ПРАКТИЧНА РОБОТА 06
Структурне програмування мовою Python
План

1. Вбудовані структури даних у мові Python.
2. Організація Python-коду за допомогою функцій.
3. Налагодження Python-коду.

Система оцінювання

№	Тема	К-ть балів
1.	*Захист принаймні одного завдання з роботи*	1
2.	Вбудовані структури даних у мові Python	1,6*
3.	Організація Python-коду за допомогою функцій	1,4*
4.	Налагодження Python-коду	0,6*
5.	*Здача звіту*	0,4
	Всього	**5**

* – діє бонусна система

Вбудовані структури даних у мові Python

1. [0,1 бала] При аналізі даних, зібраних у результаті експерименту, часто може виникати потреба у видаленні найбільш екстремальних значень до виконння обчислень. Ваша програма має видалити зі списку n (ціле додатнє число, введене користувачем), по троє найбільших та найменших значень і вивести решту значень (порядок не важливий). Також забезпечте коректну обробку ситуації, коли у списку не вистачатиме елементів для видалення.
2. [0,1 бала] Напишіть програму, яка зчитує від користувача слова, кожне з нового рядка. Ввід закінчується порожнім рядком, після чого виводиться перелік всіх унікальних слів, які були введені. Наприклад, для вводу

   first
   second
   first
   third
   second

   програма може вивести

   first
   second
   third
3. [0,2 бала] Код Морзе – схема кодування, яка використовує точки та тире для представлення цифр і букв. Напишіть програму, яка буде використовувати словник для зберігання відображення букв і цифр на код Морзе (таблиця).

Letter	Code	Letter	Code	Letter	Code	Number	Code
A	.-	J	.---	S	...	1	.----
B	-...	K	-.-	T	-	2	..---
C	-.-.	L	.-..	U	..-	3	...--
D	-..	M	--	V	...-	4	....-
E	.	N	-.	W	.--	5	.....
F	..-.	O	---	X	-..-	6	-....
G	--.	P	.--.	Y	-.--	7	--...
H	....	Q	--.-	Z	--..	8	---..
I	..	R	.-.	0	-----	9	----.

Ваша програма має читати повідомлення користувача та перетворювати його в код Морзе з наступним виводом закодованого тексту. Пробіли між словами слід залишати, а решту символів, що не є цифрою або буквою – ігнорувати. Наприклад, Hello, World! буде мати код

.... . .-.. .-.. --- .-- --- .-. .-.. -..

4. 0,2 бала Напишіть додаток, який використовує генерування випадкових чисел для створення речень. Використовуйте 4 списки рядків: article, noun, verb та preposition. Створіть речення, обравши з кожного масиву слова в такому порядку: артикль, іменник, дієслово, прийменник, артикль та іменник. Після вибору кожного слова виконайте його конкатенацію з реченням. Слова мають відокремлюватись пробілами. Загальне речення має починатись з великої літери та закінчуватись крапкою. Додаток має згенерувати та відобразити 10 речень.
   Артиклі: "the", "a", "one", "some" та "any";
   Іменники: "boy", "girl", "dog", "town" та "car";
   Дієслова: "drove", "jumped", "ran", "walked" та "skipped";
   Прийменники: "to", "from", "over", "under" та "on".
5. 0,2 бала На більшості кнопочних телефонів для набору текстового повідомлення використовують цифрову клавіатуру. Оскільки кожна кнопка прив’язана до кількох букв, її потрібно натискати декілька разів для вибору потрібної відповідно до таблиці. Натиснення цифри 2, 3, 4 або 5 разів генерує другу, третю, четверту або п’яту букву для цієї кнопки.

Кнопка	**Символи**
1	. , ? ! :
2	A B C
3	D E F
4	G H I
5	J K L
6	M N O

7	P Q R S
8	T U V
9	W X Y Z
0	пробіл

Напишіть програму, яка показує, скільки та яких натиснень кнопок потрібно, щоб набрати текст користувача. Сконструюйте словник, який відображає кожну букву чи символ на кнопку, яку потрібно натиснути. Наприклад, для тексту Hello, World! програма має вивести 4433555555566110966677755531111. Забезпечте коректну роботу як з малими, так і великими літерами. Ігноруйте символи, не представлені в таблиці.

6. 0,2 бала Багато фінансових документів передбачають запис суми грошей як у числовій, так і в текстовій формі. Напишіть програму, яка буде конвертувати суми від 0 до 10 000 грн. у текстове представлення. Наприклад, 3726 грн – три тисячі сімсот двадцять шість гривень. Використовуйте списки для переліку можливих текстових представлень для одиниць, десятків, сотень та тисяч гривень.
7. 0,2 бала Створіть плейліст, кожна пісня якого має назву, виконавця, альбом та тривалість. Плейліст дозволяє додавати та видаляти пісню, отримувати розмір та очищати його, а також виконувати форматований вивід композицій і повну тривалість його відтворення.
8. 0,2 бала Невелика авіакомпанія потребує систему автоматизації резервування місць у літаку на 12 місць. Ваша програма має пропонувати наступні альтернативи:
   Натисніть 1 для першого класу (місця 1-6)
   Натисніть 2 для економного класу (місця 7-12).
   Потім Ваш застосунок повинен відображати посадкові місця, вказуючи на замовлене місце та його клас.
   Використовуйте список булевих значень для представлення посадкових місць у літаку. Ваш застосунок має ніколи не давати резервувати вже зарезервовані місця, а повинен видавати перше вільне місце в класі. Коли місця з економного класу закінчились, програма повинна запропонувати квиток першого класу. Якщо таке не підходить, програма виведе повідомлення «Наступний виліт через 3 години».
9. 0,2 бала Напишіть гру «Однорукий бандит» для консолі. У процесі будуть відображатись три випадкових символи з переліку:
   Cherry - u"\U0001F352",
   Bell - u"\U0001F514",
   Lemon - u"\U0001F34B",

Orange - u"\U0001F34A",
Star - u"\u2606",
Skull - u"\U0001F480"

Гравцю видається кредит у розмірі 100 грн., а кожний запуск коштує 5 грн. Якщо «однорукий бандит» показує два однакових символи, користувач виграє 10 грн, а якщо три – 25 грн. У випадку випадіння трьох дзвіночків (bells) сума виграшу складає 100 грн. Якщо випадає два черепи – гравець додатково втрачає 5 грн, а при отриманні трьох черепів вся сума згорає. Гравець може зняти вигране до початку нової спроби або продовжувати грати, поки на рахунку будуть гроші.

Організація Python-коду за допомогою функцій

1. 0,1 бала Число Капрекара – це невід'ємне ціле число, квадрат якого в цій системі можна розбити на дві частини, сума яких дає початкове число. Наприклад, 45 є числом Капрекара, оскільки 45 2 = 2025, причому 20 + 25 = 45. Напишіть функцію, яка отримуватиме число та повертатиме дві частини, якщо це число Капрекара. Інакше функція поверне -1. Протестуйте роботу функції на різних числах.
2. 0,3 бала Редакційна відстань (edit distance) між двома рядками є мірою їх схожості. Чим менша редакційна відстань, тим більше схожі між собою рядки відносно мінімальної кількості вставок, видалень та замін при перетворенні одного рядка в інший.
   Наприклад, рядки kitten та sitting. Перший рядок можна перетворити в другий, якщо: замінити k на s, замінити e на i, вставити g в кінець рядка. Три – це найменша кількість операцій, що можна виконати для перетворення kitten у sitting.
   Напишіть рекурсивну функцію, яка обчислює редакційну відстань між 2 рядками. Використовуйте наступний алгоритм:

```
Візьмемо рядки s і t
If довжина s = 0
   Повернути довжину t
Else if довжина t = 0
   Повернути довжину s
Else
   cost = 0
   If останній символ в s не дорівнює останньому символу в t
      cost = 1
```

d1 = (редакційна відстань між всіма символами, крім останнього в s та всіма символами в t) + 1
d2 = (редакційна відстань між всіма символами в s та всіма символами в t, крім останнього) + 1
d3 = (редакційна відстань між всіма символами, крім останнього, в s та всіма символами в t, крім останнього) + 1
Return min(d1, d2, d3)

Використайте дану рекурсивну функцію, щоб написати програму, яка зчитує 2 рядки від користувача і виводить редакційну відстань між ними.

3. 0,1 бала Опишіть рекурсивну функцію GCD, яка повертає найбільший спільний дільник чисел x та y. Обчислення НСД відбувається рекурсивним чином відповідно до формули

$$GCD(x,y) = \begin{cases} x, & \text{якщо } x = y \\ GCD(x-y,y), & \text{якщо } x > y \\ GCD(x,y-x), & \text{якщо } x < y \end{cases}$$

4. 0,2 бала Напишіть функцію, яка перевірятиме номер кредитної картки на валідність. Наприклад, для номеру картки 4578 4230 1376 9219 маємо окремі цифри:

4-5-7-8-4-2-3-0-1-3-7-6-9-2-1

Починаючи з першої цифри (нумерація з 1) і до передостанньої, домножаємо на 2 кожну цифру, що стоїть на місці з непарним номером:

8-5-**14**-8-**8**-2-**6**-0-**2**-3-**14**-6-**18**-2-**2**

Якщо отримується двоцифрове значення, додайте його цифри та замініть на їх суму: 8-5-**5**-8-8-2-6-0-2-3-**5**-6-**9**-2-2

Нарешті, додайте всі отримані цифри:

8 + 5 + 5 + 8 + 8 + 2 + 6 + 0 + 2 + 3 + 5 + 6 + 9 + 2 + 2 = 71

Сума отриманого числа з останньою цифрою картки повинна давати число, кратне 10: 71 + 9 = 80

Інакше номер картки некоректний. Введіть номер картки як рядок з клавіатури та виведіть результат роботи функції: Correct або Incorrect.

5. 0,1 бала Генерування паролю шляхом вибору випадкових символів дає відносно безпечний, проте загалом складний для запам’ятовування пароль. Альтернативним способом є конструювання паролю з двох агнлійських слів з подальшою конкатенацією. Такий пароль простіше запам’ятати, хоча і зламати також.

Напишіть програму, яка містить список слів, обирає два випадковим чином та з’єднує їх в одне. При створенні паролю переконайтесь, що загальна його

довжина знаходиться в діапазоні від 8 до 10 символів, а кожне використане слово складається принаймні з трьох букв. Зробіть так, щоб у паролі кожне слово починалось з великої літери та виведіть пароль для користувача.

6. 0,2 бала Щоб зашифрувати повідомлення за допомогою шифру частоколу, переписуємо його у відповідному вигляді. Наприклад, для слова «криптографія» з висотою частоколу 2 отримаємо таку схему: к^{Р}и^{П}т^{О}г^{Р}а^{Ф}і^{Я}. Далі зчитуємо текст рядками, почавши з верхнього. В результаті отримуємо криптотекст «рпорфякитгаі». "Висоту" частоколу (секретний ключ шифру) можна вибирати, що й буде запропоновану користувачу.
   Напишіть функції, що будуть зашифровувати та розшифровувати введений користувачем текст за допомогою введеного ключа – висоти частоколу.
7. 0,2 бала Трикутник Паскаля – це трикутник чисел, який містить біноміальні коефіцієнти, що знаходяться за формулою

$$a_{nr} = \frac{n!}{r!\,(n-r)!}$$

   Напишіть функцію, яка будуватиме трикутник Паскаля заданої висоти (аргумент функції). Наприклад,

```
      1
    1   1
  1   2   1
1   3   3   1
1   4   6   4   1
```

   Зверніть увагу на форматування виводу трикутника: врахуйте відповідні відступи.
8. 0,2 бала Секретна інформація часто редагується чи видаляється з документів перед їх публікуванням. Після публікації документів відредагований текст часто заміняється чорними прямокутниками.
   Напишіть функцію, яка редагує всі секретні слова, заміняючи їх зірочками, навіть якщо слово з’являється всередині іншого слова. Передбачено існування списку секретних слів. Виведіть відредаговану версію початкового тексту на екран.
   Корисним може бути метод replace() для роботи з рядками. Доповніть програму незалежністю від регістру літер у тексті, тобто слова exam, Exam, ExaM та EXAM будуть вважатись однаковими.

Налагодження Python-коду

1. ***(Налагодження коду)*** [0,6 бала] Існує набагато потужніший спосіб розібратись в тому, що робить програма, ніж вводити print-інструкції в коді, - використовувати відладник (debugger).
   1) Розпочніть з простої програми, код якої записано нижче.
      name = 'Ваше ім'я та прізвище'
      greeting = 'Привіт, ' + name
      print(greeting)
      Збережіть код у файлі pdb_exercise_1.py.
      Викличте скрипт з використанням відладника за допомогою командного рядка:
      python -m pdb pdb_exercise_1.py
      Відладник pdb (Python Debugger) покаже наступний рядок коду, який буде виконуватись (-> ***name = 'Ваше ім'я та прізвище'***), а також запрошення до взаємодії з відладником (Pdb).
      Введіть ***n*** (скорочено від next), щоб відобразити на екрані наступний рядок для виконання.
      Тепер перевіримо значення змінної name. Введіть у командний рядок ***p name*** (p – скорочено від print). Повинно вивестись Ваше ім'я та прізвище. Знову введіть ***n*** для переходу до наступної команди. У змінній greeting має бути рядок 'Привіт, Ваше ім'я та прізвище'.
      При налагодженні коду дуже просто загубитись у тому, який рядок зараз налагоджується. Для того, щоб у цьому розібратись, введіть ***l*** (від list). На екрані з'явиться код поточного файлу та рядок, який налагоджується, вказаний стрілкою - >.Знову введіть n, щоб виконати останню команду. Тепер можна ввести q та вийти з відладника. Зробіть скриншоти Ваших дій та додайте їх у звіт.
   2) Потренуємось налагоджувати роботу функцій. Створіть новий файл pdb_exercise_2.py та скопіюйте у нього відповідний код:

      ```
      def greet(name):
          greeting = 'Привіт, ' + name
          return greeting

      greeting = greet('Ваше ім'я та прізвище')
      print(greeting)
      ```

      Запустіть відладник, ввівши в командному рядку

python -m pdb pdb_exercise_2.py

Після запуску введіть ***c*** (від continue execution). Результат виводу наприкінці матиме вигляд
-> def greet(name):

Тричі введіть n, щоб було виведено всі три рядка тексту основної функції (не заходячи в greet()). Натисніть ***c***, щоб перезапустити програму та один раз ***n***, щоб отримати рядок, де викликається функція greet(). Цього разу введемо ***s*** (від step), щоб зайти у функцію greet() та ще 5 разів введемо ***n***. Зверніть увагу на відмінності у виводі та відобразіть їх за допомогою скриншотів у звіті.
Наприкінці цієї вправи розглянемо команду r (від return). Вона схожа до команди c, проте замість налагодження до кінця програми відбувається запуск програми лише до кінця функції.

Перезапустимо програму, ввівши спочатку ***c***, а потім ***n*** та ***s***. Тепер Ви маєте знаходитись у функції greet(). Для перевірки правильності використайте ***l***.

Тепер натисніть ***r*** (“return”). Зауважте, що відбулося негайне переміщення у кінець функції, де має відбуватись повернення значення.

3) Працюючи з цим інструментом нечасто, дуже просто забути назви команд та їх суть. Проте можна використовувати команду help, яка відновить інформацію в пам’яті.
(Pdb) help
Викличте help для команд, які розглядались до цього:
(Pdb) help n
(Pdb) help s
(Pdb) help c
(Pdb) help r
(Pdb) help l
(Pdb) help help

4) До цього моменту не було помилок у скриптах, які довелось би виправляти. Скопіюйте представлений нижче код у файл з назвою pdb_exercise_4.py.

```
import sys

def magic(x, y):
    return x + y * 2

x = sys.argv[1]
y = sys.argv[1]

answer = magic(x, y)
print('The answer is: {}'.format(answer))
```

Нехай скрипт запускається з числами 1 та 50 (очікуємо результат 101).
python pdb_exercise_4.py 1 50
Чи вивело відповідь 101?
Тепер замість вставки print-інструкцій по всьому коду, перевіримо код у відладнику коду:

python -m pdb pdb_exercise_4.py 1 50

Доберемось до точки, де відбувається доступ до змінних *x* та *y*.

```
(Pdb) n
(Pdb) n
(Pdb) n
(Pdb) n
```

Спочатку переглянемо, які атрибути доступні в скоупі програми. Це можна зробити за допомогою ***p*** (відповідає print).

```
(Pdb) p dir()
```

Також є ***pp*** (pretty print).

```
(Pdb) pp dir()
```

Чому дорівнює *x*?

```
(Pdb) p x
```

Зауважте, що можна використовувати Python-код у відладнику. Визначимо тип *x*:

```
(Pdb) type(x)
```

Виконуючи Python-код, наприклад, можна динамічно змінити вхідні змінні:

```
(Pdb) x = int(x)
(Pdb) y = int(y)
```

Перевіримо тепер значення до запуску програми.

```
(Pdb) p x, y
```

Чому *y* дорівнює 1, а не 50? Переглядаючи код, було встановлено, що забули оновити індекс при копіюванні вхідного parsing-рядка.

Змінимо значення *y* на 50 у відладнику до перевірки на відповідність роботи коду очікуванням, дозволивши пропрацювати йому до кінця.

```
(Pdb) y = 50
(Pdb) c
```

5) До цього часу ми проходили по скриптах від початку до кінця. Проте працюючи у більших програмах це часто непрактично. Для моделювання такої ситуації скопіюйте нижче наведений код у файл з назвою pdb_exercise_5.py.

```
import time

def slow_subtractor(a, b):
    """Return a minus b."""
    time.sleep(5)
    return a - b

some = slow_subtractor(12, 8)
crazy = slow_subtractor(12, 78)
scientific = slow_subtractor(56, 31)
experiment = slow_subtractor(101, 64)

total = some + crazy + scientific + experiment

experimental_fraction = experiment / total
```

Після запуску коду отримаємо ZeroDivisionError. Прохід по коду у відладнику може бути annoying, оскільки потрібно натискати n кожного разу, коли викликається функція slow_subtraction(). Замість цього додаток точку розриву (breakpoint) перед рядком, який генеруватиме помилку. Це виконується за допомогою імпортування модуля pdb та використання функції pdb.set_trace().

```
import time

def slow_subtractor(a, b):
    """Return a minus b."""
    time.sleep(5)
    return a - b

some = slow_subtractor(12, 8)
crazy = slow_subtractor(12, 78)
scientific = slow_subtractor(56, 31)
experiment = slow_subtractor(101, 64)
```

```
total = some + crazy + scientific + experiment

import pdb; pdb.set_trace()
experimental_fraction = experiment / total
```

Якщо запустимо код тепер, то перейдемо в режим налагодження до виконання проблемного рядку коду.

```
python pdb_exercise_5.py
(Pdb) p total
(Pdb) p some, crazy, scientific, experiment
```

Зі змінною crazy буде робитись щось незрозуміле. Можливо, вхідні аргументи (input arguments) було задано неправильно.