МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

Факультет механіки, енергетики
та інформаційних технологій

Кафедра інформаційних
технологій

Методичні рекомендації
до вивчення дисципліни

«Об’єктно-орієнтоване програмування»

для студентів спеціальностей
122 "Комп’ютерні науки" та
126 "Інформаційні системи та технології"

Львів – 2024

Рекомендовано до друку
методичною радою університету
Протокол №__ від _____ р.

Укладач: к.т.н., в.о. доц. Татомир А.В.

Рецензент: _____

Видається в авторській редакції

© Львівський національний аграрний університет, 2024

МЕТА І ЗАВДАННЯ ВИВЧЕННЯ ДИСЦИПЛІНИ

Метою вивчення дисципліни є набуття теоретичних та практичних знань про специфіку об'єктно-орієнтованої розробки додатків, а також формування у фахівців з інформаційних технологій знань та вмінь професійного використання принципів об'єктно орієнтовано програмування. Додатково передбачається вивчення основних принципів і особливостей інших сучасних парадигм програмування.

Завдання вивчення дисципліни:

основними завданнями вивчення дисципліни "Об'єктно-орієнтоване програмування" є підготовка фахівців, які здатні забезпечити самостійне розв'язування інженерних задач, пов'язаних із аналізом та проектуванням складних додатків, розвинення у студентів бачення переваг та недоліків застосування різних парадигм програмування, в першу чергу – об'єктно-орієнтованого програмування.

ВИМОГИ ДО ЗНАНЬ, УМІНЬ ТА НАВИЧОК

В результаті вивчення дисципліни студент повинен знати:

- сучасні парадигми програмування;
- основи синтаксису об'єктно-орієнтованої мови програмування Python;
- методи та засоби об'єктно-орієнтованого програмування;
- поширені шаблони проєктування програмного забезпечення.

В результаті вивчення дисципліни студент повинен вміти:

- розрізняти стилі написання комп'ютерних програм та вміти вибирати необхідний стиль залежно від поставленої задачі;
- застосовувати методи та прийоми об'єктно-орієнтованого програмування;
- проектувати, оформляти та розгортати додатки довільної складності;
- володіти навичками ієрархічного аналізу даних та проектування структур даних;
- застосовувати сучасні технології розробки комп'ютерних програм, зокрема об'єктно-орієнтоване програмування.

АНОТАЦІЯ ДИСЦИПЛІНИ

Дисципліна “Об’єктно-орієнтоване програмування” вивчає способи організації програмного коду для полегшення його написання, відлагодження, підтримки та розширення.

Під час вивчення дисципліни студенти отримують знання та вміння, спрямовані на підвищення якості написання коду відповідно до найкращих сучасних практик. В процесі навчання вони знайомляться з популярною об’єктно-орієнтованою мовою програмування Python, отримують знання про принципи об’єктно-орієнтованого програмування та вчаться організовувати програмний код відповідно до існуючих виробничих стандартів.

ТЕМАТИКА І КОРОТКИЙ ЗМІСТ ЛЕКЦІЙНИХ ЗАНЯТЬ

Тема 1

«Вступ»

Мета, структура і предмет курсу.

Парадигми і мови програмування.

Передумови виникнення ООП. Історичний процес розвитку підходів до програмування. Структуроване програмування. Процедурне програмування. Складність коду. Концепція об’єктно-орієнтованого програмування.

Переваги та недоліки ООП. Альтернативи ООП. Продовження і варіянти реалізації ООП у різних мовах.

Тема 2

«Системи контролю версій програмного забезпечення»

Концепція версійности програмного коду. Відслідковування змін в коді. Пошук несправностей програмного забезпечення.

Система контролю версій Git. Основні команди та функціонал.

Розширене використання систем контролю версій. Робота з віддаленими репозиторіями. Колаборація.

Тема 3

«Загальний огляд об’єктно-орієнтованої мови програмування Python. Запуск і робота з Python»

Налаштування робочого середовища Python 3 для систем на базі Unix. Особливості роботи у Windows. Редактори коду та інтегровані середовища розробки.

Поняття компіляції та интерпретації. Динамічна типізація.

Змінні та оператори.

Базові та комплексні типи даних. Вбудовані типи даних у Python 3. Списки (Lists) та словники (Dictionaries). Розширення типів. Формат даних JSON.

Базові оператори для роботи з простими типами даних (математичні та логічні операції, робота зі стрічками). Приведення типів.

Тема 4

«Структурне і процедурне програмування»

Складні алгоритми та програми. Історичні витоки появи структурного та процедурного програмування.

Конструкції для підтримки структурованого програмування. Лінійне виконання. Оператори розгалуження. Цикли в Python 3. Переваги та недоліки структурованого програмування.

Процедурне програмування. Функції та процедури. Оголошення та виконання функцій. Аргументи і параметри функцій. Рекурсія.

Тема 5

«Принцип інкапсуляції даних»

Класифікація об’єктів. Стан об’єкта. Створення літералів об’єктів.

Взаємозв’язки між класами і об’єктами. Класи як типи даних.

Атрибути (властивості, поля) об’єктів. Змінні класу.

Оголошення методів. Створення об’єкта (class instance) за допомогою класу (конструктори і деструктори). Звертання класу до об’єкта через параметр “self”. Методи об’єкта. Методи класу (параметр “cls”). Статичні методи.

Безпечне звертання до атрибутів. Концепція публічних, приватних та захищених полів об’єктів.

Тема 6

«Наслідування в ООП»

Взаємозв’язки між класами. Оголошення наслідування.

Перевикористання властивостей та методів (resolution order). Глобальний (вбудований) object.

Пряме та непряме наслідування. Функція super.

Множинне успадкування в Python.

Розуміння наслідування (функції help, isinstance, issubclass).

Формальне представлення наслідування. UML-діаграми класів.

Особливості прототипно-орієнтованих мов програмування.

Тема 7

«Поліморфізм»

Перевантаження методів. Способи організації поліморфізму.

“Магічні методи” в Python 3 та їх використання. Основні вбудовані оператори, що допускають перевизначення.

Декоратори. Getter’и, setter’и та deleter’и.

Тема 8

«Модулі та пакети»

Базове ядро Python 3.

Розширення функціональності за рахунок перевикористання стороннього коду. Імпорт пакетів. Роботи з PIP. Open source software.

Написання та підключення власних модулів.

Тема 9

«Шаблони (патерни) проєктування програмного забезпечення»

Поняття шаблону проєктування. Передумови та історія появи.

Основні групи шаблонів (твірні, структурні, поведінкові).

Використання шаблонів у виробництві.

Тема 10

«Твірні шаблони проєктування»

Основні твірні шаблони. Розгляд базового прикладу.

Тема 11

«Структурні шаблони проєктування»

Основні структурні шаблони. Розгляд базового прикладу.

Тема 12

«Поведінкові шаблони проєктування»

Основні поведінкові шаблони. Розгляд базового прикладу.

Тема 13

«Принципи проєктування програмного забезпечення»

Практичні та теоретичні принципи проєктування. DRY, AHA, KISS тощо.

Комплекс принципів проєктування SOLID. Принцип єдиної відповідальності. Принцип відкритості/закритості. Принцип підстановки Лісков. Принцип розділення інтерфейсу. Принцип інверсії залежностей.

Тема 14

«Якість коду. Рефакторинг»

Поняття якості коду. Складові якості коду (читабельність, розширюваність, підтримуваність, testability тощо). “Чистий” та “брудний” код. ”Code smells”. Досвід розробників програмного забезпечення (developer experience).

Процес рефакторингу. Тестування в процесі рефакторингу. Використання систем контролю версій. Техніки рефакторингу (перейменування, переміщення, спрощення або переписування тощо).

Статична перевірка коду.

Тестування програмного забезпечення. Види тестування. Автоматичне тестування. Unit, integration та E2E тестування.

Тема 15

«Переваги декларативного стилю програмування. Загальні відомості про функціональне програмування»

Відмінності між імперативним та декларативним стилем написання програм. Когнітивна складність коду.

Загальні відомості про лямбда-числення.

Основні положення функціонального програмування.

Відмінності між ООП та ФП. Імутабельність. Композиція замість наслідування.

Використання функціонального програмування в Python 3. Лямбда-функції.

Тема 16

«Застосування ООП та інших парадигм програмування в реальному виробництві. Підведення підсумків»

Сучасні підходи до розробки програмного забезпечення. Мультипарадигменна розробка.

Якість коду. Статична перевірка коду.

Тестування програмного забезпечення. Види тестування. Автоматичне тестування. Unit, integration та E2E тестування.

ПЕРЕЛІК ЗАПИТАНЬ НА ІСПИТ

1. Передумови виникнення ООП.
2. Основні принципи ООП. Сфери застосування ООП.
3. Сучасні парадигми програмування.
4. Відмінності процедурного та об’єктно-орієнтованого програмування.
5. Класи та об’єкти.
6. Визначення та оголошення класу. Вбудовані функції класу.
7. Життєвий цикл об’єкта. Конструктори та деструктори.
8. Принцип інкапсуляції даних.
9. Атрибути та методи об’єктів.
10. Змінні класу та об’єкта.
11. Інтерфейс класу. Реалізація класу.
12. Методи класу, статичні методи. Альтернативні конструктори.
13. Приховування даних.
14. Принцип наслідування. Ієрархія об’єктів і класів.
15. Поліморфізм.
16. Перевантаження операторів.
17. Загальна характеристика мови програмування Python, основні її концепції. Об’єктно-орієнтоване програмування у Python 3.
18. Способи створення об’єктів.
19. Поточний об’єкт self.
20. Ідентичність об’єктів. Копіювання, ініціалізація і присвоєння об’єктів.
21. Тривалість життя змінних. Автоматичне і примусове очищення пам’яті.
22. Області видимості змінних класу, об’єкта, методу.

23. Ліквідація об’єктів. Властивості деструкторів.
24. Get-тери і set-тери.
25. Базовий та дочірній класи. Перевірка приналежності об’єкта до класу та дочірнього класу до батьківського.
26. Пряме та непряме наслідування. Функція super.
27. Приватні та захищені властивості класів.
28. “Магічні методи”.
29. Декоратори методів.
30. Модулі та пакети.
31. Шаблони проєктування. Основні групи шаблонів.
32. Твірні шаблони проєктування. Опишіть приклад твірного шаблону.
33. Структурні шаблони проєктування. Опишіть приклад структурного шаблону.
34. Поведінкові шаблони проєктування. Опишіть приклад поведінкового шаблону.
35. Загальні принципи проєктування. DRY, WET тощо.
36. Принципи проєктування ООП. Принцип єдиної відповідальності.
37. Принципи проєктування ООП. Принцип відкритості/закритості.
38. Принципи проєктування ООП. Принцип підстановки Лісков.
39. Принципи проєктування ООП. Принцип розділення інтерфейсу.
40. Принципи проєктування ООП. Принцип інверсії залежностей.
41. Поняття якості коду. Складові якості коду.
42. Рефакторинг. Запахи коду.
43. Рефакторинг. Основні техніки рефакторингу.

ЗАДАЧІ

1. Реалізувати лічильник створених об’єктів класу. Передбачити можливість видалення об’єктів та виведення необхідної інформації.
2. Запрограмувати двозв’язний список, включаючи основні методи для роботи з ним (ініціалізація, перехід до наступного/попереднього значення, виведення поточного елемента).
3. Описати клас для структури даних типу стек. Передбачити додавання, видалення елементів, виведення необхідних значень.
4. Реалізувати клас для опису фігури “прямокутник”. Запрограмувати методи для обчислення його геометричних характеристик. Реалізувати наслідування для опису фігури “квадрат”.
5. Описати клас для роботи з фігурою “прямокутний трикутник, заданий катетами”. Реалізувати методи для обчислення основних геометричних та тригонометричних параметрів.
6. Реалізувати клас для роботи з комплексними числами. Операції визначити через перевантаження операторів за загальноприйнятими формулами.
7. Реалізувати клас для роботи з двовимірними векторами в декартових координатах. Операцію додавання векторів визначити через перевантаження операторів за загальноприйнятими формулами.
8. Реалізувати клас для представлення векторів у декартових та полярних координатах. Організувати конструктори для задання векторів обома способами та передбачити переведення векторів з одного представлення в інше.
9. Реалізувати клас для роботи з дробовими числами. Реалізувати

можливість представлення числа в експоненційній формі.

10. Реалізувати клас для роботи з функціями. Клас повинен мати конструктор для задання довільної функції та методи для обчислення значення функції в точці і наближеного значення похідної функції в точці.
11. Реалізувати клас для статистичного опрацювання даних спостережень. Передбачити можливість додавання нових значень спостережень та обчислення середнього арифметичного для наявної вибірки.
12. Реалізувати клас для роботи з множинами. Операції (щонайменше - об’єднання та перетин) визначити через перевантаження операторів за загальноприйнятими формулами.
13. Запрограмувати адресну книгу, кожен запис якої повинен містити ім’я, прізвище, повне ім’я та номер телефону. Реалізувати альтернативний конструктор з можливістю задання початкових даних текстовою стрічкою. Передбачити можливість зміни даних.
14. Реалізувати клас, що описує функціонал двостороннього словника. Слова в кожній з мов можуть мати по декілька синонімів.
15. Запрограмувати модель організаційної структури типу “працівник - управлінець”. Окрім спільної загальної інформації, управлінець повинен володіти інформацією про працівників, за яких він відповідає. Реалізувати методи підпорядкування, перепідпорядкування та виведення інформації.

ТЕМАТИКА І КОРОТКИЙ ЗМІСТ ПРАКТИЧНИХ ЗАНЯТЬ

ТЕМА 1

Вступне заняття. Налаштування середовища розробки Python 3 і запуск програм

Мета

Мета роботи – ознайомитися інструментарієм розробки веб-додатків.

Завдання

1. Навчитися встановлювати середовище розробки Python 3 в різних операційних системах і запускати програми.
2. Написати базову програму згідно виданого завдання та проаналізувати її роботу.
3. На прикладі написаної програми навчитися застосовувати систему Git для контролю версій програмного забезпечення.

Запитання для самоконтролю

1. Як встановити середовище розробки Python? Як запустити виконання програми?
2. Які основні відмінності в роботі під ОС Windows та Unix-подібнними системами?
3. Поясніть структуру найпростішої програми на Python 3.

4. Як здійснюються базові операції вводу-виводу в Python 3? Які ще часто використовувані операції (і відповідні оператори Python 3) Вам відомі?
5. Які Ви знаєте команди Git?

ТЕМА 2

Вивчення простих типів даних і методів роботи з ними у Python 3

Мета

Метою роботи є вивчення основ розробки додатків на Python 3.

Завдання

1. Вивчити основи синтаксису текстів програм в Python 3.
2. Розглянути поняття змінних в Python 3 та засвоїти їх прості типи.
3. Розглянути поняття динамічної типізації.
4. Навчитися здійснювати базові операції та приведення типів. Розв’язати приклад згідно виданого завдання.
5. Ознайомитися з типом “List”. Навчитися задавати та зчитувати значення їх елементів. Розв’язати приклад.
6. Ознайомитися основними операторами мови Python 3. Розв’язати завдання.
7. Навчитися основам форматування стрічок і текстового виводу.
8. Освоїти роботу зі стрічками як зі списками (“Lists”).
9. Ознайомитися з типом даних “словник” (“Dictionary”). Розв’язати завдання.

Запитання для самоконтролю

1. Що таке індентація? Для чого індентація використовується в Python 3?
2. Як задати змінну в Python 3? Як задати її тип? Чи можна змінити тип змінної? Чи можна задати значення декільком змінним одночасно?

3. Перелічіть відомі Вам прості типи змінних.
4. Як здійснюється приведення типів у Python 3?
5. Що таке список (list)? Наведіть основні оператори для роботи зі списками.
6. Наведіть приклади базових операторів для роботи з числовими змінними.
7. Які операції можна здійснювати над стрічками (string)? Які Ви знаєте способи їх форматування?
8. Що таке словники (dictionaries)? Як задавати та зчитувати значення їх елементів?
9. В чому полягає різниця в роботі з списками та словниками? На Ваш погляд, який з цих типів більш універсальний? Чи можна котрийсь із них отримати завдяки наслідуванню іншого? Як саме?

ТЕМА 3

Основи структурного програмування в Python 3

Мета

Мета роботи – ознайомлення основними прийомами структурного програмування у Python 3.

Завдання

1. Засвоїти суть структурного програмування.
2. Навчитися використовувати умовні оператори
3. Познайомитися з операторами булевої логіки в Python 3.
4. Навчитися проектувати складні умови.
5. Опанувати методи роботи з циклами. Навчитися визначати доцільність застосування різних типів циклів залежно від поставленої задачі.
6. Розв’язати приклади згідно виданих завдань.

Запитання для самоконтролю

1. Дайте визначення поняттю “структурне програмування”.
2. Який синтаксис умовних операторів у Python 3? Наведіть приклади відомих Вам умовних конструкцій.
3. Що таке булеві оператори?
4. Які логічні оператори можуть застосовуватися до змінних типу список (list)? В чому різниця між операторами “==” та “is”?

5. Що таке цикл? Що таке умова та тіло циклу?
6. Перелічіть відомі Вам способи організації циклів. Коли доцільно використовувати кожен з них?
7. Що таке оператори модифікації циклів?

ТЕМА 4

Основи процедурного програмування в Python 3

Мета

Мета роботи полягає у засвоєнні студентами методів та прийомів роботи з функціями.

Завдання

1. Освоїти синтаксис функцій в Python 3.
2. Закріпити поняття параметрів та аргументів функції.
3. Вивчити методи роботи з функціями, викликати функції, передавати їх в якості параметрів у інші функції (“callback”).
4. Розв’язати заданий приклад.

Запитання для самоконтролю

1. Назвіть переваги використання функцій.
2. Як оголосити функцію в Python 3?
3. Як викликати функцію?
4. Що таке аргументи функції? Що таке параметри функції?
5. Як задати аргументам функції за замовчуванням?
6. Що таке функції першого класу?
7. Що таке функції вищого порядку?

ТЕМА 5

Створення та використання класів.

Мета

Мета роботи – ознайомитися з поняттями класів та об’єктів та закріпити на практиці методи їх створення та використання.

Завдання

1. Навчитися оголошувати класи в Python 3.
2. Навчитися створювати об’єкти (“class instances”).
3. Ознайомитися з поняттям змінної (властивості) об’єкту. Навчитися їх задавати та отримувати їхні значення.
4. Ознайомитися з поняттям функцій об’єкту. Навчитися їх оголошувати та викликати.
5. Розв’язати приклад згідно виданого завдання.

Запитання для самоконтролю

1. Дайте визначення поняттю “клас”. Як оголосити найпростіший клас в Python 3?
2. Що таке об’єкт? Як створюється об’єкт?
3. В чому полягає різниця між класом і об’єктом?
4. Що таке змінна об’єкту?
5. Як викликати функцію класу?
6. Як імпортувати клас з зовнішнього файлу?
7. Яку відповідність з реального світу Ви могли б навести для понять

класу та об’єкта?

ТЕМА 6

Змінні класу та об’єкта

Мета

Мета роботи – ознайомитися з різними типами змінних в об’єктно-орієнтованому програмуванні

Завдання

1. Набути навичок у створенні класів. Створити клас, який приймає декілька аргументів. На їх основі у конструкторі класу створити набір атрибутів об’єкта (класу), один з яких створюється на основі інших.
2. Реалізувати метод, який генерує опис об’єкта на основі його властивостей.
3. Навчитися створювати об’єкти. Створити декілька об’єктів на основі класу. Викликати реалізований метод, використовуючи об’єкт і клас.
4. Познайомитися з поняттям змінної класу. Реалізувати змінну класу і метод, що її використовує.
5. Реалізувати лічильник створених за допомогою класу об’єктів.

Запитання для самоконтролю

1. Як створити порожній клас?
2. Який метод призначений для створення нового об’єкта?
3. Що виведеться у консоль, якщо виконати команду “print(obj.method)”

(метод без лапок)?

4. Яка різниця між змінною об’єкта і класу?

ТЕМА 7

Використання методів класу і статичних методів

Мета

Мета роботи полягає в ознайомленні з різними типами методів у об’єктно-орієнтованому програмуванні.

Завдання

1. Засвоїти різницю між звичайними методами, методами класу та статичними методами.
2. Навчитися створювати альтернативні конструктори.
3. Для створеного у попередній роботі класу реалізувати “метод класу”, який повинен працювати зі змінними класу.
4. Реалізувати альтернативний конструктор класу за допомогою методу класу.
5. Створити статичний метод і перевірити його роботу.

Запитання для самоконтролю

1. В чому полягає різниця між звичайними методами та методами класу?
2. Що таке “альтернативний конструктор”? Для чого його можна састосовувати?
3. Для чого потрібні статичні методи? Чи можливо обійтися без використання статичних методів у ООП?

ТЕМА 8

Наслідування в об’єктно-орієнтованому програмуванні

Мета

Мета роботи - оволодіти концепцією наслідування класів.

Завдання

1. Навчитися повторно використовувати існуючий код завдяки наслідуванню.
2. Створити один батьківський клас і декілька дочірніх. Наслідувати частину властивостей та функціоналу від батьківського класу в дочірніх класах. Освоїти використання методу *super*.
3. У одному з дочірніх класів організувати використання методів об’єктів-представників іншого дочірнього класу.
4. Ознайомитися з методами *instanceof, issubclassof.*

Запитання для самоконтролю

1. Дайте визначення поняттю “наслідування” в об’єктно-орієнтованому програмуванні.
2. Для чого використовується наслідування?
3. Що таке “батьківський” і “дочірній” класи?
4. Опишіть функціонал методу *super.*
5. Як звернутися до класу, що має спільного із поточним класом предка?

ТЕМА 9

Поліморфізм в Python 3

Мета

Мета роботи - засвоїти застосування принципу поліморфізму в об’єктно-орієнтованому програмуванні.

Завдання

1. Ознайомитися з поняттям поліморфізму в ООП.
2. Навчитися перевизначати поведінку методів.
3. Реалізувати декілька “магічних методів” для роботи з визначеними раніше класами.

Запитання для самоконтролю

1. Поясніть концепцію поліморфізму.
2. Наведіть приклад застосування поліморфізму.
3. Що таке “магічні методи”? Які з них Вам відомі?

ТЕМА 10

Використання декораторів методів

Мета

Мета роботи - освоїти роботу з декораторами в Python 3.

Завдання

1. Навчитися використовувати декоратор “@property”. Написати принаймні один метод з його використанням.
2. Ознайомитися з концепцією *getter*’ів, *setter*’ів і *delete*’рів. Реалізувати хоча б один метод з їх використанням.

Запитання для самоконтролю

1. Що таке “декоратор методу”?
2. Які декоратори Вам відомі?
3. Для чого використовуються *getter*’и і *setter*’и? В чому їх перевага над іншими способами доступу до властивостей об’єктів?

ТЕМА 11

Твірні шаблони проєктування

Мета

Мета роботи - познайомитися з групою твірних шаблонів проєктування.

Завдання

1. Дати теоретичний опис твірної групи шаблонів.
2. Відповідно до индивідуального завдання:
    - дати теоретичний опис даного шаблону;
    - навести приклад коду який реалізовує даний шаблон;
    - скласти його UML-діяграму.

Запитання для самоконтролю

1. Що таке твірні шаблони?
2. Які твірні шаблони Вам відомі?
3. Поясніть як реалізовано шаблон у Вашому прикладі.

ТЕМА 12

Структурні шаблони проєктування

Мета

Мета роботи - познайомитися з групою структурних шаблонів проєктування.

Завдання

1. Дати теоретичний опис структурної групи шаблонів.
2. Відповідно до индивідуального завдання:
    - дати теоретичний опис даного шаблону;
    - навести приклад коду який реалізовує даний шаблон;
    - скласти його UML-діяграму.

Запитання для самоконтролю

1. Що таке структурні шаблони?
2. Які структурні шаблони Вам відомі?
3. Поясніть як реалізовано шаблон у Вашому прикладі.

ТЕМА 13

Поведінкові шаблони проєктування

Мета

Мета роботи - познайомитися з групою поведінкових шаблонів проєктування.

Завдання

1. Дати теоретичний опис поведінкової групи шаблонів.
2. Відповідно до индивідуального завдання:
    - дати теоретичний опис даного шаблону;
    - навести приклад коду який реалізовує даний шаблон;
    - скласти його UML-діяграму.

Запитання для самоконтролю

1. Що таке поведінкові шаблони?
2. Які поведінкові шаблони Вам відомі?
3. Поясніть як реалізовано шаблон у Вашому прикладі.

ТЕМА 14

Принципи проєктування програмного забезпечення

Мета

Мета роботи - познайомитися з найбільш поширеними сучасними принципами проєктування програмного забезпечення.

Завдання

1. Дати загальний опис принципів проєктування.
2. Дати детальний опис одного із принципів SOLID з наведення прикладу коду.

Запитання для самоконтролю

1. Що таке принципи проєктування? Які принципи Вам відомі?
2. Що таке принципи SOLID? Опишіть відомі Вам принципи детальніше.

ТЕМА 15

Рефакторинг програмного забезпечення

Мета

Мета роботи - познайомитися з основними принципами та найбільш поширеними техніками рефакторингу програмного забезпечення.

Завдання

1. Дати загальний опис принципів рефакторингу.
2. Ознайомитися із основними техніками рефакторингу. Детально описати одну із технік.
3. Познайомитися із поняттям “запахів коду”. Дати детальний опис одного із “запахів”.

Запитання для самоконтролю

1. Що таке рефакторинг коду? Які принципи рефакторингу Ви знаєте?
2. Опишіть відомі Вам техніки рефакторингу.
3. Перелічіть “запахи коду”, з якими Ви стикалися.
4. Як забезпечити збереження наявного функціоналу в процесі рефакторингу?

ТЕМА 16

Тестування програмного коду

Мета

Мета роботи - ознайомитися із техніками тестування програмного коду.

Завдання

1. Ознайомитися із видами тестів та пірамідою тестування.
2. Написати та виконати unit-тест на Python 3 використовуючи вбудований модуль unittest.

Запитання для самоконтролю

1. Для чого потрібне тестування?
2. Які види тестування Вам відомі? Яка між ними різниця? Наведіть переваги та недоліки кожного із них.
3. Як писати та виконувати unit-тести на Python 3?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

БАЗОВА

1. Онлайн підручник з Python 3 – офіційна документація для розробників: https://docs.python.org/uk/3/tutorial/index.html
2. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. / Гради Буч и др. - М.: "И.Д. Вильямс", 2008. - 720 с.: ил.
3. Патерни проектування: https://refactoring.guru/uk/design-patterns

ДОДАТКОВА

4. Портал об’єктно-орієнтованого програмування: http://oop.in.ua/tag/FAQ/
5. Книжки з програмування: як читати і що саме: https://dou.ua/lenta/articles/programming-books/
6. Українська технічна література. Програмування: https://ukrtechlibrary.wordpress.com/tag/програмування/
7. Задачі програмування із прикладами розв’язку: http://purecodecpp.com/uk/archives/433
8. Задачі програмування із прикладами розв’язку: http://library.nuft.edu.ua/ebook/datathree.php?ID=138
9. Задачі програмування із прикладами розв’язку: http://abramov.org.ua/blog/category/opp/obchislennya-%D1%96z-zber%D1%96gannyam-posl%D1%96dovnosti/
10. Вступ до Python 3: https://www.learnpython.org/
11. Відеуроки для вивчення ООП у Python 3: https://www.youtube.com/watch?v=ZDa-Z5JzLYM https://www.youtube.com/watch?v=BJ-VvGyQxho https://www.youtube.com/watch?v=rq8cL2XMM5M

https://www.youtube.com/watch?v=RSl87lqOXDE

https://www.youtube.com/watch?v=3ohzBxoFHAY

https://www.youtube.com/watch?v=jCzT9XFZ5bw

12. Dusty Phillips. Python 3 Object-oriented Programming. Second edition / Dusty Phillips. – Packt Publishing Ltd, 2015. – 431 pp.