УДК 376:004.43
DOI:

Тарас Кобильник, *кандидат педагогічних наук, доцент,*
доцент кафедри інформаційних систем і технологій
Національного університету "Львівська політехніка"

МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ ВБУДОВАНИХ СТРУКТУР ДАНИХ МОВИ PYTHON

У статті розглядаються деякі проблеми навчання основ алгоритмізації та програмування у шкільному курсі інформатики для учнів старших класів з поглибленим вивченням інформатики. Зокрема, нами звертається увага на вивчення вбудованих структур даних мови Python. Аналізуються методичні особливості навчання учнів таких вбудованих типів даних мови Python, як списки, словники, кортежі, множини. Перспективним напрямом майбутніх розвідок вбачаємо розробку навчально-методичних матеріалів з основ алгоритмізації та програмування з використанням мови Python.

Ключові слова: *основи алгоритмізації та програмування; мова Python; вбудовані структури даних.*
Літ. 7.

Taras Kobylnyk, *Ph.D. (Pedagogy), Associate Professor,*
Associate Professor of the Information Systems and Technologies Department,
Lviv Polytechnic National University

METHODOLOGICAL FEATURES OF TEACHING BUILT-IN DATA STRUCTURES OF THE PYTHON LANGUAGE

In the article, we consider some problems of learning the basics of algorithmization and programming in a school computer science course for high school students with an in-depth study of computer science. In particular, we pay attention to the study of the built-in data structures of the Python language. We analyze the methodological features of teaching students such as built-in Python data types such as lists, dictionaries, tuples, and sets. For each built-in data structure, we provide examples of its use and explain the obtained result in detail. We focus on list assignment operations and the use of slices. The modern level of programming involves using various data structures as a necessary tool in constructing software code. Without an understanding of data structures and algorithms, creating a serious software product is impossible. Therefore, the realization of the learning goal involves students' understanding of various data structures, their presentation methods, and processing methods. The study of data structures is fundamental in the formation of future programmers. Data structures are an integral part of programming. Understanding the principles of algorithms and data structures during software development makes it possible to improve program performance, improve code quality, and speed up its work. We see prospects for further research in developing educational and methodological materials for learning the basics of algorithmization and programming using the Python language, taking into account its features, particularly dynamic typing and too "big" high-levelness. We consider it expedient to analyze in more detail the methodological aspects of the study of other data structures, such as stacks, queues, hash tables, trees, and graphs in Python and C++ languages.

Keywords: *basics of algorithmization and programming; Python language; built-in data structures.*

Постановка проблема. Сьогодні спостерігається тенденція з впровадженням мови Python як засобу навчання змістової лінії алгоритмізації та програмування у шкільному курсі інформатики у 7–9-х класах та як основної мови програмування у 10–11-х класах, які вивчають інформатику на профільному рівні. Зауважимо, що навчальними програмами з інформатики не визначено якоїсь конкретної мови програмування. Вибір залишається за вчителем чи закладом освіти і залежить від наявності навчально-методичного та матеріально-технічного забезпечення.

У 10–11-х класах з поглибленим вивченням інформатики увага акцентується вже не на вивченні алгоритмів, а власне на певній мові програмування. Вивчення будь-якої мови програмування починається з ознайомленням зі синтаксисом та типами даних. Програмою з інформатики (профільний рівень) [1] передбачається вивчення різних структур даних, способи реалізації та застосування їх на практиці.

Аналіз останніх досліджень та публікацій. На важливість вміння програмувати вказують автори статті [5], у якій зазначають, що це сприяє розумовому розвитку учнів, формуванню вміння концентруватися на розв'язанні поставленої проблеми, становленню алгоритмічного стилю мислення та умінню моделювати різні процеси.

У статті [2] автори аналізують деякі проблеми навчання змістової лінії основи алгоритмізації та програмування у шкільному курсі інформатики, зокрема увага акцентується на виборі мови програмування і наявності навчально-методичних матеріалів.

Вивчення структур даних є фундаментальним у становленні майбутніх програмістів, оскільки, як вважають автори статті [3], розуміння принципів

роботи алгоритмів і структур даних під час розроблення програмного забезпечення дає змогу поліпшити продуктивність програм, якість коду і прискорити його роботу.

Проєктування та аналіз ефективних структур даних є надзвичайно важливим інструментом у програмуванні, і відповідно знаходять відображення у навчальних програмах з інформатики. Автори [6] зазначають, що побудова структур даних і алгоритмів вимагає, щоб програміст передав комп'ютеру докладні інструкції. Для здійснення такого зв'язку пропонується використання мови Python.

Також мову Python для навчання алгоритмів та структур даних пропонують використовувати і автори книги [7], перші розділи якої присвячені основам структур даних і алгоритмів, наступні – охоплюють складніші розділи з цієї теми.

Мета статті: методичні аспекти навчання структур даних на основі мови Python для учнів старших класів.

Виклад основного матеріалу. Інформатику у старшій школі вивчають на рівні стандарту та профільному. На рівні стандарту основи програмування можливі тільки у вибірковому модулі "Креативне програмування". На профільному рівні навчальна програма містить кілька розділів, які пов'язані з програмування, зокрема, "Мова програмування та структури даних", "Алгоритми", "Парадигми та технології програмування".

Зупинимось детальніше на розділі "Мова програмування та структури даних", зокрема, на методичних аспектах навчання учнів вбудованих структур даних та їх реалізації мовою Python. У підручнику з інформатики [4] для профільного рівня описуються такі вбудовані структури даних, як списки, словники, кортежі, множини. Проаналізуємо деякі з них.

У кожній мові програмування передбачені структури даних. У мові Python є такі базові структури даних, як список (list), кортеж (tuple), словник (dictionary), множина (set). Кожна з них має свої особливості. Зауважимо, що структури даних становлять невід'ємну складову програмування.

Важливо зауважити, що списки – це найпоширеніша структура даних мови Python. Під списком розуміють набір будь-яких об'єктів, які поміщені у квадратні дужки і розділені комою. Списки можуть бути вкладені, з ними можна виконувати різні дії, зокрема, генерувати список, видаляти елемент з нього, вставляти елемент, сортувати тощо. Для цього використовуються відповідні методи та функції. Тут доцільно учням пояснити відмінність між методами та функціями опрацювання списків. У першому випадку вихідний список змінюється, у другому – ні. Продемонструємо це на такому прикладі.

Списки є вбудованими за замовчуванням у мову програмування Python, а масиви – ні. Для створення та опрацювання масивів необхідно імпортувати модуль array. Крім того, треба звернути увагу на те, що масив – це набір однотипних елементів (на відміну від списку), тобто масив містить елементи одного типу.

Кортежі. Кортежі використовують для зберігання кількох об'єктів разом. Їх розглядають як аналог списків, але без такої значної функціональності, як у списків. Однією з важливих особливостей кортежів є те, що вони, нам відміну від списку, – незмінювана структура даних, тобто модифікувати їх не можна. Кортежі створюються аналогічно до списків (замість квадратних дужок використовуються круглі). Наприклад:

```
tup=(1,2.3,5,58)
```

Важливо зауважити, що дужки можна і не ставити:

```
tup=1,2.3,5,58
```

Круглі дужки є необов'язковими, хоча рекомендується їх використовувати для чіткого визначення початку та кінця кортежу.

Кортеж, який містить тільки один елемент, задається так:

```
tup=(2,)
```

Тобто після значення єдиного його елемента ставиться кома. Таким чином, "повідомляється" Python, що створюється кортеж.

Суттєвою перевагою кортежу є те, що він може бути присвоєний відразу кільком змінним, які набудуть значення, що містяться в його елементах. Наприклад:

```
a1,a2,a3,a4=tup
```

У цьому випадку змінна a1 набуде значення першого елемента кортежу tup, змінна a2 – значення його другого елемента і т.д.

Розглянемо приклад.

```
boys = ('Іван', 'Василь', 'Степан')
girls = 'Олена', 'Софія', 'Анастасія', 'Анна'
clas = boys, girls
print('Кількість хлопців у класі -', len(boys))
print('Кількість дівчат у класі:', len(girls))
print('Всі учні класу:', boys,girls)
print('Кількість учнів у класі -', len(boys)+len(girls))
```

Результат

```
Кількість хлопців у класі - 3
Кількість дівчат у класі: 4
Всі учні класу: ('Іван', 'Василь', 'Степан') ('Олена', 'Софія', 'Анастасія', 'Анна')
Кількість учнів у класі - 7
```

Змінні boys та girls – це кортеж елементів. За функцією len визначається довжина кортежу. Це вказує на те, що кортеж є послідовністю. Хлопці і дівчата навчаються у класі, тому кортеж clas містить імена хлопців і дівчат.

Хоча дужки є не обов'язковими, рекомендуємо завжди вказувати їх, щоб було видно, що це дійсно кортеж, особливо в неоднозначних випадках. Наприклад, print (2,3,4,5) та print ((2,3,4,5)). У першо-

му випадку виводяться чотири числа, а другому – кортеж, який містять ці чотири числа.

Словники. Словники – це особливий список, елементи якого мають індекси, що задані програмістом. Індексами можуть бути тільки незмінювані об'єкти, тобто числа, рядки та кортежі. Проте на практиці, як правило, найчастіше використовують рядки. Важливо зауважити, що у такий спосіб задані індекси називають *ключами*.

Увести поняття "словник" можна так. Словник – це певний аналог адресної книги, у якій можна відшукати адресу або контактні відомості про людину, знаючи тільки її ім'я. Тобто деякі **ключі** (імена) певним чином пов'язані зі **значеннями** (відомостями). Тут треба зауважити, що ключ повинен бути унікальним, оскільки не можна отримати коректні відомості, якщо в адресній книзі містяться дві (або більше) людини з абсолютно однаковими іменами.

Словники створюються аналогічно до списків, за винятком таких моментів. По-перше, список елементів поміщений у фігурні дужки (а не в квадратні). По-друге, елементи записують у форматі <ключ>: <значення>. Наприклад:

```
d1={"name":"Python","version":"3.10.5"}
d2={"name":"Visual Studio","version":"2022"}
```

Так було створено два словники, кожен з яких має два елементи з рядковими ключами "name" та "version".

Доступ до будь-якого елемента словника або його зміна здійснюється за його ключем, наприклад:

```
vesr=d2["version"]
d2["version"]= "2019"
```

Словники, як і звичайні списки, можна змінювати. Як правило, словники використовуються для зберігання даних з однієї області. Також важливо знати, що пари ключ-значення є не впорядкованими у словнику. Якщо потрібна певна впорядкованість, то додатково необхідно буде відсортувати словник перед зверненням до нього (або до його елементів).

Розглянемо такий приклад.

```
address_book = { 'Іваненко' : 'ivanenko@ukr.net',
                 'Петренко' : 'petrenko@meta.ua',
                 'Василенко' : 'vasylenko@i.ua',
                 'Павленко' : 'pavlenko@gmail.com'
                }
print("e-mail Іваненка: ", address_book['Іваненко'])
# Видалення пари ключ-значеннння
del address_book['Іваненко']
print('\nУ адресній книзі {0}
контакти(ів)\n'.format(len(address_book)))
for name, address in address_book.items():
   print('Контакт {0} має e-mail: {1}'.format(name,
address))
# Додавання пари ключ-значення
address_book['Семеренко'] =
'semerenko@yahoo.com'
if 'Семеренко' in address_book:
   print("\nАдреса Семеренко:",
address_book['Семеренко'])
```

Результат буде таким:

```
e-mail Іваненка: ivanenko@ukr.net

У адресній книзі 3 контакти(ів)

Контакт Петренко має e-mail: petrenko@meta.ua
Контакт Василенко має e-mail: vasylenko@i.ua
Контакт Павленко має e-mail: pavlenko@gmail.com

Адреса Семеренко: semerenko@yahoo.com
```

Пояснимо отриманий результат. Створюється словник address_book. Потім звертається до пар ключ-значення, вказуючи ключ в операторі індексування (аналогічно як для списків та кортежів). Видаляти пари ключ-значення можна за допомогою оператора del. Для цієї операції нема необхідності знати, яке значення відповідає цьому ключу, тобто видалення елемента словника відбувається за ключем. Далі здійснюється звернення до всіх пар ключ-значення словника address_book з використанням методу items, за яким повертається список кортежів, кожен з яких містить пару елементів: ключ і значення. У циклі for..in відповідно їм присвоюється значення змінних name та address. Ці значення виводяться на екран.

Нові пари ключ-значення додаються так: потрібно звернутися до необхідного ключа за допомогою оператора індексування і присвоїти йому певне значення (як було зроблено у прикладі для Семеренко). Перевірка існування пари ключ здійснюється за допомогою оператора in.

Послідовності. Списки, кортежі та рядки є прикладами послідовностей. Що у них є спільного і важливого? По-перше, це можливість здійснювати перевірку належності (тобто оператори in та not in) та оператор індексування, за допомогою якого отримується безпосередній доступ до елемента послідовності. Крім того, для цих типів послідовностей (списки, кортежі та рядки) є можливість використовувати зрізи, тобто отримувати не один елемент, а кілька.

Множини. Під множиною розуміють невпорядкований набір простих об'єктів. Такий тип структури даних використовується тоді, коли наявність об'єкта в наборі є важливішого від того, скільки разів цей об'єкт там зустрічається і в якому місці. Використовуючи множини, можна здійснювати перевірку належності, визначати, чи дана множина є підмножиною іншої множини, знаходити перетин, об'єднання множин тощо.

Наведемо деякі приклади.

```
>>>states = set(['Україна', 'США', 'Польща',
'Німеччина'])
```

```
>>>‘Польща’ in states
    True
>>>‘Франція’ in states
    False
>>>states1 = states.copy()
>>>states1.issuperset(states)
    True
>>>states.remove(‘США’)
>>>states & states1 # Або bri.intersection(bric)
{‘Польща’, ‘Україна’, ‘Німеччина’}
```

Присвоювання списків. Посилання.

Доцільно зупинитися на присвоюванні списків. Коли створюється об’єкт і його присвоюється змінній, то змінна тільки посилається на об’єкт, а не є власне тим об’єктом. Тобто ім’я змінної вказує на ту частину пам’яті комп’ютера, де зберігається об’єкт.

Наприклад, створимо такий список:

```
lst1=[1,2,3,4]
```

і присвоїмо його іншій змінній:

```
lst2=lst1
```

Тепер змінимо значення першого елемента списку lst1 на інше:

```
lst1[0]=12
```

і перевіримо, що які значення елементів будуть у списку lst2:

```
lst2
    [12, 2, 3, 4]
```

Бачимо, що значення першого елемента списку lst2 також змінилося на 12. Отже, у змінних lst2 та lst1 зберігається один і той самий список. Пояснимо це детальніше. Коли створюється список і присвоюється його змінній, остання як значення отримує не власне список, а посилання на нього, яке можна вважати вказівником на певну ділянку оперативної пам’яті, де він міститься. Коли присвоюється значення змінної, де міститься список, іншій змінній, то туди копіюється посилання. Як наслідок, обидві змінні фактично зберігають один і той самий список (посилання на одну і ту ж ділянку оперативної пам’яті).

Те саме стосується і чисел, рядків та логічних величин. У випадку їх присвоювання іншій змінній у неї копіюється не сама величина, а тільки посилання на неї.

Зауважимо, що якщо необхідно зробити копію списку (або подібної послідовності), то потрібно використати зріз. Оскільки якщо просто присвоїти ім’я змінної іншій змінній, то обидві вони буди посилатися на один і той самий об’єкт, що може призвести до проблем.

Висновки та перспективи подальших досліджень. Сучасний рівень програмування передбачає застосування різних структур даних як необхідного інструменту у побудові програмного коду. Відповідно знання теоретичних питань структур даних, практичних навичок їх подання та опрацювання є важливим елементом у вивченні розділу “Мова програмування та структури даних”. У статті наведені методичні аспекти навчання вбудованих структур даних мови Python, зокрема списки, кортежі, словники, множини.

Без розуміння структур даних і алгоритмів неможливо створити серйозний програмний продукт. Тому реалізація мети навчання передбачає розуміння учнями різноманітних структур даних, способів їх подання та методів опрацювання.

Перспективи подальших досліджень вбачаємо у розробці навчально-методичних матеріалів для навчання основ алгоритмізації та програмування з використанням мови Python з врахуванням її особливостей, зокрема динамічної типізації та надто великої високорівневості. Детальніше зупинитися на методичних аспектах навчання інших структур даних, таких як стек, черга, хеш-таблиці, дерева, графи.

ЛІТЕРАТУРА

1. Інформатика для 10–11 класів (профільне навчання). URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx (дата звернення 12.05.2022).

2. Кобильник Т., Когут У., Сікора О., Жидик В. Деякі проблемні аспекти навчання основ алгоритмізації та програмування у школі. *Молодь і ринок*. 2022, № 3–4 (201–202). С.97–101.

3. Прокоп Ю.В., Трофименко О.Г., Дикий О.В. *Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки.* 2021, Том 32 (71) Ч.1 №2. С. 216–220.

4. Руденко В.Д., Речич Н.В., Потієнко В.О. Інформатика (профільний рівень) : підруч. для 10 кл. закл. загал. серед. освіти. Харків, 2019. 256 с.

5. Семеніхіна О.В., Руденко Ю.О. Проблеми навчання програмувати учнів старших класів та шляхи їх подолання. *Інформаційні технології і засоби навчання.* 2018, Вип. 66(4). С. 54–64.

6. Goodrich M., Tamassia R., Goldwasser M. Data Structures and Algorithms in Python. Wiley. 2013. 748 p.

7. Kent L., Hubbard S. Data Structures and Algorithms with Python. Springer Cham. 2015. 364 p.

REFERENCES

1. Informatyka dlia 10–11 klasiv (profilne navchannia) [Informatics for 10–11 grades (profile training)]. Available at: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx (Accessed 12 May 2022). [in Ukrainian].

2. Kobylnyk, T., Kohut, U., Sikora, O. & Zhydyk V. (2022). Deiaki problemni aspekty navchannia osnov alhorytmizatsii ta prohramuvannia u shkoli [Some Problemal Aspects of Teaching the Fundamentals of Algorithmization and Programming in School]. “*Youth and market”.* No. 3–4 (201–202). pp.97–101. [in Ukrainian].

3. Prokop, Yu.V., Trofymenko, O.H. & Dykyi, O.V. (2021). [Research of Approaches to Teaching the Course “Algorithms and Data Structures” for Computer Science Students]. *Academic notes of Tavriya National University Vernadsky V.I. Series: Technical sciences.* Vol. 32 (71), Part 1, No 2. pp. 216–220. [in Ukrainian].

4. Rudenko, V.D., Rechich, N.V. & Potienko, V.O. (2019). Informatics (profile level): a textbook for the 10th grade of

general secondary education institutions. Kharkiv: Ranok Publishing House. 256 p. [in Ukrainian].

5. Semenykhina, O.V. & Rudenko, Y.O. (2018). Problems of Educating to Programming of Students and Way of Their Overcoming. Information Technologies and Learning Tools. No. 66(4). pp. 54–64. [in Ukrainian].

6. Goodrich, M., Tamassia, R. & Goldwasser M. (2013). Data Structures and Algorithms in Python. Wiley. 748 p. [in English].

7. Kent, L. & Hubbard, S. (2015). Data Structures and Algorithms with Python. Springer Cham. 364 p. [in English].

Стаття надійшла до редакції 03.08.2022

УДК 378.14
DOI:

Федір Загура, *кандидат наук з фізичного виховання і спорту,*
завідувач кафедри атлетичних видів спорту
Львівського державного університету фізичної культури імені Івана Боберського

ПРОФЕСІЙНА ДІЯЛЬНІСТЬ ВИКЛАДАЧІВ ФІЗИЧНОГО ВИХОВАННЯ У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ: НОРМАТИВНО-ПРАВОВИЙ ДИСКУРС

У статті висвітлено сутність та особливості професійної діяльності викладачів фізичного виховання у закладах вищої освіти крізь призму нормативно-правових документів. Виконано аналіз науково-педагогічної літератури з проблеми дослідження. Представлено результати контент-аналізу нормативних документів України, що декларують вимоги до компетентності викладача сучасного університету. Представлено основні напрями роботи науково-педагогічних працівників закладів вищої освіти: науковий, навчальний, методичний, організаційний. Виконано аналіз нормативних документів щодо професійної підготовки студентів у закладах вищої освіти та зроблено висновок про те, що вони регламентують оздоровчо-профілактичну, спортивно-дозвіллєву та спортивну роботу, що належать до компетенції викладачів фізичного виховання.

Ключові слова: *науково-педагогічний працівник; університет; заклад вищої освіти; напрями професійної діяльності; викладач фізичного виховання.*

Літ. 14.

Fedir Zahura, *Ph.D. (Physical Training and Sports),*
Head of the Athletic Sports Department
Lviv Ivan Boberskyi State University of Physical Culture

PROFESSIONAL ACTIVITY OF PHYSICAL TRAINING TEACHERS IN HIGHER EDUCATION INSTITUTIONS: REGULATORY AND LEGAL DISCOURSE

The article highlights the essence and peculiarities of the professional activity of physical training teachers in institutions of higher education through the prism of regulatory and legal documents. An analysis of the scientific and pedagogical literature on the research problem was performed. Various aspects of the professional activity of scientific and pedagogical employees are studied. In particular, the issues of working hours accounting and legal regulation of university teachers' activity, their rights and duties, the professionalism of scientific and pedagogical employees and the specifics of its development, the peculiarities of evaluating the professional activity of university teachers, etc. are studied. However, such studies are considered from the perspective of public administration in the field of education and jurisprudence. Therefore, this question needs to be considered in the context of modern pedagogical research. The results of the content analysis of normative documents of Ukraine, which declare the competence requirements (knowledge and understanding of the subject area and professional activity, critical thinking skills, communication skills, empathy, ICT competence, personal and professional development skills, professional ethics, multicultural competence, social responsibility, etc.) for modern university teacher are presented.

The main areas of scientific and pedagogical employees' activity at higher education institutions are presented. Among them we distinguish scientific, educational, methodical, organizational types of activity. The analysis of regulatory documents regarding the professional training of students in higher education institutions was carried out and the conclusion was made that they regulate health-prophylactic, sports-leisure and sports activity that belong to the competence of physical training teachers. Prospects for further scientific research are offered.

Keywords: *scientific and pedagogical employee; university; institution of higher education; areas of professional activity; teacher of physical training.*

Постановка проблеми. Нині університет відіграє ключову роль у розвитку суспільства, оскільки виконує низку функцій, серед яких освітня, наукова, інноваційна, соціальна, економічна, культурологічна тощо. Відомо, що якість підготовки фахівців залежить значною мірою від професійної діяльності науково-педагогічних працівників, їхньої освіченості, професійної майстерності тощо. Відтак видається логічним вивчення специфіки їхньої діяльності, що реалізується