МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Волинський національний університет імені Лесі Українки
Кафедра комп’ютерних наук та кібербезпеки

Глинчук Л. Я.

ТЕСТИ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ З ПРОГРАМУВАННЯ МОВОЮ PYTHON

для студентів спеціальностей F3 Комп’ютерні науки та F5 Кібербезпека та захист інформації першого (бакалаврського) рівня

Луцьк 2025

УДК 004.43(079.1)

Рекомендовано до видання науково-методичною радою Волинського національного університету імені Лесі Українки (протокол № 3 від 19 листопада 2025 р.)

Рецензенти:

Пастернак В.В. – кандидат технічних наук, доцент кафедри загальної математики та методики навчання інформатики Волинського національного університету імені Лесі Українки;

Багнюк Н.В. – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та кібербезпеки Луцького національного технічного університету.

Г 54 Тести для підсумкового контролю з програмування мовою Python. [Електронний ресурс] / укладач Л. Я. Глинчук; ВНУ ім. Лесі Українки. Електронні текстові данні (1 файл: 125 КБ). Луцьк : ВНУ ім. Лесі Українки, 2025. 70 с.

Тести призначені для проведення одного з видів модульного контролю для здобувачів освіти спеціальностей F3 Комп'ютерні науки та F5 Кібербезпека та захист інформації першого (бакалаврського) рівня. Розробка містить тестові питання до трьох змістових модулів: Вступ до програмування мовою Python, Об'єктно-орієнтоване програмування мовою Python, Подійно-орієнтоване програмування у Python. Розроблені питання мають вигляд не тільки тестових запитань, є також і питання у яких відповідь потрібно вписати самому, або проставити відповідність. Після запитань є розділ «Правильні відповіді», де вписані правильні відповіді до усіх запитань. Питання підготовлені відповідно до тем силабусів ОК «Програмування» для обох спеціальностей та відповідають теорії та практичним прикладам, що розглядається на відповідних лекційних заняттях. Методичка буде корисна усім, хто бажає перевірити рівень своїх знань з мови програмування Python або використати в навчальних цілях для перевірки знань здобувачів освіти.

Методична розробка підготовлена з урахуванням досвіду викладання дисциплін з програмування на факультеті інформаційних технологій і математики Волинського національного університету імені Лесі Українки.

ЗМІСТ

Змістовий модуль «Вступ до програмування мовою Python»

Тема 1. Вступ до мови програмування Python. Основи мови Python (синтаксис, змінні, типи даних). Структура програми

1.1 Хто є автором мови програмування Python?

Варіанти відповідей:
- Гвідо ван Россум
- Ніклаус Вірт
- Б'ярне Страуструп
- Юкіхіро Мацумото

1.2 Яка мова програмування стала основою для створення Python?

Варіанти відповідей:
- Fortran
- Lisp
- C++
- ABC

1.3 Яка особливість синтаксису вкладених блоків інструкцій у Python порівняно з іншими мовами програмування?

Варіанти відповідей:
- вкладені інструкції відокремлені спеціальними операторами
- вкладені інструкції повинні мати однакові розміри
- вкладені інструкції об'єднуються в блоки за величиною відступів
- вкладені інструкції повинні мати не більше 4 рядків коду

1.4 Для введення, виведення даних у мові програмування Python використовуються функції: …

Варіанти відповідей:
- print(), output()
- print(), input()
- read(), write()
- print(), readln()

1.5 Змінна в мові Python – ...

Варіанти відповідей:
- містить пояснювальні тексти і полегшує читання і розуміння програм
- постійне значення певного типу даних, записане у вихідному коді комп'ютерної програми

- це посилання на область пам'яті комп'ютера, в якій зберігається деякі дані (посилання на деякий об’єкт)
- незмінюваний елемент в конструкції циклу

1.6 Які парадигми програмування використовуються в мові Python?

Варіанти відповідей:
- функціональна, структурна, логічна
- об’єктно-орієнтовна та структурна
- об’єктно-орієнтовна, імперативна, функціональна та структурна
- імперативна, функціональна, структурна

1.7 Які існують способи запуску програм, що написані мовою Python?

Варіанти відповідей:
- режим інтерпретатора та режим компілятора
- інтерактивний режим та пакетний режим
- режим ООП та режим структури
- інтерактивний режим та режим середовища програмування

1.8 Неявна типізація в Python означає, що ...

Варіанти відповідей:
- тип змінної визначається безпосередньо при виконанні програми
- при оголошенні змінної її тип не вказується
- не має можливості проводити операції у виразах з даними різних несумісних типів
- змінна неявно використовується при обчисленнях виразів

1.9 Як позначається однорядковий коментар в мові Python?

Варіанти відповідей:
- за допомогою символу "*"
- за допомогою символу "//"
- за допомогою символу "#"
- за допомогою символу "%"

1.10 Що є характеристикою кросплатформенної мови програмування?

Варіанти відповідей:
- єдина платформа для роботи
- портабельність на різних операційних системах без змін коду
- залежність від системно-залежних функцій
- можливість роботи тільки на Unix-сумісних операційних системах

Тема 2. Числові дані. Винятки та їх обробка. Реалізація базових алгоритмічних структур мовою Python

2.1 Що представляє собою виняток (exception) у Python?

Варіанти відповідей:
- аварійний стан, який відбувається в кодовій послідовності під час виконання програми
- тип даних, що містить інформацію про помилки, які можуть виникнути під час виконання програми
- незнайомі функції або методи, які не були оголошені у програмі
- помилки, які стають причиною аварійного відключення комп'ютера

2.2 Які блоки можуть бути присутніми в конструкції try-except для обробки винятків?

Варіанти відповідей:
- try, raise, except, finally
- try, except, as, else
- try, else, finally, as, raise
- try, except, else, finally

2.3 Як можна обробити кілька різних винятків, які можуть виникнути в програмі?

Варіанти відповідей:
- використовуючи інший, множинний блок try set except
- використовуючи кілька блоків except в конструкції try-except
- використовуючи кілька блоків except, які розташовані в різних частинах коду
- використовуючи кілька блоків except, розташованих після блоку finally

2.4 Яким чином можна зв'язати виняток зі змінною для подальшого виведення деталей про помилку?

Варіанти відповідей:
- за допомогою оператора catch
- за допомогою оператора except as
- за допомогою оператора raise
- за допомогою оператора bind

2.5 Як буде виглядати на мові Python даний вираз?

$$e^{x+y} + \frac{5}{\cos(y-x)+3}$$

Впишіть правильну відповідь самостійно.

2.6 Оператор continue призначений для:

Варіанти відповідей:
- переривання поточної ітерації циклу і повернення на початок
- переривання поточної ітерації циклу і переходу до наступної
- переривання ітерації циклу і завершення роботи програми
- правильної відповіді немає

2.7 Оператор break призначений для:

Варіанти відповідей:
- припинення роботи усіх вкладених циклів
- дострокового припинення роботи циклу (for або while)
- припинення роботи циклу тоді, коли послідовність елементів закінчилася
- того, щоб не допустити дострокове припинення роботи циклу

2.8 Який буде результат функцій int('11',2), float('1e-003'), bin(5)?

Впишіть правильні відповіді.

2.9 Який результат виведе наступна програма?

```
a = 7
b = 3
c = 10

if a > b:
    if c % a == 3:
        print("1")
    elif a + b > c or c - b < a:
        print("2")
    else:
        print("3")
else:
    if b * 2 == c:
        print("4")
    else:
        print("5")
```

Впишіть правильну відповідь самостійно.

2.10 Що буде виведено на екран після виконання наступного коду?

```
count = 0
for i in range(3, 8):
```

```
    if i % 2 == 0:
        continue
    for j in range(1, i):
        count += j
print(count)
```

Впишіть правильну відповідь самостійно.

Тема 3. Структури даних в Python: списки, кортежі, словники, рядкові величини, множини

3.1 Що буде результатом операції list(range(5)) в мові Python?

Впишіть правильну відповідь самостійно.

3.2 Як отримати кожен другий елемент списку my_list від початку до кінця?

Варіанти відповідей:
- my_list[0::2]
- my_list[2::]
- my_list[::2]
- my_list[::-2]

3.3 Які властивості ключів в словнику? Чому ключі повинні бути унікальними?

Варіанти відповідей:
- ключі повинні бути числовими значеннями та унікальними для забезпечення швидкого доступу до елементів
- ключі в словнику можуть бути будь-якого типу, але повинні бути унікальними, оскільки вони ідентифікують кожен елемент
- ключі повинні бути унікальними рядками для забезпечення доступу до відповідних значень

3.4 Що робить метод dict.get(key[, default])?

Варіанти відповідей:
- повертає значення за ключем, якщо ключ існує, в іншому випадку повертає значення за замовчуванням
- за ключем повертає номер елемента у словнику
- додає новий ключ у словник
- видаляє елемент зі словника за вказаним ключем

3.5 Як правильно створити двовимірний список розмірністю 3×3, заповнений нулями?

Варіанти відповідей:

- matrix = [[0]*3]*3
- matrix = [[0 for j in range(3)] for i in range(3)]
- matrix = [0 for i in range(3)][0 for j in range(3)]
- matrix = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

3.6 Що виведе наступний код?

```
t = (10, 20, 30, 40, 50)
print(t[1:4])
```

Варіанти відповідей:

- (10, 20, 30)
- (20, 30, 40)
- [20, 30, 40]
- (30, 40, 50)

3.7 Що буде виведено в результаті виконання цього коду?

```
s = "Python Programming"
print(s[::-1].upper()[3:10])
```

Варіанти відповідей:

- GNIMMOR
- GNIMMORP
- GNIHTYP
- GNIMMOR

3.8 Що виведе наступний код?

```
text = "Python is fun"
print(text.replace(" ", "_").upper())
```

Варіанти відповідей:

- Python_is_fun
- PYTHON IS FUN
- PYTHON_IS_FUN
- python_is_fun

3.9 Що виведе наступний код?

```
t = (1, 2, 3)
lst = [4, 5, 6]
combined = list(t) + lst
combined[0] = 10
print(t)
print(combined)
```

Варіанти відповідей:

- (1, 2, 3)
  [10, 2, 3, 4, 5, 6]
- (10, 2, 3)
  [10, 2, 3, 4, 5, 6]
- (1, 2, 3)
  [10, 4, 5, 6]
- (1, 2, 3)
  [10, 2, 3, 4, 5, 6, 6]

3.10 Яка операція не може бути виконана над окремим елементом кортежу в мові Python?

Варіанти відповідей:

- отримання значення за індексом – t[1]
- перебирання елементів у циклі – for x in t:
- заміна елемента – t[0] = 10
- перевірка наявності елемента – 5 in t

3.11 Який результат виконання коду?

```
s = {1, 2, 3}
s.add(4)
s.update([5, 6])
print(s)
```

Варіанти відповідей:

- {1, 2, 3, 4, 5, 6}
- {1, 2, 3, [4, 5, 6]}
- {1, 2, 3, (4, 5, 6)}
- {4, 5, 6}

3.12 Що буде виведено після виконання цього коду?

```
student = {"name": "Oleh", "age": 20, "grade": "A"}
student["age"] += 1
student["city"] = "Kyiv"
print(len(student))
```

Впишіть правильну відповідь самостійно.

3.13 Що буде результатом виконання коду?

```
students = {"IT": {"Anna", "Bob"}, "CS": {"Bob", "Eva"}}
print(students["IT"] | students["CS"])
```

Варіанти відповідей:

- {'Anna', 'Bob', 'Eva'}

- {'IT', 'CS'}
- {'Bob'}
- TypeError

3.14 Що виведе код?

```
text = "Python"
d = {text[0]: text[-1]}
print(d)
```

Варіанти відповідей:
- Error
- {'P': 'n'}
- {'Python': 'Python'}
- {'n': 'P'}

3.15 Якщо потрібно створити список зі значень словника data = {"a": 1, "b": 2}, можна використати вираз list(data._____()).

Варіанти відповідей:
- keys
- values
- items
- get

Тема 4. Функції, лямбда функції та рекурсія

4.1 Оберіть правильний формат опису функції у Python:

Варіанти відповідей:
- def (список формальних параметрів): ім’я_функції оператори тіла функції
- ім’я_функції = def (оператори тіла функції, список формальних параметрів)
- def ім’я_функції (список формальних параметрів): оператори тіла функції
- function ім’я_функції (список формальних параметрів) = оператори тіла функції

4.2 Якщо в тілі функції відсутня інструкція return, то що поверне функція?

Варіанти відповідей:
- 0
- порожній рядок ""
- значення None
- повідомлення про помилку

4.3 Для того щоб вказати, що параметр функції матиме значення за замовчуванням, необхідно...

Варіанти відповідей:
- записати після параметра символ : і вказати значення
- записати після параметра символ = і вказати значення
- перед параметром додати ключове слово default
- оголосити параметр після *args

4.4 Як правильно викликати функцію з іменованими параметрами?

```
def f(a, b):
    return a + b
```

Варіанти відповідей:
- f(5, 3)
- f(a=5, b=3)
- f(5=b, 3=a)
- f(a:=5, b:=3)

4.5 Де параметр за замовчуванням, а де іменований у даному прикладі?

```
def f(a=5, b=10):
    print(a, b)

f(b=20)
```

Варіанти відповідей:
- a – параметр за замовчуванням, b – параметр за замовчуванням, обидва не іменовані
- a – параметр за замовчуванням, b – іменований параметр
- a – іменований параметр, b – параметр за замовчуванням
- обидва параметри іменовані

4.6 Щоб функція могла приймати змінну кількість іменованих аргументів, при її описі необхідно вказати параметр, перед яким ставиться символ …

Варіанти відповідей:
- *
- **
- @
- #

4.7 Яка відмінність між локальними та глобальними змінними у Python?

Варіанти відповідей:

- локальні змінні створюються всередині функції і доступні лише в ній, глобальні змінні створюються поза функцією і доступні у всіх функціях
- локальні змінні доступні у всіх функціях, глобальні лише в одній функції
- глобальні змінні не можна змінювати всередині функції
- локальні змінні можна оголошувати поза функцією

4.8 Що таке lambda-функція у Python та який правильний синтаксис, до прикладу, для обчислення суми двох чисел?

Варіанти відповідей:
- lambda-функція – це функція з обмеженим доступом до змінних; синтаксис: def lambda(x, y): return x + y
- lambda-функція – це анонімна функція, яку можна визначити без імені; синтаксис: lambda x, y: x + y
- lambda-функція – це анонімна функція, яка завжди повертає none; синтаксис: lambda (x, y) => x + y
- lambda-функція – це функція, яка може мати лише глобальні змінні; синтаксис: def f = lambda(x, y): x + y

4.9 Яке значення буде виведене після виконання коду?

```
def f(a, b=2):
    return a * b

print(f(3))
```

Впиши правильну відповідь самостійно.

4.10 Яке значення буде виведене після виконання коду?

```
def f(a, b=2, *args, **kwargs):
    total = a + b
    for num in args:
        total += num
    for key in kwargs:
        total += kwargs[key]
    return total

print(f(1, 2, 3, 4, x=5, y=6))
```

Впишіть правильну відповідь самостійно.

Тема 5. Робота з файлами в Python

5.1 Яка функція використовується для читання з файлу в Python?

Варіанти відповідей:
- read_file()
- read()
- file.read()
- open.read()

5.2 Що з перерахованого в Python є фізичним файлом, а що – логічним файлом?

Варіанти відповідей:
- фізичний файл – це об’єкт у пам’яті програми, логічний файл – це файл на диску
- фізичний файл – це тимчасовий файл, логічний файл – це постійний файл на диску
- фізичний файл і логічний файл – це синоніми
- фізичний файл – це файл на диску, логічний файл – це об’єкт у пам’яті програми, через який програма працює з даними

5.3 Якою командою можна отримати всі рядки файлу у вигляді списку?

Варіанти відповідей:
- f.read()
- list(f)
- f.readlines()
- readlines(list(f))

5.4 Які переваги використання менеджера контексту with для роботи з файлами?

Варіанти відповідей:
- забезпечує виконання завершальних дій, таких як автоматичне закриття файлу
- робить файли доступними для читання тільки в рамках визначеного контексту
- виконує операції читання та запису з файлами
- автоматично відкриває файл у режимі дозапису

5.5 Які значення може приймати параметр mode у функції open()?

Варіанти відповідей:
- 'read', 'write', 'execute'
- 'on', 'off', 'standby'
- 'r', 'w', 'x'
- 'r', 'w', 'x', 'g'

5.6 Що буде виведено після виконання цього коду?

```
with open("test.txt", "w") as f:
```

```
    f.write("012345")

with open("test.txt", "r") as f:
    data = f.read(3)
    print(len(data))
```

Впишіть правильну відповідь самостійно.

5.7 Що буде надруковано в результаті виконання коду?

```
with open("file2.txt", "w") as f:
    f.write("123\n456\n789")

with open("file2.txt", "r") as f:
    lines = f.readlines()
    print(lines)
```

Варіанти відповідей:

- ['123\n', '456\n', '789']
- 123, 456, 789
- ['123', '456', '789']
- [123, 456, 789]

5.8 Що буде виведено в результаті виконання коду?

```
f = open("file3.txt", "w+")
f.write("abcdef")
f.seek(2)
print(f.read(2))
f.close()
```

Впишіть правильну відповідь самостійно.

5.9 Що буде надруковано в результаті виконання коду?

```
with open("file5.txt", "w") as f:
    f.write("Python")

with open("file5.txt", "r+") as f:
    f.seek(3)
    f.write("XYZ")
    f.seek(2)
    print(f.read())
```

Впишіть правильну відповідь самостійно.

5.10 Зіставте режим відкриття файлу з його призначенням у Python.

Режими відкриття:

1. r
2. w
3. a
4. r+
5. x

Призначення:
a) відкриття для читання та запису, файл повинен існувати
b) створення нового файлу, якщо файл уже існує – помилка
c) відкриття для читання, файл повинен існувати
d) відкриття для запису, якщо файл існує – його вміст буде видалено
e) відкриття для додавання, нові дані додаються в кінець файлу

Тема 6. Модулі в Python. Створення unit-тестів для мови Python

6.1 Модулем в Python називають...

Варіанти відповідей:
- особливий файл з програмою, який має розширення .mpy
- текстовий документ з поясненнями до коду програми
- будь який файл лише із спеціально структурованою програмою, розширення не має значення
- окремий файл програми, з розширенням .py, який можна підключати до інших програм, щоб повторно використовувати код

6.2 Інструкція import у роботі з модулями використовується для того щоб…

Варіанти відповідей:
- отримати доступ до папки з файлами
- отримати доступ до окремого компонента модуля
- імпортувати усі документи з вказаної папки
- отримати доступ до всього модуля, який підключаємо

6.3 Інструкція from має спеціальну форму from*. Ця форма дозволяє...

Варіанти відповідей:
- підключати весь модуль
- підключати всі імена заданого модуля
- підключати окремий компонент модуля
- створити/видалити весь модуль

6.4 Для чого використовують модулі у Python?

Варіанти відповідей:

- для зберігання зображень і відео
- для розділення програми на частини та повторного використання коду
- для автоматичного видалення непотрібних файлів
- для створення графічного інтерфейсу користувача

6.5 Встановіть відповідність між типом модуля та його характеристикою.
Тип модуля:
1. Вбудований модуль
2. Користувацький модуль
3. Зовнішній (сторонній) модуль

Характеристика:
а) створюється користувачем у вигляді файлу .py
б) поставляється разом із інтерпретатором Python (наприклад, math, sys)
в) потрібно встановити окремо через менеджер пакетів pip (наприклад, numpy, requests)

6.6 Що таке unit-тестування?

Варіанти відповідей:
- тестування окремих компонентів програми на їх коректність
- тестування взаємодії між компонентами програми (модуль-програма)
- тестування всього програмного коду
- тестування програмного коду на синтаксичні помилки

6.7 Які кроки включає процес написання unit-тестів?

Варіанти відповідей:
- вибір імені тесту, перевірка очікуваних результатів, розробка робочого процесу
- створення коду, запуск тестів, аналіз результатів, перевірка звітів
- визначення тестованих «юнітів», розробка тестових випадків, написання тестового коду, запуск тестів, аналіз результатів і виправлення помилок
- визначення тестованих «юнітів», написання тестового коду, запуск тестів, аналіз результатів, зберігання коду в інший файл

6.8 Установіть відповідність між елементом unit-тестування та його описом.
Елемент unit-тестування:
1. Unittest
2. Testcase
3. Assertequal()
4. Test_ (префікс у назві методу)

Опис елемента:
а) початок назви методу, який вказує, що це тест

б) клас, у якому описуються окремі тести
в) модуль стандартної бібліотеки, що використовується для створення тестів
г) метод, який перевіряє рівність очікуваного та фактичного результатів

6.9 Розгляньте код:

```
import unittest

def divide(a, b):
    if b == 0:
        raise ValueError("Division by zero!")
    return a / b

class TestMath(unittest.TestCase):
    def test_divide_values(self):
        self.assertAlmostEqual(divide(5, 2), 2.5)
        self.assertRaises(ValueError, divide, 5, 0)

if __name__ == '__main__':
    unittest.main()
```

Що перевіряє цей тест?

Варіанти відповідей:

- чи функція divide() правильно ділить числа і завжди повертає ціле число
- чи функція divide() правильно ділить числа і обробляє помилку при діленні на нуль
- чи функція divide() правильно ділить і множить два числа
- чи функція divide() повертає рядок

6.10 Розгляньте код:

```
import unittest

def divide(a, b):
    if b == 0:
        raise ValueError("Division by zero!")
    return a / b

class TestMath(unittest.TestCase):
    def test_divide_values(self):
        self.assertAlmostEqual(divide(5, 2), 2.5)
        self.assertRaises(ValueError, divide, 5, 0)

if __name__ == '__main__':
    unittest.main()
```

Що роблять методи assertAlmostEqual() та assertRaises() у цьому тесті?

Варіанти відповідей:
- assertAlmostEqual() перевіряє рівність типів значень, а assertRaises() перевіряє, що результат не дорівнює нулю
- assertAlmostEqual() перевіряє, що результат ділення близький до очікуваного, а assertRaises() перевіряє, що викликається зазначене виключення
- assertAlmostEqual() перевіряє, що результат ділення дорівнює нулю, а assertRaises() перевіряє, що викликається будь-яка помилка
- assertAlmostEqual() перевіряє наявність виключення, а assertRaises() перевіряє правильність обчислення

Змістовий модуль «Об'єктно-орієнтоване програмування мовою Python»

Тема 1. Порівняння процедурного програмування та ООП. Оголошення класу, реалізація класу мовою Python. Конструктор, деструктор. Реалізація інкапсуляції мовою Python

1.1 Встановіть відповідність між характеристикою програмування та підходом, якому вона властива.

Характеристика:
1. Програма складається з набору функцій, які викликаються у певній послідовності
2. Основною одиницею є клас та об'єкт
3. Для повторного використання коду використовують наслідування
4. Дані й функції зберігаються окремо
5. Основна увага приділяється послідовності дій (алгоритму)

Підхід:
a – об'єктно-орієнтоване програмування
b – процедурне програмування

1.2 Встановіть відповідність між поняттям і його визначенням у контексті об'єктно-орієнтованого програмування мовою Python.

Поняття:
1. Клас –
2. Об'єкт –
3. Метод –
4. Поле (атрибут) –

Варіанти визначень:
a – окремий екземпляр класу, який має власні значення атрибутів
b – змінна, що зберігає стан або дані, пов'язані з конкретним об'єктом

c – опис структури та поведінки об’єктів, своєрідний шаблон для їх створення
d – функція, що належить до класу і визначає його поведінку.

1.3 Які рівні доступу до даних надає мова Python (3)?

Варіанти відповідей:
- публічний, приватний, локальний
- глобальний, локальний, приватний
- публічний, захищений, приватний
- публічний, захищений, індивідуальний

1.4 Конструктор – це ... (в ООП у Python)

Варіанти відповідей:
- функція, що перетворює список у кортеж
- спеціальний метод, який автоматично викликається під час створення об’єкта класу
- метод, який створює копію вже існуючого об’єкта
- інструмент для побудови графічного інтерфейсу в Python

1.5 Що таке інкапсуляція в об’єктно-орієнтованому програмуванні?

Варіанти відповідей:
- спосіб створення нових класів на основі вже існуючих
- механізм, який дозволяє виконувати кілька дій одночасно
- метод, що автоматично створює нові об’єкти під час виконання програми
- процес приховування внутрішніх даних об’єкта та обмеження доступу до них ззовні

1.6 Розгляньте наведений код і визначте, який принцип об’єктно-орієнтованого програмування тут реалізовано:

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance   # прихований атрибут

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance

account = BankAccount(1000)
account.deposit(500)
print(account.get_balance())
```

Варіанти відповідей:
- спадкування
- поліморфізм
- інкапсуляція
- абстракція

1.7 Як можна забезпечити доступ до прихованого (приватного) атрибута та його значення ззовні класу у Python?

Варіанти відповідей:
- безпосередньо звернутися до атрибута через його ім’я, наприклад obj.__value
- використати гетери та сетери, які повертають або змінюють значення атрибута або через механізм коригування імен
- використати оператор global для оголошення атрибута загальнодоступним
- змінити ім’я атрибута у класі, прибравши подвійне підкреслення

1.8 Що таке механізм коригування імен (name mangling) при інкапсуляції у Python?

Варіанти відповідей:
- це автоматичне перетворення імені приватного атрибута у формат _ІмʼяКласу__ІмʼяАтрибута, щоб уникнути випадкового доступу ззовні
- це процес перейменування всіх методів класу для зручності виклику
- це механізм, який дозволяє автоматично створювати публічні копії приватних атрибутів
- це метод, що робить усі атрибути класу незмінними

1.9 Що станеться під час виконання наступного коду?

```
class BankAccount:
    def __init__(self, owner, balance):
        self.owner = owner
        self.__balance = balance

account = BankAccount("Іван", 1000)
print(account.__balance)
```

Варіанти відповідей:
- на екран буде виведено: __balance
- на екран буде виведено: 1000
- виникне помилка доступу до приватного атрибута
- програма завершиться без виведення результату

1.10 Яка з наведених характеристик правильно описує особливість деструктора в об’єктно-орієнтованому програмуванні на Python?

Варіанти відповідей:
- деструктор викликається вручну для завершення роботи об’єкта
- деструктор викликається автоматично, коли об’єкт знищується
- деструктор створює копію об’єкта перед його видаленням
- у Python деструктори не підтримуються

Тема 2. Сетери, гетери, делетери. Статичні поля та методи. Графічна бібліотека turtle та її застосування. Реалізація класу на базі уже створеного класу: композиція

2.1 Встановіть відповідність між описом та методом доступу до атрибутів класу в Python

Опис:
1. Метод, який використовується для отримання значення приватного атрибута
2. Метод, який використовується для встановлення або зміни значення атрибута
3. Метод, який використовується для видалення атрибута
4. Функція (властивість), яка об’єднує гетер, сетер і делетер в одному інтерфейсі

Метод:
a. deleter
b. setter
c. getter
d. property

2.2 Що таке статичне поле (класове поле) у Python?

Варіанти відповідей:
- атрибут, який належить конкретному об’єкту класу і може мати різні значення для кожного екземпляра
- атрибут, який належить класу і спільний для всіх екземплярів цього класу
- метод, який викликається без створення об’єкта класу
- атрибут, який видаляється після завершення роботи методу класу

2.3 Яка особливість статичного методу у Python?

Варіанти відповідей:
- може змінювати значення атрибутів конкретного об’єкта класу і отримує доступ до всіх даних екземпляра через параметр self

- викликається лише через екземпляр класу, обов’язково приймає параметр self і залежить від стану конкретного об’єкта
- визначається всередині класу, але не приймає параметр self; він не залежить від конкретного об’єкта і за потреби може існувати як звичайна функції поза класом
- автоматично видаляє локальні атрибути після завершення роботи методу, щоб звільнити пам’ять

2.4 У якому випадку доцільно використовувати статичний метод у Python?

Варіанти відповідей:
- коли метод повинен змінювати стан конкретного об’єкта через його атрибути
- коли метод працює лише з класовими або приватними атрибутами екземпляра і потребує параметр self
- коли метод виконує функціональність, яка не залежить від конкретного об’єкта класу, але логічно належить класу
- коли метод автоматично видаляє атрибути класу після завершення виконання

2.5 Розгляньте код нижче. Який метод доцільно зробити статичним і як це правильно зробити?

```
class Employee:
    company_bonus = 0.1  # 10% бонус

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def calculate_bonus(self):
        return self.salary * Employee.company_bonus

    def is_valid_salary(amount):
        if amount < 0:
            raise ValueError("Salary cannot be negative")
        return True
```

Варіанти відповідей:
- calculate_bonus, бо він працює зі специфічним об’єктом і атрибутами екземпляра; його можна зробити статичним, додавши декоратор @staticmethod над методом
- is_valid_salary, бо метод не використовує self і може перевіряти значення зарплати незалежно від конкретного об’єкта; його можна зробити статичним, додавши декоратор @staticmethod над методом

- обидва методи повинні бути статичними, бо вони відносяться до зарплати
- жоден метод не можна зробити статичним, бо клас містить атрибут компанії

2.6 Яка команда в turtle використовується для руху черепашки вперед на певну відстань?

Варіанти відповідей:
- turtle.turn(100)
- turtle.forward(100)
- turtle.move(100)
- turtle.go(100)

2.7 Який із наведених прикладів коду правильно малює квадрат за допомогою turtle?

Варіанти відповідей:
A:
```
import turtle
for _ in range(4):
    turtle.forward(100)
    turtle.right(90)
turtle.done()
```

B:
```
import turtle
turtle.create_window()
turtle.move(100)
turtle.turn(90)
```

C:
```
import turtle
turtle.draw_square(100)
turtle.close()
```

D:
```
import turtle
turtle.forward(100)
turtle.rotate(90)
```

2.8 Доповніть програму, щоб черепашка намалювала рівносторонній трикутник зі стороною 150 пікселів.

```
import turtle
t = turtle.Turtle() # Створення черепашки
```

```
# Малювання рівностороннього трикутника
for _ in range(3):
    …
    t.left(120)     # поворот на 120 градусів
turtle.done() # Завершення роботи
```

Впишіть правильну відповідь самостійно.

2.9 Що таке композиція у об’єктно-орієнтованому програмуванні?

Варіанти відповідей:

- відношення між класами, коли один клас наслідує властивості іншого класу
- коли до складу створюваного класу входять екземпляри інших класів
- процес створення статичних методів та полів у класі
- використання функцій зовні класу без створення об’єктів

2.10 Розгляньте код нижче. Який приклад демонструє використання композиції?

```
class Engine:
    def start(self):
        print("Engine started")

class Car:
    def __init__(self):
        self.engine = Engine()

    def start(self):
        self.engine.start()
        print("Car started")
```

Варіанти відповідей:

- клас Car наслідує методи від класу Engine, що є прикладом композиції
- клас Car містить об’єкт класу Engine як частину своєї структури, що є прикладом композиції
- клас Engine оголошений як статичний метод у класі Car
- клас Car створює функцію зовні класу Engine

Тема 3. Реалізація класу на базі уже створеного класу: наслідування, множинне наслідування, багаторівневе наслідування. Порядок виклику методів (MRO) в Python

3.1 Що таке наслідування в ООП та який синтаксис у Python використовується для цього?

Варіанти відповідей:
- механізм створення копій об’єктів, ключове слово import
- механізм, що дозволяє одному класу використовувати атрибути й методи іншого класу, class Child(Parent):
- спосіб приховування даних від користувача, class Child(Parent):
- метод перевантаження операторів, ключові слова super, inherits

3.2 Які переваги надає механізм наслідування? (вірно декілька відповідей)

Варіанти відповідей:
- повторне використання коду
- підвищення зв’язності між класами
- можливість перевизначати методи
- зменшення кількості об’єктів
- спрощення розширюваності програм
- підвищення гнучкості архітектури
- зменшення продуктивності програм

3.3 Яке твердження правильне для функції super()?

Варіанти відповідей:
- викликає метод із батьківського класу
- створює новий об’єкт
- викликає метод із дочірнього класу
- змінює атрибути поточного класу

3.4 Який тип наслідування представлений у прикладі:

```
class A: pass
class B(A): pass
class C(B): pass
```

Варіанти відповідей:
- множинне
- багаторівневе
- ієрархічне
- комбіноване

3.5 Що буде виведено в результаті виконання коду?

```
class Employee:
    def __init__(self, name):
        self.name = name
```

```
    def info(self):
        return f"Employee: {self.name}"

class Manager(Employee):
    def __init__(self, name, team_size):
        super().__init__(name)
        self.team_size = team_size

    def info(self):
        base_info = super().info()
        return f"{base_info}, manages {self.team_size} people"

class Director(Manager):
    def __init__(self, name, team_size, department):
        super().__init__(name, team_size)
        self.department = department

    def info(self):
        return super().info() + f", department: {self.department}"

d = Director("Anna", 10, "IT")
print(d.info())
```

Варіанти відповідей:

- employee: anna
- employee: anna, manages 10 people
- employee: anna, manages 10 people, department: it
- director: anna, manages 10 people, department: it
- виникне помилка через подвійне наслідування

3.6 У Python наслідування відображає відносини типу "is-a" (є) – тобто дочірній клас є спеціалізованим випадком батьківського. Виберіть приклади, у яких між класами існують такі відносини (оберіть 4 правильні відповіді):

Варіанти відповідей:

- dog – підклас класу animal
- car – підклас класу vehicle
- student – підклас класу person
- engine – підклас класу car
- rectangle – підклас класу shape
- database – підклас класу query
- keyboard – підклас класу computer
- person – підклас класу address

3.7 Який порядок виклику методів у разі множинного спадкування визначає Python?

Варіанти відповідей:
- алфавітний порядок імен класів
- порядок оголошення методів
- MRO (Method Resolution Order)
- випадковий порядок

3.8 Що буде виведено після виконання коду?

```
class A:
    def say(self):
        print("A")

class B:
    def say(self):
        print("B")

class C(A, B):
    pass

obj = C()
obj.say()
```

Варіанти відповідей:
- A
- B
- A B
- виникне помилка через однакові методи

3.9 Що виведе код?

```
class A:
    def show(self):
        print("A")

class B(A):
    def show(self):
        print("B")
        super().show()

class C(A):
    def show(self):
        print("C")
```

```
        super().show()

class D(B, C):
    def show(self):
        print("D")
        super().show()

d = D()
d.show()
```

Варіанти відповідей:

A	B	C	D
D B C A	D C B A	A B C D	Помилка через конфлікт super()

3.10 Яка основна проблема може виникати при використанні множинного наслідування в Python і як вона вирішується?

Варіанти відповідей:

- виникає помилка компіляції, яку можна усунути за допомогою try...except
- методи батьківських класів не викликаються, тому потрібно вручну імпортувати їх
- наслідування стає неможливим, якщо класи не оголошені у спільному модулі
- виникає конфлікт імен методів або атрибутів у батьківських класах, який вирішується за допомогою порядку пошуку методів (MRO)

Тема 4. Реалізація поліморфізму мовою Python. Віртуальний метод. Зв'язки між класами: асоціація, агрегація, композиція, спадкування, залежність, реалізація

4.1 У контексті об’єктно-орієнтованого програмування поліморфізм означає: ... (оберіть дві правильні відповіді)

Варіанти відповідей:

- можливість створення об’єктів на основі шаблонів класів
- використання одного інтерфейсу для об’єктів різних типів
- об’єкти двох або більше класів можуть реагувати по-різному на однакові команди
- приховування внутрішньої реалізації від користувача
- збереження стану об’єкта між викликами методів

4.2 Прикладами поліморфізму в Python є використання методу/ів: (оберіть правильні відповіді)

Варіанти відповідей:
- __init__() для створення об’єктів
- len() для об’єктів різних типів (рядків, списків, кортежів тощо)
- print() для виведення різних типів даних
- input() для введення даних користувачем
- append() для додавання елементів до списку

4.3 Що буде результатом виконання наведеного коду?

```
class Equation:
    def solve(self):
        return "Невідомий тип рівняння"

class Linear(Equation):
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def solve(self):
        if self.a == 0:
            return "Розв’язків немає"
        return f"x = {-self.b / self.a}"

class Quadratic(Equation):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
    def solve(self):
        D = self.b**2 - 4*self.a*self.c
        if D < 0:
            return "Дійсних розв’язків немає"
        elif D == 0:
            return f"x = {-self.b / (2*self.a)}"
        else:
            x1 = (-self.b + D**0.5) / (2*self.a)
            x2 = (-self.b - D**0.5) / (2*self.a)
            return f"x1 = {x1}, x2 = {x2}"

equations = [Linear(2, -6), Quadratic(1, -3, 2), Equation()]

for eq in equations:
    print(eq.solve())
```

Варіанти відповідей:

- x = 3.0
  x1 = 2.0, x2 = 1.0
  Невідомий тип рівняння

- x = -3.0
  x1 = 1.0, x2 = 2.0
  Розв’язків немає

- x = 3.0
  x = 1.5
  Дійсних розв’язків немає

- x1 = 3.0, x2 = 2.0
  x = -3.0
  Невідомий тип рівняння

- Помилка виконання

4.4 Як у теорії програмування (ООП) називається властивість, коли під час виконання програми метод визначається залежно від типу об’єкта?

Варіанти відповідей:

- наслідування
- абстракція
- віртуальність (динамічне зв’язування)
- інкапсуляція

4.5 Яка роль віртуальних методів у реалізації поліморфізму в Python?

Варіанти відповідей:

- вони забезпечують приховування реалізації від користувача
- вони дозволяють створювати узагальнений код, який працює як з об'єктами базового класу, так і з об'єктами будь-якого з його нащадків
- вони використовуються лише для ініціалізації об’єктів
- вони дозволяють обмежити доступ до атрибутів та потрібні лише для роботи з приватними методами

4.6 Який приклад демонструє поліморфізм у Python без використання наслідування?

Варіанти відповідей:

- class A:
  def show(self):

```
        print("A")

class B(A):
    def show(self):
        print("B")
```

- ```
  def add(a, b):
      return a + b

  print(add(3, 5))
  print(add("Hello, ", "World!"))
  ```

- ```
  class Animal:
      def sound(self):
          print("Some sound")

  a = Animal()
  a.sound()
  ```

- ```
  x = 10
  y = 20
  print(x * y)
  ```

4.7 Яке твердження правильно описує асоціацію між класами в об’єктно-орієнтованому програмуванні?

Варіанти відповідей:
- асоціація означає, що один клас успадковує властивості та методи іншого класу
- асоціація забезпечує структурний зв’язок між екземплярами різних класів, дозволяючи об’єктам посилатися один на одного
- асоціація потрібна лише для того, щоб створити список об’єктів у класі
- асоціація – це коли методи одного класу автоматично викликають методи іншого класу
- асоціація гарантує, що об’єкти класу існують лише всередині об’єктів іншого класу

4.8 Встановіть відповідність між типом зв’язку та його описом.

Тип зв’язку:
1. Композиція
2. Агрегація
3. Залежність
4. Спадкування
5. Реалізація
  ```

Опис варіантів:

a. клас містить об’єкти іншого класу як частину себе, і ці об’єкти не можуть існувати окремо
b. один клас містить об’єкти іншого класу, але ці об’єкти можуть існувати самостійно
c. клас використовує об’єкти іншого класу тимчасово, без збереження довготривалих посилань
d. клас наслідує властивості та методи іншого класу
e. клас реалізує методи інтерфейсу або абстрактного класу

4.9 Для кожного з наведених прикладів визначте, який тип зв’язку між класами використано: композиція, агрегація, залежність, спадкування або реалізація.

Варіанти коду:

1.

```
class Engine:
    def __init__(self, power):
        self.power = power

class Car:
    def __init__(self):
        self.engine = Engine(150)  # Об'єкт Engine створюється всередині Car
```

2.

```
class Student:
    def __init__(self, name):
        self.name = name

class Group:
    def __init__(self, students):
        self.students = students  # Список студентів передається ззовні

s1 = Student("Іван")
s2 = Student("Марія")
g = Group([s1, s2])
```

3.

```
class Printer:
    def print_document(self, text):
        print(text)

class Computer:
    def send_to_printer(self, printer, text):
        printer.print_document(text)  # Тимчасово використовує об’єкт Printer
```

4.
```
class Animal:
    def speak(self):
        print("Some sound")

class Dog(Animal):  # Dog наслідує Animal
    def speak(self):
        print("Woof")
```

5.
```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Rectangle(Shape):  # Реалізує абстрактний метод area()
    def __init__(self, w, h):
        self.w = w
        self.h = h
    def area(self):
        return self.w * self.h
```

4.10 Що в об’єктно-орієнтованому програмуванні називається потужністю асоціації, і які її можливі типи?

Варіанти відповідей:
- потужність асоціації – це кількість методів, які клас наслідує від іншого класу; потужність асоціації буває тільки один-до-одного
- потужність асоціації – це кількість об’єктів одного класу, які можуть бути пов’язані з об’єктами іншого класу; потужність асоціації буває трьох типів: один-до-одного, один-до-багатьох, багато-до-багатьох
- потужність асоціації – це кількість атрибутів у класі; буває тільки один-до-багатьох
- потужність асоціації – це кількість класів, які реалізують один інтерфейс; потужність асоціації буває будь-яка
- потужність асоціації – це кількість рядків коду, потрібних для створення зв’язку; буває будь-яка

Тема 5. Спеціальні поля та методи. Особливості перевантаження операторів

5.1 Що таке спеціальні поля та методи (магічні) у Python?

Варіанти відповідей:

- це звичайні змінні та функції, які користувач створює для зручності
- це поля та методи з подвійними підкресленнями на початку і в кінці імені, що визначають поведінку об’єктів у стандартних операціях (наприклад, __init__, __str__, __dict__)
- це локальні та глобальні змінні класу, що зберігають значення за замовчуванням
- це спеціальні функції, в яких є поля та методи, які автоматично видаляють об’єкти після завершення програми

5.2 Встановіть відповідність між групами спеціальних полів і методів Python та їхнім призначенням.

Групи:

1. __init__, __del__
2. __str__, __repr__
3. __add__, __sub__, __mul__
4. __dict__, __class__, __name__
5. __len__, __getitem__, __setitem__

Призначення:

a. робота з колекціями та елементами об’єкта за індексом
b. перевизначення арифметичних операцій між об’єктами
c. отримання службової інформації про об’єкт або клас
d. рядкове подання об’єкта для користувача або відлагодження
e. створення та знищення об’єкта класу

5.3 Який із наведених варіантів правильно демонструє виклик спеціального методу __len__() для об’єкта списку data? Оберіть усі правильні відповіді.

Варіанти відповідей:

- data.__len__()
- len.__call__(data)
- len(data)
- __len__(data)

5.4 Що виведе наступний код?

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __str__(self):
        return f"{self.title} – {self.author}"
```

```
    def __eq__(self, other):
        return self.title == other.title and self.author == other.author

b1 = Book("Python 101", "Іван Петренко")
b2 = Book("Python 101", "Іван Петренко")
b3 = Book("Python для початківців", "Марія Іваненко")

print(b1)
print(b1 == b2)
print(b1 == b3)
print(b1.__dict__)
```

Варіанти відповідей:

- Python 101 – Іван Петренко
  True
  False
  {'title': 'Python 101', 'author': 'Іван Петренко'}

- <__main__.Book object at 0x0001>
  True
  True
  {'title': 'Python 101', 'author': 'Іван Петренко'}

- Python 101 – Іван Петренко
  False
  False
  {'title': 'Python 101', 'author': 'Іван Петренко'}

- Book object
  True
  False
  {'Book': 'Python 101', 'author': 'Іван Петренко'}

5.5 Що виведе наступний код?

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __add__(self, other):
        return Book(f"{self.title} & {other.title}", f"{self.author} & {other.author}")

    def __getitem__(self, index):
        return self.title[index]
```

```
    def __call__(self):
        return f"{self.title} написав {self.author}"

b1 = Book("Python 101", "Іван Петренко")
b2 = Book("Python для початківців", "Марія Іваненко")

b3 = b1 + b2

print(b3())
print(b1[0])
print(b1.__dict__)
print(b1.__class__)
```

Варіанти відповідей:

- Python 101 & Python для початківців написав Іван Петренко & Марія Іваненко
  P
  {'title': 'Python 101', 'author': 'Іван Петренко'}
  <class '__main__.Book'>

- Book('Python 101 & Python для початківців', 'Іван Петренко & Марія Іваненко')
  P
  {'title': 'Python 101', 'author': 'Іван Петренко'}
  <class '__main__.Book'>

- Python 101 & Python для початківців – Іван Петренко & Марія Іваненко
  P
  {'title': 'Python 101', 'author': Марія Іваненко}
  <class 'Book'>

- Book object
  Python 101 & Python для початківців – Іван Петренко & Марія Іваненко
  P
  {'Book': 'Python 101', 'author': 'Іван Петренко'}
  <class '__main__.Book'>

5.6 Для чого в Python використовується перевантаження операторів (наприклад, __add__, __eq__, __lt__)?

Варіанти відповідей:

- щоб змінювати стандартну поведінку операторів для об’єктів власних класів, це робить код більш зрозумілим та дозволяє використовувати оператори для роботи з об’єктами так, як з простими типами даних

- щоб створювати нові типи даних без використання класів та використовувати оператори для роботи з об’єктами так, як з простими типами даних
- щоб викликати вбудовані функції Python без використання методів, бо перевантаження лише змінює, як оператори працюють з об’єктами вашого класу
- щоб об’єкти автоматично зберігалися у словнику __dict__, бо перевантаження операторів має відношення до зберігання атрибутів

5.7 Встановіть відповідність між спеціальним методом і прикладом виклику оператора, який він перевантажує.

Метод:

1. __iadd__(self, other)
2. __isub__(self, other)
3. __imul__(self, other)
4. __itruediv__(self, other)
5. __imod__(self, other)

Приклад виклику:

A. a += b
B. a -= b
C. a *= b
D. a /= b
E. a %= b

Варіанти відповідей:

- 1–A, 2–B, 3–C, 4–D, 5–E
- 1–B, 2–A, 3–D, 4–E, 5–C
- 1–C, 2–B, 3–A, 4–E, 5–D
- 1–A, 2–C, 3–B, 4–E, 5–D

5.8 Встановіть відповідність між спеціальним методом і прикладом виклику оператора порівняння, який він перевантажує.

Метод:

1. __eq__(self, other)
2. __ne__(self, other)
3. __lt__(self, other)
4. __le__(self, other)
5. __gt__(self, other)
6. __ge__(self, other)

Приклад виклику:

A. a > b
B. a < b
C. a != b

D. a <= b
E. a == b
F. a >= b

Варіанти відповідей:
- 1–E, 2–C, 3–B, 4–D, 5–A, 6–F
- 1–B, 2–A, 3– F, 4–C, 5–D, 6–E
- 1–A, 2–C, 3–B, 4–E, 5–D, 6–F
- 1–C, 2–B, 3–A, 4–D, 5–E, 6–F

5.9 Розгляньте наведений код. Використано перевантаження унарних операторів. Визначте, який результат буде виведено на екран (впишіть правильну відповідь самостійно).

```
class Number:
    def __init__(self, value):
        self.value = value

    def __neg__(self):
        return Number(self.value + 3)

    def __pos__(self):
        return Number(self.value * 2)

x = Number(4)
y = -x
z = +y
print(z.value)
```

5.10 Розгляньте наведений код. Використано перевантаження бінарних операторів. Визначте, який результат буде виведено на екран (впишіть правильну відповідь самостійно).

```
class Number:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return Number(self.value + other.value + 2)

    def __mul__(self, other):
        return Number(self.value * other.value)

x = Number(3)
y = Number(4)
```

```
z = x + y * Number(2)
print(z.value)
```

Тема 6. Абстрактні класи. Поняття про метапрограмування. Декоратори

6.1 Абстрактним називається клас…

Варіанти відповідей:
- який не містить жодного конструктора
- якщо створення його екземплярів не має сенсу і не передбачається
- який не має методів, лише поля
- який можна створювати напряму без обмежень

6.2 У чому полягає призначення та особливість абстрактного методу?

Варіанти відповідей:
- особливість – він виконує обчислення, призначення – створює об’єкт класу
- особливість – він не має реалізації, призначення – декларація методу, тобто визначення його сигнатури
- особливість – він автоматично викликається при створенні об’єкта, призначення – зберігати дані
- особливість – має реалізацію, призначення – визначати конструктор класу

6.3 Щоб створити абстрактний клас, необхідно…

Варіанти відповідей:
- оголосити його як abstract і створити конструктор
- успадкувати від будь-якого класу і перевизначити всі методи
- використати модуль abc та позначити хоча б один метод як абстрактний за допомогою @abstractmethod
- створити клас без полів, але з методами, які завжди повертають одне значення

6.4 Коли доцільніше використовувати звичайний клас, а коли – абстрактний при програмуванні на Python?

Варіанти відповідей:
- абстрактні класи варто застосовувати, коли потрібно визначити загальний інтерфейс для групи пов’язаних класів, забезпечуючи структуру, безпеку та чіткість, а звичайні класи – коли необхідна більша гнучкість у створенні об’єктів і не потрібно обмежувати реалізацію
- абстрактні класи використовують лише для зберігання змінних, а звичайні – для написання методів, оскільки абстрактні класи не можуть містити логіку чи поведінку, лише дані

- звичайні класи створюють тільки для наслідування і не можна створювати їхні об’єкти, а абстрактні класи, навпаки, застосовують для створення конкретних екземплярів без потреби у наслідуванні
- абстрактні та звичайні класи виконують однакові функції, тому між ними немає суттєвої різниці, і вибір залежить лише від стилю програмування розробника, а не від логіки проєкту

6.5 У яких із наведених прикладів використано метапрограмування у Python? (Оберіть усі правильні відповіді)

Варіанти відповідей:

- result = eval("2 * 4 + 6")
  print(result)

- code = "x = 10\nprint(x)"
  exec(code)

- class Student:
    def init(self, name):
      self.name = name

- for i in range(5):
    print(i)

- numbers = [3, 5]
  print(sum(numbers))

6.6 Що таке декоратори для функцій у Python і для чого їх використовують?

Варіанти відповідей:

- це спеціальні функції, які дозволяють змінювати або розширювати поведінку інших функцій без зміни їхнього коду, зазвичай через використання символу @
- це функції, які автоматично створюють графічний інтерфейс для будь-якої програми
- це вбудований механізм для перетворення функцій на класи та об’єкти
- це методи, які дозволяють перевантажувати оператори та змінювати внутрішні атрибути функцій під час виконання програми

6.7 Розгляньте наведений фрагмент коду. Яка з функцій у ньому є декоратором?

```
def log(func):
    def wrapper():
        print("Функція виконується...")
        func()
```

```
        print("Функція завершена.")
    return wrapper

@log
def greet():
    print("Привіт, світе!")

greet()
```

Варіанти відповідей:

- log – це декоратор, оскільки вона приймає іншу функцію як аргумент і повертає нову функцію з розширеною поведінкою
- wrapper – це декоратор, оскільки вона виводить повідомлення перед і після виконання greet()
- greet – це декоратор, бо вона викликає іншу функцію всередині себе
- жодна з функцій не є декоратором, бо в коді не використовується механізм наслідування або метакласи

6.8 Який результат буде виведено після виконання наведеного коду?

```
def fancy_decorator(func):
    def wrapper(*args):
        print("*" * 5)
        result = func(*args)
        result = result ** 2 + 3
        print("*" * 5)
        return result
    return wrapper

@fancy_decorator
def add(a, b):
    return a + b

print(add(2, 3))
```

Варіанти відповідей:

- *****
  *****
  5

- *****
  *****
  28

- *****

28
*****

- *****
8
*****

6.9 Що буде виведено на екран після виконання наведеного коду?

```
def log_decorator(func):
    def wrapper(*args, **kwargs):
        print("Запуск функції...")
        result = func(*args, **kwargs)
        print("Завершення функції.")
        return result + 1
    return wrapper

def multiply_decorator(func):
    def wrapper(*args, **kwargs):
        print("× Виконуємо множення результату...")
        result = func(*args, **kwargs)
        print("× Множення завершено.")
        return result * 2
    return wrapper

@log_decorator
@multiply_decorator
def calculate(a, b):
    return a + b

print(calculate(2, 3))
```

Варіанти відповідей:

- Запуск функції...
× Виконуємо множення результату...
× Множення завершено.
Завершення функції.
11

- × Виконуємо множення результату...
Запуск функції...
11
Завершення функції.
× Множення завершено.

- Запуск функції...
  × Виконуємо множення результату...
  × Множення завершено.
  Завершення функції.
  9

- × Виконуємо множення результату...
  × Множення завершено.
  Запуск функції...
  Завершення функції.
  11

6.10 Що таке декоратор класу в Python і яке його основне призначення?

Варіанти відповідей:
- декоратор класу – це спеціальний метод класу, який викликається автоматично після створення об’єкта для його ініціалізації
- декоратор класу – це синтаксичний елемент, який дозволяє успадкувати клас від кількох батьків одночасно без явного зазначення у дужках
- декоратор класу – це вбудована функція, яка автоматично перетворює усі методи класу на статичні
- декоратор класу – це функція, яка приймає клас як аргумент і може змінювати або повертати нову версію цього класу, розширюючи його можливості

6.11 Що буде виведено після виконання наведеного коду?

```
def class_decorator(cls):
    class NewClass(cls):
        def greet(self):
            return super().greet().upper() + "!!!"
    return NewClass

@class_decorator
class Person:
    def __init__(self, name):
        self.name = name

    def greet(self):
        return f"Привіт, я {self.name}"

p = Person("Олена")
print(p.greet())
```

Варіанти відповідей:

- Привіт, я Олена
- ПРИВІТ, Я ОЛЕНА!!!
- Привіт, я ОЛЕНА!!!
- Помилка, оскільки декоратор класу не може створювати новий клас

6.12 Що буде виведено після виконання наступного коду?

```
def adjust_args(func):
    def wrapper(self, a, b):
        print(f"Оригінальні аргументи: {a}, {b}")
        return func(self, a*2, b+1)
    return wrapper

def add_method(cls):
    cls.factor = 10
    return cls

@add_method
class Calculator:
    @adjust_args
    def compute(self, a, b):
        return (a + b) * self.factor

calc = Calculator()
print(calc.compute(2, 3))
```

Варіанти відповідей:

- Оригінальні аргументи: 2, 3
  50
- Оригінальні аргументи: 2, 3
  80
- Оригінальні аргументи: 2, 3
  70
- Помилка виконання, бо декоратор функції не враховує self

Змістовий модуль «Подійно-орієнтоване програмування у Python»

Тема 1. Створення графічних програм GUI за допомогою модуля Tkinter на мові програмування Python. Перша програма. Віджет Button (кнопка). Позиціювання: Pack, Place, Grid. Обробка подій

1.1 Які з наведених тверджень правильно описують особливості та властивості модуля Tkinter у Python? Виберіть чотири правильні відповіді.

Варіанти відповідей:

1. Tkinter є вбудованим модулем Python і не потребує додаткового встановлення.
2. Tkinter створює графічні елементи, які завжди виглядають сучасно та схоже на PyQt5.
3. Tkinter є інтерфейсом Python для бібліотеки Tk, яка спочатку була розроблена для мови Tcl.
4. Tkinter не працює на Mac OS та Linux – лише на Windows.
5. Графічні елементи Tkinter відображаються за допомогою власних елементів операційної системи, що забезпечує природний вигляд на різних платформах.
6. Tkinter легко вивчати завдяки готовим віджетам та простому синтаксису.
7. Tkinter є комерційним програмним продуктом і не може використовуватися у відкритих проектах.

1.2 Що таке віджети у бібліотеці Tkinter в Python і які з наведених прикладів до них належать?

Варіанти відповідей:

- віджети – це графічні елементи інтерфейсу, такі як кнопки, поля введення, мітки, які дозволяють взаємодіяти з користувачем. до них належать button, label, entry, text та ін.
- віджети – це модулі, які використовуються для імпорту бібліотек у python. приклад: math, os, sys
- віджети – це змінні, які зберігають дані користувача під час роботи програми. приклад: int, str, list
- віджети – це окремі файли ресурсів інтерфейсу, що створюються під час компіляції; приклади: Entry, Text та ін.

1.3 Встановіть відповідність між віджетом бібліотеки Tkinter та його призначенням.

Віджет:

1. Button
2. Label
3. Entry
4. Text
5. Checkbutton
6. Radiobutton
7. Frame
8. Menu
9. Scrollbar
10. Spinbox

Призначення віджету

A. Однорядкове текстове поле

B. Багаторядкове текстове поле
C. Елемент меню
D. Смуга прокручування
E. Список значень зі стрілками для переміщення між ними
F. Прапорець
G. Текстова мітка
H. Кнопка
I. Перемикач (радіокнопка)
J. Платформа (рамка) для групування віджетів

Варіанти відповідей:
- 1 – H, 2 – G, 3 – A, 4 – B, 5 – F, 6 – I, 7 – J, 8 – C, 9 – D, 10 – E
- 1 – A, 2 – G, 3 – H, 4 – F, 5 – B, 6 – I, 7 – J, 8 – D, 9 – E, 10 – C
- 1 – B, 2 – A, 3 – D, 4 – C, 5 – F, 6 – I, 7 – J, 8 – C, 9 – D, 10 – E
- 1 – H, 2 – G, 3 – A, 4 – B, 5 – D, 6 – I, 7 – E, 8 – C, 9 – F, 10 – J

1.4 Розгляньте уважно код. Що відбувається при натисканні кнопки мишею та клавіші f у наведеній програмі Tkinter?

```
import tkinter as tk
root = tk.Tk()
root.title("Початкове вікно")
root.geometry("400x300")

def change_t_and_c():
    root.title("Назву змінено!")
    root.geometry("200x200")
    root['bg']='green'

def change_s_and_c(event):
    root.geometry("600x400")
    root['bg']='red'

btn = tk.Button(root, text="Змінити назву та колір", command=change_t_and_c)
btn.pack(pady=30)
root.bind("f", change_s_and_c)

root.mainloop()
```

Варіанти відповідей:
- при натисканні кнопки змінюється назва вікна і фон стає зеленим; при натисканні клавіші f вікно збільшується та фон стає червоним
- при натисканні кнопки зменшується вікно, а при натисканні клавіші f відкривається нове вікно

- при натисканні кнопки змінюється шрифт у мітці, а клавіша f змінює колір тексту на синій
- при натисканні кнопки змінюється назва та розміри вікна і фон стає зеленим; при натисканні клавіші f вікно збільшується та фон стає червоним

1.5 У наведеній програмі на Python з використанням бібліотеки Tkinter містяться синтаксичні та логічні помилки. Які з наведених дій потрібно виконати, щоб програма запрацювала правильно?

```
import Tkinter as
root = tk.tk()

root.Title("Початкове вікно")
root.Geometry("400*300")

def change_t_and_c():
   root.Title("Назву змінено!")
   root.geometry(200,200)
   root.bg='green'

def change_s_and_c(event):
   root.geometry["600x400"]
   root['background']='red'

btn = tk.Button(root, text="Змінити назву та колір", command=change_t_and_c)
btn.pack(pady=30)
root.Bind("f", change_s_and_c)

root.Mainloop()
```

Варіанти відповідей:

- залишити import Tkinter as tk, але змінити root.tk() на tk.Window(), root.Title() на root.Titlecase(), root.Geometry() на root.size(), root.Bind() на root.event(), root.Mainloop() на root.start()
- замінити import Tkinter as tk на import tkinter as tk, root.tk() на tk.Tk(), root.Title() на root.title(), root.Geometry("400*300") на root.geometry("400x300"), root.Bind() на root.bind(), root.Mainloop() на root.mainloop()
- замінити import Tkinter as tk на from tkinter import *, але залишити решту викликів без змін (root.Title(), root.Geometry(), root.Bind())
- використати import tkinter, але змінити створення вікна на root = Tkinter(), залишивши root.Title() і root.Geometry("400*300") без змін

1.6 Що визначає менеджер розташування pack() у бібліотеці Tkinter?

Варіанти відповідей:

- розташовує віджети за координатами x і y, які задаються вручну
- використовується для створення сітки елементів у вигляді рядків і стовпців
- розташовує віджети відносно країв вікна (зверху, знизу, зліва, справа) та автоматично підлаштовує їхній розмір
- використовується для створення контейнера, у якому зберігаються всі елементи інтерфейсу без їх відображення

1.7 Уважно розгляньте поданий код.

```
import tkinter as tk

root = tk.Tk()
root.title("Pack Example")
root.geometry("300x200")

label_top = tk.Label(root, text="Верхній напис", bg="lightblue")
label_top.pack(side="top", fill="x")

label_left = tk.Label(root, text="Ліворуч", bg="lightgreen")
label_left.pack(side="left", fill="y")

label_right = tk.Label(root, text="Праворуч", bg="lightyellow")
label_right.pack(side="right", fill="y")

button_bottom = tk.Button(root, text="Кнопка внизу", bg="lightpink")
button_bottom.pack(side="bottom", fill="x")

root.mainloop()
```

У програмі віджети займають різні позиції у вікні, і деякі з них розтягуються вздовж горизонтальної або вертикальної осі. Який параметр визначає, чи віджет заповнює доступний простір у напрямку? Відповідь – одне слово.

1.8 У наведеній програмі віджети позиціонуються за допомогою методу place(). Для яких віджетів у програмі задано відносне розташування у відсотках від ширини та висоти вікна?

```
import tkinter as tk

root = tk.Tk()
root.title("Place Example")
root.geometry("400x300")

label1 = tk.Label(root, text="Верхній лівий кут", bg="lightblue")
```

```
label1.place(x=10, y=10)

label2 = tk.Label(root, text="По центру", bg="lightgreen")
label2.place(relx=0.5, rely=0.5, anchor="center")

label3 = tk.Label(root, text="Внизу праворуч", bg="lightpink")
label3.place(relx=1.0, rely=1.0, x=-10, y=-10, anchor="se")

root.mainloop()
```

Варіанти відповідей:

- label1
- label1 та label2
- label2 та label3
- label1, label2 та label3

1.9 Уважно розгляньте код. Що треба вписати у відповідні рядки?

```
import tkinter as tk

root = tk.Tk()
root.title("Grid Fill-in Example")
root.geometry("300x200")

label1 = tk.Label(root, text="A", bg="lightblue")
label1._____ # Що треба вставити, щоб віджет був у рядку 0, стовпець 0?

label2 = tk.Label(root, text="B", bg="lightgreen")
label2._____  # Що треба вставити, щоб віджет був у рядку 0, стовпець 1?

label3 = tk.Label(root, text="C", bg="lightpink")
label3._____  # Що треба вставити, щоб віджет був у рядку 1 та займав обидва стовпці?

root.mainloop()
```

Варіанти відповідей:

- pack(row=0, column=0), place(row=0, column=1), grid(row=1, column=0, columnspan=1)
- grid(row=0, column=0), grid(row=0, column=1), grid(row=1, column=0, columnspan=2)
- grid(row=1, column=1), grid(row=1, column=1), grid(row=1, column=0)
- grid(row=1, column=1), grid(row=0, column=1), grid(row=0, column=0, columnspan=2)

1.10 У наведеній програмі натискання на кнопку викликає виконання функції on_click. Який метод дозволяє прив’язати подію до віджета, щоб при певній дії (наприклад, натисканні кнопки миші) виконувалась функція? Відповідь – одне слово.

```
import tkinter as tk

def on_click(event):
    print("Ви натиснули кнопку миші!")

root = tk.Tk()
root.title("Обробка подій")
root.geometry("300x200")

button = tk.Button(root, text="Натисни мене")
button.pack(pady=50)
button.bind("<Button-1>", on_click)

root.mainloop()
```

Тема 2. Особливості, властивості та робота з віджетами: Label, Frame, Text, Scrollbar, Entry, Radiobutton, Checkbutton

2.1 Виберіть вірний рядок коду встановлення шрифту для мітки:

Варіанти відповідей:

- label = ttk.Label(font="Arial", 24, text=("Я мітка"))
- label = ttk.Label(text="Hello Мітка", font=("Arial", 24))
- label = ttk.Label(text="Hello Мітка", font=("Arial"))
- label = ttk.Label(text="Hello Мітка", font=(Arial, "24"))

2.2 Розгляньте фрагмент коду на Python:

```
import tkinter as tk

root = tk.Tk()
frame = tk.Frame(root, bg="lightblue", width=200, height=100)
frame.pack()
label = tk.Label(frame, text="Привіт з фрейму!")
label.pack()

root.mainloop()
```

Яке призначення віджета Frame у цьому коді?

Варіанти відповідей:

- створює нове головне вікно програми
- слугує контейнером для розміщення інших віджетів
- відображає однорядкове текстове поле з повідомленням у вікні
- додає смугу прокрутки до вікна

2.3 Віджет ... доступний лише в основному пакеті tkinter, у пакеті tkinter.ttk йому аналога немає. Оберіть всі правильні відповіді.

Варіанти відповідей:

- Frame
- Text
- Scrollbar
- Entry
- Listbox
- Menu
- MessageBox
- Canvas

2.4 Розгляньте фрагмент коду:

```
import tkinter as tk

root = tk.Tk()
text_box = tk.Text(root, height=3, width=20)
text_box.insert("1.0", "Python – мова програмування.")
text_box.pack()

root.mainloop()
```

Що відбудеться після виконання цього коду?

Варіанти відповідей:

- відобразиться вікно з однорядковим полем введення, в яке не можна вводити текст
- відобразиться багаторядкове текстове поле із попередньо вставленим на кінець текстом
- відобразиться багаторядкове текстове поле розміром 3×20 символів із попередньо вставленим на початок текстом
- вікно не відкриється через помилку вставки тексту
- текст не з’явиться, оскільки insert() треба викликати до створення віджета

2.5 Розгляньте фрагмент коду:

```
import tkinter as tk

root = tk.Tk()
scroll = tk.Scrollbar(root)
text_box = tk.Text(root, yscrollcommand=scroll.set)
scroll.pack(side="right", fill="y")
text_box.pack(side="left", fill="both", expand=True)
scroll.config(command=text_box.yview)

root.mainloop()
```

Яке твердження є правильним щодо роботи цього коду?

Варіанти відповідей:
- віджет Scrollbar не буде працювати, бо треба використовувати метод config() до створення Text
- після запуску програми з’явиться текстове поле з вертикальною прокруткою, що працює синхронно з віджетом Text
- прокрутка працює лише у горизонтальному напрямку
- код викликає помилку через неправильне розташування pack()

2.6 Розгляньте фрагмент коду:

```
import tkinter as tk

root = tk.Tk()
entry = tk.Entry(root, width=25)
entry.insert(0, "Введіть ім’я користувача")
entry.pack()

root.mainloop()
```

Що відобразиться після виконання даного коду?

Варіанти відповідей:
- вікно з багаторядковим полем введення для введення тексту
- вікно з однорядковим полем, у якому спочатку показано текст “Введіть ім’я користувача”
- вікно з порожнім текстовим полем, бо insert() не спрацьовує для Entry
- вікно з кнопкою, яка дозволяє вводити текст

2.7 Яке з наведених тверджень найточніше описує принцип роботи віджета Radiobutton у Tkinter?

Варіанти відповідей:

- дозволяє користувачу вибрати кілька варіантів одночасно, подібно до Checkbutton
- використовується для введення текстових даних у форму
- дає змогу вибрати лише один варіант із групи взаємовиключних параметрів, пов’язаних через одну змінну
- служить для відображення повідомлень і не реагує на дії користувача

2.8 Розгляньте код:

```
from tkinter import *

root = Tk()
cb1 = Checkbutton(root, text="опція 1")
cb2 = Checkbutton(root, text="опція 2")
cb3 = Checkbutton(root, text="опція 3")

cb1.pack()
cb2.pack()
cb3.pack()

root.mainloop()
```

Що відображатиме вікно після запуску даної програми?

Варіанти відповідей:
- три незалежні прапорці з підписами «опція 1», «опція 2» та «опція 3», які можна ставити або знімати
- три залежні прапорці з підписами «опція 1», «опція 2» та «опція 3», які можна ставити або знімати
- три кнопки, при натисканні на які буде виконуватися якась команда
- три поля для введення тексту з відповідними підписами «опція 1», «опція 2» та «опція 3»

2.9 Основні операції з віджетом Text у Python: ...

Варіанти відповідей:
- insert, get, start, delete, replace
- editor, insert, get, delete, replace
- insert, get, delete, replace
- insert, get, pack, delete, replace

2.10 Яку роль відіграє параметр variable у віджетах Radiobutton і Checkbutton?

Варіанти відповідей:
- він визначає, яке значення буде відображатися на кнопці

- він зберігає стан вибору (значення) віджета та пов’язує його з відповідною змінною
- він використовується лише для задання тексту кнопки
- він задає функцію, яка викликається при натисканні кнопки

Тема 3. Особливості, властивості та робота з віджетами: Scale, Listbox, Canvas, Menu, MessageBox, звичайні вікна та діалогові

3.1 Яке призначення віджета Scale у Tkinter?

Варіанти відповідей:
- відображає список вибору з кількох варіантів
- дозволяє вводити текстові дані
- дозволяє користувачу вибрати числове значення, переміщаючи повзунок
- відображає зображення у вікні програми

3.2 Розгляньте наведений код. Яке з тверджень є правильним щодо його виконання?

```
import tkinter as tk

root = tk.Tk()
root.title("Приклад Listbox")
listbox = tk.Listbox(root, selectmode=tk.MULTIPLE)

listbox.insert(1, "Python")
listbox.insert(2, "Java")
listbox.insert(3, "C++")
listbox.insert(4, "Kotlin")
listbox.pack()

root.mainloop()
```

Варіанти відповідей:
- у вікні буде відображено список із чотирьох мов програмування, і користувач зможе вибрати лише одну
- у вікні буде відображено список із чотирьох елементів, і користувач зможе вибрати декілька елементів одночасно
- програма викличе помилку, оскільки метод insert() не підтримує рядкові значення
- віджет Listbox автоматично додасть вертикальну смугу прокрутки

3.3 Що відображає наведений нижче код після запуску програми?

```
import tkinter as tk

root = tk.Tk()
root.title("Приклад Canvas")
canvas = tk.Canvas(root, width=200, height=150, bg="white")
canvas.pack()

canvas.create_rectangle(30, 30, 120, 100, fill="lightblue", outline="blue")
canvas.create_oval(60, 50, 150, 120, fill="pink", outline="red")

root.mainloop()
```

Варіанти відповідей:

- вікно без елементів, бо Canvas не має методів для малювання, а прямокутник та овал будуть виведені збоку у вікні
- вікно з білим фоном, на якому зображено прямокутник і овал
- вікно з текстом «Приклад Canvas»
- вікно з кнопкою, яка малює фігури при натисканні

3.4 Що відбудеться після виконання цього коду?

```
import tkinter as tk

root = tk.Tk()
canvas = tk.Canvas(root, width=300, height=200, bg="white")
canvas.pack()

rect = canvas.create_rectangle(20, 30, 100, 80, fill="lightgreen")
oval = canvas.create_oval(120, 40, 200, 100, fill="yellow")
line = canvas.create_line(10, 150, 250, 150, fill="black", width=2)
polygon = canvas.create_polygon(60, 120, 90, 160, 30, 160, fill="orange")

# Зміна кольору прямокутника
canvas.itemconfig(rect, fill="blue")
canvas.move(oval, 50, 20)
canvas.move(polygon, 40, -10)

root.mainloop()
```

Варіанти відповідей:

- у вікні відобразиться лише пряма лінія, інші фігури не з’являться
- прямокутник стане синім, овал зміститься вправо і вниз, полігон відносно прямокутника – вправо і трохи вниз
- прямокутник стане світлозеленим, овал зміститься вліво і вниз, полігон – вліво і трохи вгору

- всі фігури зникнуть після зміни кольору прямокутника

3.5 Що відбудеться після запуску наведеної програми?

```
import tkinter as tk

def show_message():
    print("Обрано пункт меню!")

root = tk.Tk()
root.title("Приклад Menu")
menubar = tk.Menu(root)

file_menu = tk.Menu(menubar, tearoff=0)
file_menu.add_command(label="Відкрити", command=show_message)
file_menu.add_command(label="Зберегти", command=show_message)
file_menu.add_separator()
file_menu.add_command(label="Вихід", command=root.quit)
menubar.add_cascade(label="Файл", menu=file_menu)
root.config(menu=menubar)

root.mainloop()
```

Варіанти відповідей:

- програма створить кнопку «Файл» у головному вікні, але натискання на неї нічого не відображатиме, бо до меню не додано жодних пунктів
- у вікні з’явиться рядок меню з пунктом «Файл», при натисканні на який відкриється спадне меню з командами «Відкрити», «Зберегти», розділювачем (separator) і пунктом «Вихід»; вибір будь-якої команди викликає відповідну дію (вивід повідомлення або закриття програми)
- програма видасть помилку, оскільки метод add_separator() не можна використовувати без додаткових параметрів
- у вікні відобразиться меню, але всі пункти будуть неактивними, оскільки не вказано параметр tearoff=1 при створенні підменю

3.6 Яким чином у Tkinter можна створити нове додаткове вікно поверх головного, а також коректно закрити його під час виконання програми?

Варіанти відповідей:

- нове вікно створюється повторним викликом Tk(), а закривається автоматично при завершенні головного циклу
- для створення нового вікна використовується Frame(), а метод destroy() видаляє лише окремі віджети, але не вікна

- у Tkinter створення кількох вікон можливе за допомогою Tk(), проте не рекомендується, а закривається викликом методу destroy() для цього об’єкта
- створити нове вікно можна лише через модуль messagebox, а закриття виконується натисканням кнопки "ОК"

3.7 Яке призначення віджета messagebox у Tkinter?

Варіанти відповідей:
- використовується для створення стандартних діалогових вікон із повідомленнями, попередженнями чи питаннями до користувача
- служить для введення багаторядкового тексту користувачем
- відображає інформацію у вигляді текстового поля з можливістю редагування
- використовується для розміщення віджетів у вигляді таблиці

3.8 Який метод модуля messagebox у Tkinter використовується для створення вікна підтвердження операції (з кнопками Yes і No), та яке значення він повертає?

Варіанти відповідей:
- messagebox.showinfo() – повертає True, якщо користувач натиснув ОК
- messagebox.askyesno() – повертає True, якщо натиснуто Yes, і False, якщо No
- messagebox.showwarning() – повертає текст повідомлення
- messagebox.askquestion() – повертає рядок "OK" або "Cancel"

3.9 Яке призначення модуля filedialog у Tkinter і який метод використовується для вибору файлу користувачем?

Варіанти відповідей:
- filedialog використовується для введення тексту; метод askstring() дозволяє вводити назву файлу
- модуль filedialog служить для виведення повідомлень; метод showinfo() відкриває інформаційне вікно
- filedialog створює лише вікна підтвердження дій; метод askyesno() повертає True або False
- модуль filedialog використовується для створення діалогових вікон вибору файлів; метод askopenfilename() відкриває вікно для вибору одного файлу

3.10 Який модуль і метод у Tkinter використовуються для відкриття діалогового вікна вибору кольору та яке значення повертається після вибору?

Варіанти відповідей:
- модуль filedialog, метод askcolor(), який повертає лише назву кольору

- модуль colorchooser, метод askcolor(), який повертає кортеж із RGB-значенням і шістнадцятковим кодом кольору
- модуль messagebox, метод showcolor(), який повертає рядок із вибраним кольором
- модуль colorbox, метод choosecolor(), який повертає тільки RGB-представлення кольору

Тема 4. Дизайн графічного інтерфейсу користувача: бібліотека PyQt. ПЗ QtDesigner

4.1 Яке основне призначення бібліотеки PyQt5 у Python?

Варіанти відповідей:
- створення серверних застосунків для роботи з базами даних
- розробка графічних інтерфейсів користувача (GUI)
- написання скриптів для автоматизації системних процесів
- побудова нейронних мереж для машинного навчання

4.2 Яке основне призначення програми Qt Designer?

Варіанти відповідей:
- для написання SQL-запитів до бази даних
- для візуального створення графічних інтерфейсів користувача без написання коду
- для компіляції Python-програм у виконувані файли
- для налагодження мережевих підключень у PyQt5

4.3 Розгляньте уважно код. Що робить функція uic.loadUi('hello_form.ui', self) у наведеному коді?

```
from PyQt5 import QtWidgets, uic
import sys

class Ui(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui, self).__init__()
        uic.loadUi('hello_form.ui', self)
        self.show()

app = QtWidgets.QApplication(sys.argv)
window = Ui()
app.exec_()
```

Варіанти відповідей:

- завантажує інтерфейс користувача з файлу .ui, створеного в Qt Designer, і застосовує його до поточного вікна
- створює новий файл .ui на основі класу Ui
- викликає головний цикл подій програми
- імпортує усі віджети PyQt5 у поточний проєкт

4.4 Розгляньте уважно код. Що відбудеться після натискання кнопки pushButton у наведеній програмі?

```
from PyQt5 import QtWidgets, uic
import sys

class Ui(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui, self).__init__()
        uic.loadUi('hello_form.ui', self)
        self.show()
        self.pushButton = self.findChild(QtWidgets.QPushButton, 'pushButton')
        self.lineEdit = self.findChild(QtWidgets.QLineEdit, 'lineEdit')
        self.label = self.findChild(QtWidgets.QLabel, 'label')
        self.pushButton.clicked.connect(self.printButtonPressed)
    def printButtonPressed(self):
        imya = self.lineEdit.text()
        self.label.setText(imya)

app = QtWidgets.QApplication(sys.argv)
window = Ui()
app.exec_()
```

Варіанти відповідей:

- програма завершить свою роботу
- у віджеті label з’явиться текст, який користувач увів у lineEdit
- віджет lineEdit очиститься
- у консолі буде виведено текст, який користувач увів у lineEdit

4.5 Розгляньте уважно код. Яку роль у програмі виконує метод findChild()?

```
from PyQt5 import QtWidgets, uic
import sys

class Ui(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui, self).__init__()
        uic.loadUi('hello_form.ui', self)
        self.show()
```

```
        self.pushButton = self.findChild(QtWidgets.QPushButton, 'pushButton')
        self.lineEdit = self.findChild(QtWidgets.QLineEdit, 'lineEdit')
        self.label = self.findChild(QtWidgets.QLabel, 'label')
        self.pushButton.clicked.connect(self.printButtonPressed)
    def printButtonPressed(self):
        imya = self.lineEdit.text()
        self.label.setText(imya)

app = QtWidgets.QApplication(sys.argv)
window = Ui()
app.exec_()
```

Варіанти відповідей:

- використовується для пошуку дочірнього віджета за його типом і об’єктною назвою у формі
- створює новий дочірній віджет у вікні
- видаляє дочірній віджет із головного вікна
- змінює властивості знайденого віджета без звернення до його імені

4.6 Який формат файлу створює Qt Designer після збереження розробленої форми інтерфейсу?

Варіанти відповідей:

- .py
- .exe
- .ui
- .xml

4.7 Що відбудеться після натискання кнопки "Збільшити" у вікні, створеному за цим кодом?

```
from PyQt5 import QtWidgets, uic
import sys

class CounterApp(QtWidgets.QMainWindow):
    def __init__(self):
        super(CounterApp, self).__init__()
        uic.loadUi('counter_form.ui', self)
        self.pushButton = self.findChild(QtWidgets.QPushButton, 'pushButton')
        self.label = self.findChild(QtWidgets.QLabel, 'label')
        self.count = 0
        self.pushButton.clicked.connect(self.increase)
        self.show()

    def increase(self):
```

```
        self.count += 1
        self.label.setText(f"Лічильник: {self.count}")

app = QtWidgets.QApplication(sys.argv)
window = CounterApp()
app.exec_()
```

Варіанти відповідей:

- після натискання кнопки на етикетці з’явиться випадкове число
- кожне натискання кнопки збільшує значення лічильника на 1
- програма завершить роботу
- вікно оновлюється, але текст на етикетці не змінюється

Правильні відповіді

Змістовий модуль «Вступ до програмування мовою Python»

1.1 Гвідо ван Россум
1.2 ABC
1.3 вкладені інструкції об'єднуються в блоки за величиною відступів
1.4 print(), input()
1.5 це посилання на область пам'яті комп'ютера, в якій зберігається деякі дані (посилання на деякий об’єкт)
1.6 об’єктно-орієнтовна, імперативна, функціональна та структурна
1.7 інтерактивний режим та пакетний режим
1.8 при оголошенні змінної її тип не вказується
1.9 за допомогою символу "#"
1.10 портабельність на різних операційних системах без змін коду

2.1 аварійний стан, який відбувається в кодовій послідовності під час виконання програми
2.2 try, except, else, finally
2.3 використовуючи кілька блоків except в конструкції try-except
2.4 за допомогою оператора except as
2.5 math.exp(x+y)+5/(math.cos(y-x)+3)
2.6 переривання поточної ітерації циклу і переходу до наступної
2.7 дострокового припинення роботи циклу (for або while)
2.8 3, 0.001, '0b101'
2.9 1
2.10 34

3.2 [0, 1, 2, 3, 4]
3.2 my_list[::2]

3.3 ключі в словнику можуть бути будь-якого типу, але повинні бути унікальними,
оскільки вони ідентифікують кожен елемент
3.4 повертає значення за ключем, якщо ключ існує, в іншому випадку повертає значення
за замовчуванням
3.5 matrix = [[0 for j in range(3)] for i in range(3)]
matrix = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
3.6 (20, 30, 40)
3.7 GNIMMOR
3.8 PYTHON_IS_FUN
3.9 (1, 2, 3)
[10, 2, 3, 4, 5, 6]
3.10 Заміна елемента – t[0] = 10
3.11 {1, 2, 3, 4, 5, 6}
3.12 4
3.13 {'Anna', 'Bob', 'Eva'}
3.14 {'P': 'n'}
3.15 values

4.1 def ім’я_функції (список формальних параметрів): оператори тіла функції
4.2 значення None
4.3 записати після параметра символ = і вказати значення
4.4 f(a=5, b=3)
4.5 a – параметр за замовчуванням, b – іменований параметр
4.6 **
4.7 локальні змінні створюються всередині функції і доступні лише в ній, глобальні змінні створюються поза функцією і доступні у всіх функціях
4.8 lambda-функція – це анонімна функція, яку можна визначити без імені; синтаксис: lambda x, y: x + y
4.9 6
4.10 21

5.1 read()
5.2 фізичний файл – це файл на диску, логічний файл – це об’єкт у пам’яті програми, через який програма працює з даними
5.4f.readlines()
5.4 забезпечує виконання завершальних дій, таких як автоматичне закриття файлу
5.5 'r', 'w', 'x'
5.6 3
5.7 ['123\n', '456\n', '789']
5.8 cd
5.9 tXYZ
5.10 1 → c, 2 → d, 3 → e, 4 → a, 5 → b

6.1 окремий файл програми, з розширенням .py, який можна підключати до інших програм, щоб повторно використовувати код
6.2 отримати доступ до всього модуля, який підключаємо
6.3 підключати всі імена заданого модуля
6.4 для розділення програми на частини та повторного використання коду
6.5 1 – б, 2 – а, 3 – в
6.6 тестування окремих компонентів програми на їх коректність
6.7 визначення тестованих «юнітів», розробка тестових випадків, написання тестового коду, запуск тестів, аналіз результатів і виправлення помилок
6.8 1 – в, 2 – б, 3 – г, 4 – а
6.9 чи функція divide() правильно ділить числа і обробляє помилку при діленні на нуль
6.10 assertAlmostEqual() перевіряє, що результат ділення близький до очікуваного, а assertRaises() перевіряє, що викликається зазначене виключення

Змістовий модуль «Об'єктно-орієнтоване програмування мовою Python»
1.1 1 – b, 2 – a, 3 – a, 4 – b, 5 – b
1.2 1 – c, 2 – a, 3 – d, 4 – b
1.3 публічний, захищений, приватний
1.4 спеціальний метод, який автоматично викликається під час створення об'єкта класу
1.5 процес приховування внутрішніх даних об'єкта та обмеження доступу до них ззовні
1.6 інкапсуляція
1.7 використати гетери та сетери, які повертають або змінюють значення атрибута або через механізм коригування імен
1.8 це автоматичне перетворення імені приватного атрибута у формат _ІмʼяКласу__ІмʼяАтрибута, щоб уникнути випадкового доступу ззовні
1.9 виникне помилка доступу до приватного атрибута
1.10 деструктор викликається автоматично, коли об'єкт знищується

2.1 1 – c, 2 – b, 3 – a, 4 – d
2.2 атрибут, який належить класу і спільний для всіх екземплярів цього класу
2.3 визначається всередині класу, але не приймає параметр self; він не залежить від конкретного об'єкта і за потреби може існувати як звичайна функції поза класом
2.4 коли метод виконує функціональність, яка не залежить від конкретного об'єкта класу, але логічно належить класу
2.5 is_valid_salary, бо метод не використовує self і може перевіряти значення зарплати незалежно від конкретного об'єкта; його можна зробити статичним, додавши декоратор @staticmethod над методом
2.6 turtle.forward(100)
2.7 A

2.8 t.forward(150)
2.9 коли до складу створюваного класу входять екземпляри інших класів
2.10 клас Car містить об’єкт класу Engine як частину своєї структури, що є прикладом композиції

3.1 механізм, що дозволяє одному класу використовувати атрибути й методи іншого класу, class Child(Parent):
3.2 повторне використання коду, можливість перевизначати методи, спрощення розширюваності програм, підвищення гнучкості архітектури
3.3 викликає метод із батьківського класу
3.4 багаторівневе
3.5 employee: anna, manages 10 people, department: it
3.6 dog – підклас класу animal, car – підклас класу vehicle, student – підклас класу person, rectangle – підклас класу shape
3.7 MRO (Method Resolution Order)
3.8 A
3.9 A
3.10 виникає конфлікт імен методів або атрибутів у батьківських класах, який вирішується за допомогою порядку пошуку методів (MRO)

4.1 використання одного інтерфейсу для об’єктів різних типів
об’єкти двох або більше класів можуть реагувати по-різному на однакові команди
4.2 len() для об’єктів різних типів (рядків, списків, кортежів тощо)
print() для виведення різних типів даних
4.3 x = 3.0
x1 = 2.0, x2 = 1.0
Невідомий тип рівняння
4.4 віртуальність (динамічне зв’язування)
4.5 вони дозволяють створювати узагальнений код, який працює як з об'єктами базового класу, так і з об'єктами будь-якого з його нащадків
4.6
```
def add(a, b):
    return a + b

print(add(3, 5))
print(add("Hello, ", "World!"))
```
4.7 асоціація забезпечує структурний зв’язок між екземплярами різних класів, дозволяючи об’єктам посилатися один на одного
4.8 1 → a, 2 → b, 3 → c, 4 → d, 5 → e
4.9 1 → композиція
2 → агрегація
3 → залежність
4 → спадкування
5 → реалізація

4.10 потужність асоціації – це кількість об’єктів одного класу, які можуть бути пов’язані з об’єктами іншого класу; потужність асоціації буває трьох типів: один-до-одного, один-до-багатьох, багато-до-багатьох

5.1 це поля та методи з подвійними підкресленнями на початку і в кінці імені, що визначають поведінку об’єктів у стандартних операціях (наприклад, __init__, __str__, __dict__)
5.2 1 – e, 2 – d, 3 – b, 4 – c, 5 – a
5.3 data.__len__(), len(data)
5.4 Python 101 – Іван Петренко
True
False
{'title': 'Python 101', 'author': 'Іван Петренко'}
5.5Python 101 & Python для початківців написав Іван Петренко & Марія Іваненко
P
{'title': 'Python 101', 'author': 'Іван Петренко'}
<class '__main__.Book'>
5.6 щоб змінювати стандартну поведінку операторів для об’єктів власних класів, це робить код більш зрозумілим та дозволяє використовувати оператори для роботи з об’єктами так, як з простими типами даних
5.7 1–A, 2–B, 3–C, 4–D, 5–E
5.8 1–E, 2–C, 3–B, 4–D, 5–A, 6–F
5.9 14
5.10 13

6.1 якщо створення його екземплярів не має сенсу і не передбачається
6.2 особливість – він не має реалізації, призначення – декларація методу, тобто визначення його сигнатури
6.3 використати модуль abc та позначити хоча б один метод як абстрактний за допомогою @abstractmethod
6.4 абстрактні класи варто застосовувати, коли потрібно визначити загальний інтерфейс для групи пов’язаних класів, забезпечуючи структуру, безпеку та чіткість, а звичайні класи – коли необхідна більша гнучкість у створенні об’єктів і не потрібно обмежувати реалізацію
6.5 result = eval("2 * 4 + 6")
print(result)

code = "x = 10\nprint(x)"
exec(code)
6.6 це спеціальні функції, які дозволяють змінювати або розширювати поведінку інших функцій без зміни їхнього коду, зазвичай через використання символу @
6.7 log – це декоратор, оскільки вона приймає іншу функцію як аргумент і повертає нову функцію з розширеною поведінкою
6.8 *****

*****
28

6.9 Запуск функції...
× Виконуємо множення результату...
× Множення завершено.
Завершення функції.
11

6.10 декоратор класу – це функція, яка приймає клас як аргумент і може змінювати або повертати нову версію цього класу, розширюючи його можливості

6.11 ПРИВІТ, Я ОЛЕНА!!!

6.12 Оригінальні аргументи: 2, 3
80

Змістовий модуль «Подійно-орієнтоване програмування у Python»

1.1 1, 3, 5, 6

1.2 віджети – це графічні елементи інтерфейсу, такі як кнопки, поля введення, мітки, які дозволяють взаємодіяти з користувачем. до них належать button, label, entry, text та ін

1.3 1 – H, 2 – G, 3 – A, 4 – B, 5 – F, 6 – I, 7 – J, 8 – C, 9 – D, 10 – E

1.4 при натисканні кнопки змінюється назва та розміри вікна і фон стає зеленим; при натисканні клавіші f вікно збільшується та фон стає червоним

1.5 замінити import Tkinter as tk на import tkinter as tk, root.tk() на tk.Tk(), root.Title() на root.title(), root.Geometry("400*300") на root.geometry("400x300"), root.Bind() на root.bind(), root.Mainloop() на root.mainloop()

1.6 розташовує віджети відносно країв вікна (зверху, знизу, зліва, справа) та автоматично підлаштовує їхній розмір

1.7 fill

1.8 label2 та label3

1.9 grid(row=0, column=0), grid(row=0, column=1), grid(row=1, column=0, columnspan=2)

1.10 bind

2.1 label = ttk.Label(text="Hello Мітка", font=("Arial", 24))

2.2 слугує контейнером для розміщення інших віджетів

2.3 Text, Listbox, Canvas

2.4 відобразиться багаторядкове текстове поле розміром 3×20 символів із попередньо вставленим на початок текстом

2.5 після запуску програми з’явиться текстове поле з вертикальною прокруткою, що працює синхронно з віджетом Text

2.6 вікно з однорядковим полем, у якому спочатку показано текст “Введіть ім’я користувача”

2.7 дає змогу вибрати лише один варіант із групи взаємовиключних параметрів, пов’язаних через одну змінну

2.8 три незалежні прапорці з підписами «опція 1», «опція 2» та «опція 3», які можна ставити або знімати
2.9 insert, get, delete, replace
2.10 він зберігає стан вибору (значення) віджета та пов’язує його з відповідною змінною

3.1 дозволяє користувачу вибрати числове значення, переміщаючи повзунок
3.2 у вікні буде відображено список із чотирьох елементів, і користувач зможе вибрати декілька елементів одночасно
3.3 вікно з білим фоном, на якому зображено прямокутник і овал
3.4 прямокутник стане синім, овал зміститься вправо і вниз, полігон відносно прямокутника – вправо і трохи вниз
3.5 у вікні з’явиться рядок меню з пунктом «Файл», при натисканні на який відкриється спадне меню з командами «Відкрити», «Зберегти», розділювачем (separator) і пунктом «Вихід»; вибір будь-якої команди викликає відповідну дію (вивід повідомлення або закриття програми)
3.6 у Tkinter створення кількох вікон можливе за допомогою Tk() проте не рекомендується, а закривається викликом методу destroy() для цього об’єкта
3.7 використовується для створення стандартних діалогових вікон із повідомленнями, попередженнями чи питаннями до користувача
3.8 messagebox.askyesno() – повертає True, якщо натиснуто Yes, і False, якщо No
3.9 модуль filedialog використовується для створення діалогових вікон вибору файлів; метод askopenfilename() відкриває вікно для вибору одного файлу
3.10 модуль colorchooser, метод askcolor(), який повертає кортеж із RGB-значенням і шістнадцятковим кодом кольору

4.1 розробка графічних інтерфейсів користувача (GUI)
4.2 для візуального створення графічних інтерфейсів користувача без написання коду
4.3 завантажує інтерфейс користувача з файлу .ui, створеного в Qt Designer, і застосовує його до поточного вікна
4.4 у віджеті label з’явиться текст, який користувач увів у lineEdit
4.5 використовується для пошуку дочірнього віджета за його типом і об’єктною назвою у формі
4.6 .ui
4.7 кожне натискання кнопки збільшує значення лічильника на 1

СПИСОК СИКОРИСТАНИХ ДЖЕРЕЛ

1. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування / Ерік Маттес. – Київ : Видавництво Старого Лева, 2021. – 600 с.
2. Задачі з програмування мовою Python / В. Сліпченко. – 2021. – (посібник із задачами).
3. Програмування в Python. Теорія і практика / Васильєв О. М. – Київ : Ліра-К, 2023. – ISBN 978-617-520-513-6. – Режим доступу: https://lira-k.com.ua/preview/13141.pdf
4. Путівник мовою програмування Python / Олександр Мізюк. – [Е-ресурс] – 2024. – Режим доступу: https://pythonguide.rozh2sch.org.ua/pythonguide.rozh2sch.org.ua
5. Програмування для всіх: основи Python – Курс на платформі Prometheus (українською мовою) – Режим доступу: https://prometheus.org.ua/prometheus-free/python-fundamentals-for-everyone/
6. Уроки Python – Курс для початківців на сайті itProger – Режим доступу: https://itproger.com/ua/course/python?utm_source
7. Практикум з програмування мовою Python – Режим доступу: https://pythonexercises.rozh2sch.org.ua/
8. Уроки програмування мовою Python [Електронний ресурс] // Acode – онлайн-школа програмування. – Режим доступу: https://acode.com.ua/lessons-python/
9. Офіційна документація Python 3. Підручник користувача [Електронний ресурс] // Python Documentation (українська версія). – Режим доступу: https://docs.python.org/uk/3/tutorial/index.html
10. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. 180 с. – Режим доступу: https://epub.chnpu.edu.ua/jspui/bitstream/123456789/5584/1/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%BC%D0%BE%D0%B2%D0%BE%D1%8E%20Python.pdf

УДК 004.43(079.1)
Г 54

Електронне мережне навчальне видання

Л. Я. Глинчук

ТЕСТИ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ З ПРОГРАМУВАННЯ МОВОЮ PYTHON

для студентів спеціальностей F3 Комп’ютерні науки та F5 Кібербезпека та захист інформації першого (бакалаврського) рівня