

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи програмування на Python»

для бакалаврів спеціальності 123 "Комп'ютерна інженерія"
денної форми навчання

Частина II

2023

Методичні вказівки до лабораторних робіт з дисципліни «Основи програмування на Python» для бакалаврів спеціальності 123 "Комп'ютерна інженерія", денної форми навчання.. Частина II / Укл.: А.В. Тіменко, Н.А. Куликовська – Запоріжжя: НУ «Запорізька політехніка», 2023. – 17 с.

Укладач: А.В. Тіменко, старший викладач
Н.А. Куликовська, старший викладач

Рецензент: М.Ю. Тягунова, к.т.н., доцент

Відповідальний
за випуск: А.В. Тіменко, старший викладач

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 5
від 30 січня 2023р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 6
від 31 січня 2023 р.

ЗМІСТ

1	ЛАБОРАТОРНА РОБОТА. СЛОВНИКИ І МНОЖИНИ.....	4
1.1	Теоретичні відомості	4
1.2	Хід роботи	9
1.3	Зміст звіту	10
1.4	Контрольні питання	10
2	ЛАБОРАТОРНА РОБОТА. МОДУЛІ І ПАКЕТИ	12
2.1	Теоретичні відомості	12
2.2	Хід роботи	14
2.3	Зміст звіту	15
2.4	Контрольні запитання	15
	ЛІТЕРАТУРА.....	17

1 ЛАБОРАТОРНА РОБОТА №1

СЛОВНИКИ І МНОЖИНИ

Мета роботи: отримати навички програмування на мові Python з використанням словників і множин.

1.1 Теоретичні відомості

У мові Python є можливість використовувати словники. Словники – це вбудований тип даних, який є асоціативним масивом або хешем і базується на відображенні пар типу (ключ:значення).

Переваги застосування словників:

- з допомогою словників можна розробляти ефективні структури даних;
- для словників не потрібно писати вручну алгоритми пошуку даних, оскільки ці операції вже є реалізовані;
- словники можуть містити об'єднані дані у вигляді записів;
- словники ефективні при представленні розріджених структур даних.

Основні характеристики словників наступні:

- у словниках доступ до елементів виконується за ключем, а не за індексом. Словники визначають взаємозв'язок пар ключ-значення. За ключем відбувається доступ до значення. Якщо у словнику реалізовувати доступ до значення за індексом, то в цьому випадку індекс представляє собою ключ, а не зміщення відносно початку;
- словники представляють неупорядковані колекції довільних об'єктів. У словниках елементи (ключі) зберігаються у невизначеному порядку. Порядок формування елементів у словнику визначає інтерпретатор. Це необхідно для забезпечення більш швидкого пошуку.
- словники мають змінну довжину. Кількість елементів у словнику може збільшуватись або зменшуватись;
- гетерогенність – словники можуть містити об'єкти будь-яких типів;
- довільна кількість вкладень – словники мають можливість створення довільної кількості рівнів вкладень, оскільки словники можуть містити списки, інші словники тощо;

– словники відносяться до категорії змінюваних об'єктів. Тому, у словниках немає сенсу використовувати операції які мають фіксований порядок слідування елементів (наприклад, конкатенація);

– словники є таблицями посилань на об'єкти або хеш-таблицями і відносяться до об'єктів-відображень. Це означає, що у словниках об'єкти відображають (представляють) ключі на значення.

Між словниками і списками існують наступні основні відмінності:

– списки є впорядкованими колекціями, словники не є впорядкованими колекціями;

– у списках елементи витягуються з допомогою зміщення, яке визначає позицію елементу в списку. У словниках елементи витягуються за допомогою ключа;

– на відміну від списків у словниках немає підтримки операцій над послідовностями (наприклад, конкатенація, витягування зрізу та інше);

– списки є масивами посилань на об'єкти. Словники є масивами невпорядкованих таблиць посилань на об'єкти, які підтримують доступ до об'єктів за ключем.

У мові Python існує велика кількість операцій та методів для обробки словників. Усі ці засоби коротко перераховані тут:

– стандартний метод `len()` – визначає кількість елементів у списку;

– операція `D[key]` – доступ до елементу словника `D` за значенням ключа `key`;

– операція `del` – видалення елемента за ключем;

– операції `in`, `not in` – визначення наявності чи відсутності ключа у словнику;

– метод `iter()` – отримати ітератор за ключами словника;

– метод `clear()` – видалити всі елементи зі словника;

– метод `copy()` – вернути копію словника;

– метод `dict.fromkeys()` – створити новий словник з ключів та значень;

– метод `get()` – отримати значення за ключем;

– метод `items()` – вернути представлення елементів словника;

– метод `keys()` – отримати нове представлення ключів словника;

– метод `pop()` – видалення елементів зі словника з поверненням значення;

- метод `popitem()` – витягування зі словника довільної пари ключ:значення;
- метод `setdefault()` – встановити елемент за замовчуванням;
- метод `update()` – оновлення словника за заданим списком пар ключ:значення;
- метод `values()` – отримати список значень зі словника.

У словниках, в якості ключів можуть використовуватись числові типи: цілі або дійсні (з плаваючою комою). У випадку представлення ключів числовими типами можна виділити такі особливості:

- при пошуку потрібного значення за ключем використовується проста операція порівняння. Якщо в якості ключа вказувати значення різних числових типів, наприклад 1 і 1.0, то такі значення вважаються взаємозамінними (звертаються до одного й того самого елементу словника);

- представлення ключа числовим типом з плаваючою комою не рекомендується, оскільки значення цих типів є наближеними. Якщо в якості ключа потрібно вказати числовий тип, то для цього доцільно використовувати цілі числові типи.

Словник можна створити зручним (природнім) способом з допомогою оператора присвоювання `=`.

Лістинг 1.1 - Створення словника з допомогою оператора присвоювання `=`

```
# Створення словника
# з допомогою оператора присвоювання

# 1. Пустий словник
A = {}
print('A = ', A)

# 2. Словник з елементів
# 2.1. Містить цілочисельні ключі і рядки значень
B = { 1:'Mon', 2:'Tue', 3:'Wed', 4:'Thu',
      5:'Fri', 6:'Sat', 7:'Sun' }
print('B = ', B)
print('B[2] = ', B[2]) # B[2] = Tue

# 2.2. Містить рядкові ключі і дійсні значення
C = { 'Pi':3.1415, 'Exp':1.71 }
```

```

print('C = ', C)
print('C[Exp] = ', C['Exp']) # C[Exp] = 1.71

# 2.3. Словник, що містить набори значень
D = { 'Table1':[ 1, 2, 4], 'Table2':[ 8, -2, 2.33] }
print('D = ', D)
print('D[Table1] = ', D['Table1']) # D[Table1] = [1, 2, 4]
print('D[Table2] = ', D['Table2']) # D[Table2] = [8, -2, 2.33]

```

Лістинг 1.2 - Результат виконання програми

```

A = {}
B = {1: 'Mon', 2: 'Tue', 3: 'Wed', 4: 'Thu', 5: 'Fri',
6: 'Sat', 7: 'Sun'}
B[2] = Tue
C = {'Pi': 3.1415, 'Exp': 1.71}
C[Exp] = 1.71
D = {'Table1': [1, 2, 4], 'Table2': [8, -2, 2.33]}
D[Table1] = [1, 2, 4]
D[Table2] = [8, -2, 2.33]

```

Словник може бути створений з допомогою одного з конструкторів `dict()`, які реалізовані у класі `dict`. Згідно з документацією Python у класі `dict` використовується 3 конструктори, які мають наступну загальну форму

Лістинг 1.3 - Створення словника з допомогою конструктора `dict()`

```

dict(**keyword_arg)
dict(mapping, **keyword_arg)
dict(iterable, **keyword_arg)

```

де

–`keyword_arg` – необов'язковий ключовий аргумент. Якщо конструктор викликається без ключового аргументу (наприклад, `dict()`), то створюється пустий словник. Якщо в конструкторі задається декілька ключових аргументів, то вони розділяються комою (у загальній формі конструктора використано символи `**`);

–mapping – відображуваний об’єкт, на основі якого будується словник;

–iterable – ітерований об’єкт, на основі якого будується словник.

В залежності від аргументів, інтерпретатор Python викликає відповідний конструктор. У всіх конструкторах перший об’єкт кожного елементу стає ключем, другий стає відповідним значенням.

У випадку, якщо ключ зустрічається більше одного разу, то приймається останнє значення, встановлене за цим ключем.

Лістинг 1.4 - Демонструється створення словників різними способами

```
# Створення словника з допомогою конструктора dict
# 1. Різновид конструктора dict(**keyword_arg)
# 1.1. Створити пустий словник
A = dict()
print('A = ', A) # A = {}

# 1.2. Створити список
# {'Winter': 1, 'Spring': 2, 'Summer': 3, 'Autumn': 4}
SEASONS = dict( Winter=1, Spring=2, Summer=3, Autumn=4)
print('SEASONS = ', SEASONS)

# -----
# 2. Різновид конструктора dict(mapping, **keyword_arg)
# 2.1. Застосування функції zip()
DAYS = [ 1, 2, 3 ]
DAYS_NAMES = [ 'Mon', 'Tue', 'Wed' ]
DICT_DAYS = dict(zip(DAYS, DAYS_NAMES))
print(DICT_DAYS) # {1: 'Mon', 2: 'Tue', 3: 'Wed'}

# 2.2. Застосування пар (key:value)
DICT_DAYS = dict([(1, 'Mon'), (2, 'Tue'), (3, 'Wed')])
print(DICT_DAYS) # {1: 'Mon', 2: 'Tue', 3: 'Wed'}

# -----
# 3. Різновид конструктора dict(iterable,
**keyword_arg)
# Застосування конструктора за зразком B = {}
B = dict({1: 'Mon', 2: 'Tue', 3: 'Wed'})
print('B = ', B) # B = {1: 'Mon', 2: 'Tue', 3: 'Wed'}
```

Лістинг 1.5 - Результат виконання програми

```
A = {}
SEASONS = {'Winter': 1, 'Spring': 2, 'Summer': 3,
            'Autumn': 4}
{1: 'Mon', 2: 'Tue', 3: 'Wed'}
{1: 'Mon', 2: 'Tue', 3: 'Wed'}
B = {1: 'Mon', 2: 'Tue', 3: 'Wed'}
```

1.2 Хід роботи

Виконайте в інтерактивному інтерпретаторі такі завдання:

1. Використайте словник для збереження інформації про видатну особистість (вченого, письменника, винахідника тощо). Збережіть ім'я, прізвище, століття, в якому народилась ця людина, і одне з досягнень. Словник повинен містити ключі з такими іменами, як `first_name`, `last_name`, `century` і `progress`. Виведіть кожен фрагмент інформації, що зберігається у словнику.

2. Створіть словник для зберігання цифр. Візьміть п'ять імен (назви цифр) і використайте їх як ключі словника. Для кожного ключа надайте значення відповідної цифри. Виведіть по черзі назву цифри і її значення для усіх елементів словника.

3. У французько-англійському словнику `f2e` містяться такі слова: `blanc / white`, `rose / pink`, `violet / purple` і `argent / silver`. Опишіть цей словник. Створіть і виведіть на екран множину французьких слів з ключів словника.

4. Створіть множину із списку: `['boat', 'bus', 'plane', 'train']`.

5. Створіть множину із кортежу: `('cyclist', 'driver', 'pedestrian')`.

6. Створіть множину із словника: `{'breakfast': 'coffee', 'lunch': 'milk', 'dinner': 'tea'}`.

Напишіть програми у середовищі програмування для розв'язування таких завдань:

1. Словники Python можуть використовуватися для моделювання «справжнього» словника (назвемо його глосарієм).

Оберіть кілька термінів з програмування (або із іншої області), які ви знаєте на цей момент. Використайте ці слова як ключі глосарію, а їх визначення - як значення. Виведіть кожне слово і його визначення у спеціально відформатованому вигляді. Наприклад, ви можете вивести слово, потім двокрапка і визначення; або ж слово в одному рядку, а його визначення - з відступом в наступному рядку. Використовуйте символ нового рядка (`\n`) для вставки порожніх рядків між парами «слово: визначення» і символ табуляції для встановлення відступів (`\t`) у вихідних даних.

2. Створіть словник з трьома річками і регіонами, територією яких вони протікають. Одна з можливих пар «ключ: значення» - 'Amazon': 'South America'. Додайте ще дві пари «річка: регіон» у словник. Виведіть повідомлення із назвами річки і регіону - наприклад, «The Amazon runs through South America.» для усіх елементів словника, враховуючи те, що у повідомлення у відповідні місця підставляються назви річок і територій.

3. Створіть словник з чотирма назвами мов програмування (ключі) та іменами розробників цих мов (значення). Виведіть по черзі для усіх елементів словника повідомлення типу `My favorite programming language is Python. It was created by Guido van Rossum..` Видаліть, на ваш вибір, одну пару «мова: розробник» із словника. Виведіть словник на екран.

1.3Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

1.4Контрольні питання

1. Як виглядає код для порожнього словника?
2. Як виглядає елемент словника з ключем `key` і значенням 81?
3. Опишіть основні відмінності між словником і списком.

4. У змінній `result` зберігається словник. Яка різниця між такими виразами: `'battery' in result` і `'battery' in result.keys()`?

5. У змінній `result` зберігається словник. Яка різниця між такими виразами: `'battery' in result` і `'battery' in result.values()`?

6. Що буде відображено на місці ... при спробі отримання доступу до елементів, використовуючи такий код:

```
>>> room = {'bookcase': 1, 'armchair': 3, 'table': 1,
'clock': True}
>>> room['armchair']
...
>>> room['mirror']
...
>>> room['clock']
...
>>> room['table']
...
>>> room.get('blinds', 'Blinds are missing.')
...
```

2 ЛАБОРАТОРНА РОБОТА №2

МОДУЛІ І ПАКЕТИ

Мета роботи: отримати навички програмування на мові Python з використанням модулів і пакетів.

2.1 Теоретичні відомості

При написанні великих програм або при збільшенні розміру програми все більше стає потреба у правильному (коректному) розбитті програми на структурно-незалежні частини. Метою такого структурування є:

- полегшення написання чи розвитку програми;
- покращення супроводу програми;
- зменшення кількості логічних помилок, а, отже, підвищення якості розробки програми.

Для забезпечення структурування у мові Python використовуються так звані модулі. Модуль – це програмна одиниця, яка містить програмний код та дані і призначена для організації взаємодії між структурно-незалежними (структурно-унікальними) частинами програми. Кожному модулю відповідає відповідний файл програми. І, навпаки, кожен файл – це окремий модуль.

Будь-який окремий модуль може бути імпортований (підключений, використаний) іншим модулем. Також модуль може імпортувати (використовувати) інший модуль.

Компоненти, що реалізовані в модулі, називаються атрибутами. Компоненти – це імена змінних що зв'язані з функціями, об'єктами, класами тощо.

Також виникає поняття імпорт модуля. Імпорт модуля – це підключення модуля з допомогою спеціальних інструкцій з метою отримання доступу до програмних засобів (інструментів) цього модуля. Після імпортування стають доступними для використання атрибути модуля.

Використання модулів дозволяє структурувати програмні компоненти в автономні пакети, які ще називаються просторами імен. Як в будь-якому просторі імен при імпортуванні модуля всі імена є

атрибутами об'єкту імпортованого модуля. Ці імена описуються в глобальній області видимості імпортованого модуля.

Використання модулів є важливим з наступних причин:

- повторне використання коду. Кожний модуль представлений у вигляді файлу. Це дає змогу підключати його код в різних (декількох) клієнтських частинах програми. У модулі формується набір імен, або, іншими словами, набір атрибутів цього модуля;

- розподіл просторів імен. Модулі є природнім способом групування компонент програмної системи. З допомогою модулів імена ізолюються, що дозволяє уникнути конфлікту та плутанини в іменах. Імена, що використовуються в модулях є згруповані за деяким критерієм. Об'єкти та програмний код є неявно включеними в модулі;

- сумісне використання програмних компонент та даних. Тут мається на увазі, що один і той самий модуль може використовуватись в різних компонентах програмної системи. Модуль тільки один – клієнтських програмних компонент декілька.

Як правило, будь-яка сучасна програма на мові Python побудована так, що вона використовує або складається з декількох файлів, які ще називають модулями. Якщо програма побудована так, що вона містить декілька модулів, то в цій програмі виділяється один головний файл. Цей файл здійснює запуск усієї програми. У цьому файлі, як правило, підключаються інші модулі всієї програми. Головний файл визначає хід виконання основного потоку. У свою чергу, підключені модулі можуть використовувати для своєї роботи засоби інших модулів.

Якщо головний файл не буде імпортований в інші файли, то його розширення може бути будь-яким (не обов'язково розширення *.py). Важливо, щоб це розширення мали всі інші модулі, які підключаються у цьому файлі. У будь-якому випадку, рекомендовано розширення головного файлу задавати як *.py, оскільки це розширення робить призначення даного файлу більш очевидним та в подальшому дозволить його імпортувати в інші файли.

Інші або другорядні файли програми також можуть запускатись на виконання якщо вони активні у вікні редактора Python. Якщо у цих файлах містяться бібліотеки функцій, то цю властивість можна використати для тестування роботи цих функцій у цьому ж модулі. Після завершення тестування важливо не забути прибрати цей код,

оскільки виконавчий код кожного модуля виконується також при запуску коду головного модуля.

2.2 Хід роботи

Виконайте в інтерактивному інтерпретаторі такі завдання:

1. Створіть файл, який називається **city.py**. У ньому оголошіть функцію `library_services()`, яка виводить на екран рядок `Internet access is open 24 hours..` Використайте інтерактивний інтерпретатор, щоб імпортувати модуль **city.py** і викликати його функцію `library_services()`.

2. У інтерактивному інтерпретаторі імпортуйте модуль **city.py** з ім'ям `st` і викличте його функцію `library_services()`.

3. Залишаючись у інтерпретаторі, імпортуйте безпосередньо функцію `library_services()` з модуля **city.py** і викличте її.

4. Імпортуйте функцію `library_services()` з ім'ям `info` і викличте її.

Напишіть програми у середовищі програмування для розв'язування таких завдань:

1. Файл `models.py` містить програмний код, поданий нижче, що імітує друкування 3D-моделей різних об'єктів. Перенесіть функції `print_models()` і `show_completed_models()` у окремий файл з ім'ям `printing_functions.py`. Виконайте імпорт цих функцій у файл `models.py`, змінивши файл так, щоб у ньому імпортовані функції можна було використовувати.

Лістинг 2.1 – Вихідний файл

```
def print_models(unprinted_designs, completed_models):
    """
```

Імітує друк 3D-моделей, доки список не стане порожнім.

Кожна модель після друку переміщується у `completed_models`.

```
    """
    while unprinted_designs:
        current_design = unprinted_designs.pop()
        # Імітація друку моделі на 3D-принтері.
        print("Printing model: " + current_design)
```

```

        completed_models.append(current_design)

def show_completed_models(completed_models):
    """Виводить інформацію про усі надруковані
    моделі."""
    print("\nThe following models have been printed:")
    for completed_model in completed_models:
        print(completed_model)
unprinted_designs = ['iphone case', 'robot pendant',
                    'dodecahedron']
completed_models = []
print_models(unprinted_designs, completed_models)
show_completed_models(completed_models)

```

2. Візьміть за основу одну з написаних вами програм з однією функцією. Збережіть цю функцію в окремому файлі. Імпортуйте функцію у файл основної програми і викличте функцію будь-яким із таких способів:

```

import ім'я_модуля
from ім'я_модуля import ім'я_функції
from ім'я_модуля import ім'я_функції as псевдонім
import ім'я_модуля as псевдонім

```

2.3 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

2.4 Контрольні запитання

1. Що робить інструкція **import**?
2. Яка різниця між модулем і пакетом?
3. Якщо у вас є функція `myfunc()`, яка міститься у модулі `addfunc`, то як би ви її викликали після імпортування цього модуля?
4. Для чого використовується модуль `__name__`?

5. Як отримати доступ у програмі до аргументів командного рядка?
6. Навіщо використовувати віртуальні середовища?

ЛІТЕРАТУРА

1. Python B. K. Python Cookbook / Brian K. Python, David Beazley, Brian K. Jones. – [S. l.] : O'Reilly Media, Incorporated, 2012. - 704 p.
2. Мова програмування Python для інженерів і науковців: Навчальний посібник. – Ivano-Frankivsk, Ukraine : ІФНТУНГ, 2019. – 275 с.
3. Barry P. Head First Python / Paul Barry. – [S. l.] : O'Reilly Media, Incorporated, 2015. – 624 с.