;

поточні контрольні роботи – 15 балів.

Підсумковий контроль знань та компетентностей здобувачів з навчальної дисципліни здійснюється на підставі проведення семестрового екзамену, завданням якого є перевірка розуміння здобувачів програмного матеріалу в цілому, логіки та взаємозв'язків між окремими розділами, здатності творчого використання накопичених знань, вміння формулювати своє ставлення до певної проблеми навчальної дисципліни тощо.

Більш детальну інформацію щодо системи оцінювання наведено в робочому плані (технологічній карті) з навчальної дисципліни.

Приклад екзаменаційного білету

Стереотипні завдання

Питання 1. Після виконання інструкції pow(3,3), pow(5,2) буде отримано:
Виберіть одну відповідь:
a. (27,25)
b. (6,7)
c. (9,10)

Питання 2. Після виконання інструкції math.floor(7.81) буде отримано:
Виберіть одну відповідь:
a. 7.8
b. 8

c. 7
Питання 3. У програмі, що виконується:
Виберіть одну відповідь:
a. не існує синтаксичних помилок
b. не існує логічних помилок
c. існують синтаксичні помилки
d. не існує ніяких помилок
Питання 4. Мовою програмування високого рівня називають мову, яка:
Виберіть одну або декілька відповідей:
a. не пов’язана з конкретним комп’ютером
b. використовує десяткову систему числення
c. враховує структуру комп’ютера
d. доступна для широкого кола користувачів
Питання 5. Оператор

```
if x==2:
   y=x
else:
   y=3.2*x
```

записаний:
Виберіть одну відповідь:
Правильно
Неправильно
Питання 6. Оператор

```
if x<=5:
   y=x+3 z=x-2
else:
   y=x/2.5
```

записаний:
Виберіть одну відповідь:
Правильно
Неправильно
Питання 7. Фрагмент коду

```
for i in range (5)
   k=k+1
```

записаний:
Виберіть одну відповідь:
Правильно
Неправильно
Питання 8. Після виконання інструкції 45<=15 буде отримано:
Виберіть одну відповідь:
a. False
b. True
Питання 9. Фрагмент коду

```
for i in range(5):
   i=i+1
   s=5*i
```

записаний:
Виберіть одну відповідь:
Правильно
Неправильно
Питання 10. Оператор continue:

Виберіть одну відповідь:
a. завершує виконання програми
b. перериває цикл і повертає управління на початок циклу
Питання 11. Після виконання інструкції 11.0%3.0 буде отримано:
Виберіть одну відповідь:
a. 3.0
b. 2.0
c. 2
Питання 12. У мові Python для виділення блоків команд використовуються:
Виберіть одну відповідь:
a. конструкція begin…end
b. квадратні дужки
c. відсутні спеціальні засоби
d. фігурні дужки
Питання 13. Оператор
if x<3:
  y=x+4 z=x-2
записаний:
Виберіть одну відповідь:
Правильно
Неправильно
Питання 14. Після виконання інструкції x_3=30.7 змінна x_3 матиме тип:
Виберіть одну відповідь:
a. float
b. str
c. int
Питання 15. Основними складовими мов програмування є:
Виберіть одну або декілька відповідей:
a. алфавіт
b. синтаксис
c. команди
d. повідомлення
Питання 16. Які типи об’єктів подано правильно?
Виберіть одну або декілька відповідей:
a. list
b. bools
c. int
d. string
Питання 17. Після виконання інструкції a1,a2,a3=21,42,63 змінна a2 матиме значення:
Виберіть одну відповідь:
a. 21,42
b. 21,42,63
c. 42
d. 42, 63
Питання 18. Проект мовою Python може складатися:
Виберіть одну відповідь:
a. із будь-якої кількості модулів
b. лише з одного модуля
c. лише з двох модулів
d. із вкладених модулів
Питання 19. Після виконання інструкції "принтер" not in "миша" буде отримано:
Виберіть одну відповідь:

a. True
b. False
Питання 20. Динамічна типізація змінних вимагає:
Виберіть одну відповідь:
a. оголошувати типи змінних
b. не присвоювати змінним дані
c. не оголошувати тип змінних

Діагностичні завдання

1. Дано дві точки: A (x1, y1) і B (x2, y2). Напишіть програму, яка визначає, яка із точок знаходиться далі від початку координат.

2. Напишіть програму для побудови шаблону за введеним значенням n.

Евристичне завдання

1. Вводяться n рядків. Визначити найдовший рядок і вивести його номер на екран. Якщо найдовших рядків кілька, то вивести номери всіх таких рядків.

Критерії оцінювання

Екзаменаційний білет охоплює програму дисципліни і передбачає визначення рівня знань та ступеня опанування здобувачами компетентностей. Кожен екзаменаційний білет складається із 20 стереотипних тестових завдань та 2 практичних завдань (діагностичне та евристичне завдання), які передбачають вирішення типових професійних завдань фахівця на робочому місці та дозволяють діагностувати рівень теоретичної підготовки здобувача і рівень його компетентності з навчальної дисципліни.

Екзаменаційний білет включає:

Стереотипні тестові завдання: максимальна кількість балів – 20.

Діагностичне завдання: максимальна кількість балів – 10.

Евристичне завдання: максимальна кількість балів – 10.

Загальна оцінка формується за наступною формулою:

кількість балів, отриманих за виконання кожного стереотипного завдання +

кількість балів, отриманих за виконання кожного діагностичного завдання +

кількість балів, отриманих за виконання евристичного завдання

1. Виконання стереотипного завдання полягає у розв’язанні 20 тестових запитань закритого типу, кожне з яких оцінюється в один бал.

2. Виконання діагностичного завдання полягає у розв’язанні двох завдань на програмування з використанням базових алгоритмічних конструкцій, кожне з яких оцінюється в п’ять балів. Потрібно надати розв’язок завдань у вигляді текстів програм мовою Python. Програмний код повинен супроводжуватися коментарями, що пояснюють роботу програми. За неповного виконання завдання чи помилок у програмному коді оцінка за виконання буде пропорційно зменшена.

3. Виконання евристичного завдання полягає у розв’язанні завдання на програмування з використанням базових алгоритмічних конструкцій та структур даних, що оцінюється в десять балів. Потрібно надати розв’язок завдання у вигляді тексту програми мовою Python. Програмний код повинен супроводжуватися коментарями, що пояснюють роботу програми. За неповного виконання завдання чи помилок у програмному коді оцінка за виконання буде пропорційно зменшена.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Васильєв О.М. Програмування мовою Python. – Л.: Bohdan Books, 2022. – 504 с.
2. Беррі П. Head First. Python. – Х.: ФАБУЛА, 2021. – 624 с.
3. Маттес Е. Пришвидшений курс Python. – Л.: Видавництво Старого Лева, 2021. – 600 с.
4. Щербаков, О. В. Основи об'єктно-орієнтованого програмування [Електронний ресурс] : навч. посіб. / О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко ; Харківський національний економічний університет ім. С. Кузнеця. - Електрон. текстові дан. (2,13 МБ). - Харків : ХНЕУ ім. С. Кузнеця, 2019. - 236 с. : іл. - Загол. з титул. екрану. - Бібліогр.: с. 231-232.- Режим доступу: http://www.repository.hneu.edu.ua/handle/123456789/23847
5. Щербаков О. В. Сучасні тенденції розвитку мов програмування на прикладі Java та C# / О.В. Щербаков, Ю.І. Скорін // Інформаційні технології та системи : матеріали міжнар. наук.-практ. конф., 8 – 9 квіт. 2021 р. : тези допов. – Харків : ХНЕУ ім. С. Кузнеця, 2021. – С. 36. - Режим доступу: http://www.repository.hneu.edu.ua/handle/123456789/25604

Додаткова

6. Фрімен Е., Робсон Е. Head First. Патерни проєктування. – Х.: ФАБУЛА, 2020. – 672 с.
7. Мартін Р. Чистий кодер. – Х.: ФАБУЛА, 2023. – 256 с.
8. Креневич А.П. Python у прикладах і задачах. Частина 1. Структурне програмування Навчальний посібник із дисципліни "Інформатика та програмування" – К.: ВПЦ "Київський Університет", 2017. – 206 с.
9. Креневич А.П. Python у прикладах і задачах. Частина 2. Об’єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 с.
10. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки", спеціалізації "Інформаційні технології в біології та медицині" / А. В. Яковенко. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 195 с.

Інформаційні ресурси

11. Яценко Р.М. Персональна навчальна система з навчальної дисципліни «Програмування» [Електронний ресурс]. – Режим доступа : https://pns.hneu.edu.ua/course/view.php?id=8027.
12. W3Schools Online Web Tutorials Distribution [Електронний ресурс]. – Режим доступа : https://www.w3schools.com/.
13. Heroku: Cloud Application Platform [Електронний ресурс]. – Режим доступа : https://www.heroku.com/.

14. Сайт Національної бібліотеки України ім. Вернадського [Електронний ресурс]. – Режим доступа : www.nbuv.gov.ua.
15. Сайт Python Programming Language [Електронний ресурс]. – Режим доступа : https://www.python.org/.
16. Teach Python 3 and web design with 200+ exercises [Електронний ресурс]. – Режим доступа : https://snakify.org/en/