УДК 376:004.43

**Кобильник Тарас Петрович**
кандидат педагогічних наук, доцент, доцент кафедри інформатики та інформаційних систем
Дрогобицький державний педагогічний університет імені Івана Франка, м. Дрогобич, Україна
ORCID ID 0000-0002-2703-7570
*kobylnyktaras@gmail.com*

**Сікора Оксана Володимирівна**
кандидат технічних наук, доцент, завідувачка кафедри інформатики та інформаційних систем
Дрогобицький державний педагогічний університет імені Івана Франка, м. Дрогобич, Україна
ORCID ID 0000-0002-4043-778X
*sikora60@ukr.net*

**Жидик Володимир Богданович**
старший викладач кафедри інформатики та інформаційних систем
Дрогобицький державний педагогічний університет імені Івана Франка, м. Дрогобич, Україна
ORCID ID 0000-0002-9876-0053
*Zhvb63@gmail.com*

**Шаран Олександра Василівна**
кандидат педагогічних наук, доцент, доцентка кафедри математики, інформатики та методики їх викладання у початковій школі
Дрогобицький державний педагогічний університет імені Івана Франка, м. Дрогобич, Україна
ORCID ID 0000-0003-3198-8026
*sharan_oleks@ukr.net*

# PYTHON ЯК ЗАСІБ НАВЧАННЯ ОСНОВ АЛГОРИТМІЗАЦІЇ У ЗАКЛАДАХ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ

**Анотація.** На сьогодні спостерігається тенденція до поширення програмування, а мови програмування стають більш простими та зручними. Як наслідок кількість програмістів буде збільшуватися. Водночас рівень глибокого розуміння основ алгоритмізації буде знижуватися. Це неминуче, але не є причиною відмови від вивчення сучасних високорівневих мов програмування. Проблема вибору мови програмування для навчання основ алгоритмізації у закладах загальної середньої освіти є актуальною і складною. На основі аналізу наукових публікацій та власного досвіду визначено критерії вибору мови програмування як засобу навчання основ алгоритмізації: сучасність, безкоштовність, зрозумілість, лаконічність. Обґрунтовано вибір мови Python як засобу навчання основ алгоритмізації учнів та проаналізовано певні її характеристики, які ілюструються конкретними прикладами. Для кращого розуміння особливостей мови Python програмна реалізація алгоритмів наводиться паралельно з мовою C++. Проте вивчення учнями Python як першої мови програмування може викликати деякі побоювання. Це – динамічна типізація змінної і «надто велика» високорівневість мови. Основною проблемою для вивчення мови програмування Python у школах є відсутність розроблених методик її навчання, на відміну від Pascal. Потребує уточнення, які теми, алгоритми потрібно вивчати у школах та як сформувати послідовність тем та обрати відповідну методику навчання. Краще зосередитись на навчанні алгоритмів, формуванні навичок розв'язування різноманітних задач, які вони в майбутньому зможуть використовувати у практичній діяльності. Тому треба основний акцент робити не на навчанні певної мови програмування, а на навчанні основ алгоритмізації, а мову програмування використовувати як допоміжний засіб. Мову програмування доцільно обирати ту, яку легко читати і якою легко писати. Це допоможе підвищити рівень впевненості учнів, які мають не надто добрі знання з основ алгоритмізації і які в майбутньому не пов'язують свою професійну діяльність з програмуванням. Подальші дослідження будуть спрямовані на розробленні методики навчання основ алгоритмізації та програмування з використанням мови Python у класах інформатичного профілю.

**Ключові слова:** мова Python; заклад загальної середньої освіти; основи алгоритмізації, програмування.

 

## 1. ВСТУП

**Постановка проблеми.** Однією зі змістових ліній у шкільному курсі інформатики є основи алгоритмізації та програмування. Зауважимо, що основи алгоритмізації та програмування є частиною навчальної дисципліни «Інформатика». Відповідно постає питання, як навчати цьому учнів? Навчальними програмами з інформатики не передбачається вивчення конкретної мови програмування, основний акцент робиться на алгоритмізації. Завдання, які ставляться, полягають у формуванні та розвитку логічного та алгоритмічного стилю мислення. Вибір мови програмування для навчання основ алгоритмізації залежить від учителя чи закладу освіти. Зауважимо, що основи алгоритмізації та програмування школярі починають вивчати ще з початкових класів. Тут постає цілком слушне питання: яку мову програмування обрати, щоб не знизити мотивацію до навчання протягом тривалого часу. Для навчання основ алгоритмізації і програмування використовують різні мови: від Pascal до Java та C#.

Ще у 2013 році редакція журналу «Комп'ютер у школі та сім'ї» звернулась до відомих учителів інформатики, фахівців у галузі навчальної інформатики з проханням взяти участь в обговоренні питання «Яку мову програмування вивчати у школі?». Таке обговорення вилилось у кілька публікацій [1], [2] у яких знані педагоги (та й не тільки вони) висловлювали думки з приводу цього питання. У кожного з них є своя обґрунтована позиція щодо вивчення мови програмування у школі: Pascal, C/C++, Visual Basic, Visual Basic For Application, JavaScript, Python тощо. Підсумовуючи, можна зробити висновок, що всі аргументи на користь тієї чи іншої мови програмування є важливими, але останнє слово завжди залишається за вчителем.

**Аналіз останніх досліджень і публікацій.** У дослідженні [3] автори зазначають, що учнів необхідно навчати концепції програмування, абстрагуючись від конкретної мови програмування. Вони розглядають мову програмування як допоміжний інструмент у навчанні основ алгоритмізації і програмування.

Як зазначено у [4], для навчання програмуванню використовують два основних типи мов. На думку авторів, до першої належать, наприклад, мови Scratch та App Inventor, у яких програму можна створювати шляхом маніпулювання графічними об'єктами. До другого належать мови, наприклад, Python і Logo, у яких програми створюються шляхом введення тексту. Автори зазначають, що мова Python пропонує простий спосіб вивчення програмування з використанням середовище RUR-PLE [5].

Порівнюючи C++ та Python, автори статті [6] роблять висновок, що все частіше обирається Python як перша мова для навчання програмування. Це пояснюється такими причинами: простий синтаксис, легке для вивчення середовище і висока абстракція в порівнянні з C++.

Зазначається, що Python поступово стає першою мовою для вивчення [7]. У цій праці автори досліджують особливості вивчення першої мови програмування на основі порівняння Java та Python. Зокрема увага звертається на такі особливості, як-от: виконання програми (інтерпретація чи компіляція), оголошення змінних (динамічне чи статичне), синтаксис.

Дослідження [8] також присвячене порівнянню мов Java та Python (для навчання майбутніх інженерів). На думку цих авторів, мова Python значно спрощує введення базових концепцій через простий синтаксис. На основі власних спостережень для навчання студентів коледжу автори надають перевагу Python перед Java. Однією з причин цього називають те, що Python стає все більш розповсюдженою. Крім цього, у старших класах все частіше для вивчення основ програмування обирають Python.

У назві статті [9] E. Mészárosová намагається відповісти на питання: чи доцільно вивчати Python як першу мову програмування у середній школі, чи підходить вона для

всіх учнів середньої школи? Авторка на основі аналізу думок словацьких учителів та експертів робить висновок, що необхідно зосередитись не на вивченні певної мови програмування, а на навчанні основ програмування. Мова, яка найкраще підходить для реалізації даної цілі, не повинна мати складні синтаксис і середовище розробки. Формулюючи критерії вибору першої мови програмування для навчання учнів у школі, авторка висловлює думку, що мова Python якнайкраще підходить для цього. Такої ж думки дотримується й автор статті [10].

На основі досвіду навчання програмуванню учнів середньої школи автори рекомендують використовувати для написання програмного коду мову, якою легко писати і читати, таку як Python [11].

Важливими є критерії, за якими обирається та чи інша мова програмування як засіб навчання основ алгоритмізації. Шевчук П.Г. [12] наводить критерії та передумови добору мов програмування та середовищ розробки. Автор наголошує на важливості методичного забезпечення навчання основ алгоритмізації і програмування. У статті автором здійснено порівняння мов програмування C# та Pascal щодо доцільності їх використання як засобів навчання.

Тісно пов'язаним з вибором мови програмування є вибір середовища програмування. Так, Базурін В.М. [13] розкриває умови вибору середовища програмування як засобу навчання учнів програмуванню сучасними мовами, визначає основні умови, які впливають на вибір середовища програмування та аналізує основні характеристики найбільш поширених середовищ програмування мовами C/C++, C#, Java.

Обізнаність з основ алгоритмізації та програмування сприяє розумовому розвитку учнів, формуванню вміння концентруватися на розв'язуванні поставленої задачі, розвитку логічного та алгоритмічного стилю мислення [14].

Аналізуючи проблеми сучасного стану шкільної інформатики, автори, звертають увагу і на таку змістову лінію, як основи алгоритмізації та програмування, зокрема на вибір першої мови програмування [15]. На їх думку, необхідно впроваджувати такі мови програмування, які застосовуються на підприємствах ІТ-галузі. Серед цих мов (C++, Java, Python) вважають за доцільне вивчати C++ як першу мову програмування, оскільки синтаксис багатьох мов є С-подібним.

На основі порівняння мов Python і Free Pascal автори роблять висновки, що вивчення Python як першої мови програмування має певні перспективи [16]. У статті зазначається, що програмний код, написаний на Python, значно коротший, ніж написаний на Free Pascal чи С-подібних мовах, а тому є зрозумілішим для учнів. Поряд з цим зазначають, що значні проблеми у вивченні Python викликає відсутність візуального середовища розробки програм. Зокрема у середовищах, описаних у підручниках з інформатики, неможливо створити екранну форму в режимі конструктора.

У [2, с. 18] Пасіхов Ю.Я. та Кравець Г.П. наводять деякі переваги мови Python над іншими (C, C++, Pascal), серед яких слід відзначають такі. Прості програми записуються в кілька рядків, відсутні інструкції, що не стосуються безпосередньо алгоритму (наприклад, int main() у мові С чи С++). Як правило, програми мовою Python є коротшими, ніж на С, С++, Pascal, та, тим більше, C#. Мова сучасна, підтримує високорівневі складові структури даних (списки, множини, словники тощо). Мова Python із самого початку створювалась на основі парадигми об'єктно орієнтованого програмування, але чудово пристосована для структурного і функціонального програмування.

Серед вітчизняних праць відзначимо посібник [17], зміст якого відповідає навчальній програмі з інформатики (профільний рівень) для учнів 10-11 класів.

**Метою статті** є обґрунтування вибору мови Python як засобу навчання учнів основ алгоритмізації та аналіз певних її особливостей.

## 2. МЕТОДИ ДОСЛІДЖЕННЯ

Для досягнення поставленої мети було використано такі методи: аналіз навчальних програм з інформатики для закладів загальної середньої освіти; аналіз шкільних підручників з інформатики для 5-11 класів; аналіз вітчизняного та зарубіжного досвіду навчання основ алгоритмізації і програмування; узагальнення та систематизація власного досвіду навчання основ алгоритмізації і програмування.

## 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Дистрибутив Python можна завантажити за посиланням https://www.python.org/downloads/. До його складу входить також інтегроване середовище IDLE. У мові Python передбачено два способи написання програм:

1) режим командного рядка: відображення результатів опрацювання кожної функції;
2) режим запуску файлу: виконання файлу з вихідним текстом програми в цілому.

Режим командного рядка зручно використовувати, коли необхідно швидко перевірити невеликий фрагмент коду або як калькулятор з великим набором функцій. В інших випадках доцільно використовувати режим запуску файлу. Крім консольного інтерпретатора мови, Python має кілька різних середовищ розробки програм, зокрема IDLE (стандартне середовище, що налкжить до дистрибуту, є простим і зрозумілим).

Відповідно є вже розроблені шкільні підручники [18], у яких описано мову Python, зокрема версія Python 3.4, де вивчення основ алгоритмізації і програмування пропонується здійснювати на основі середовища програмування IDLE.

У старших класах учні більш пристосовані до засвоєння складних понять, добре володіють комп'ютером. Мова програмування Python якнайкраще може бути використана для вивчення інформатики в 10-11 класах профільного навчання.

Аналізуючи шкільні підручники з інформатики для 5-9 класів, бачимо, що мову Python як правило, починають вивчати з 7-го класу (за винятком окремих підручників для 5-го класу). Протягом навчання у 5-9-х класах учні набувають початкових знань з алгоритмізації та програмування. Можливо, це і буде межа, де школярі зупиняться на вивченні основ алгоритмізації та програмування, ознайомившись з поняттям «алгоритм», базовими алгоритмічними структурами, деякими структурами даних та їх реалізацією на мові Python.

У 10-11 класах вивчення інформатики проводиться на двох рівнях: рівень стандарту та профільний рівень. На рівні стандарту в старшій школі вивчення програмування передбачено у вибірковому модулі «Креативне програмування» [19]. Тож не всі учні будуть вивчати цей модуль. Це й зрозуміло: не всі бажають і можуть бути програмістами. Програмою з інформатики (профільний рівень) для учнів 10-11 класів [20] передбачено вивчення таких розділів, як-от «Мова програмування та структури даних», «Алгоритми» та «Парадигми програмування», які так чи інакше передбачають опанування певною мовою програмування. Слід зазначити, що вивчення основ програмування, мови, синтаксису, структури та правил написання програм передбачено в розділі «Мова програмування та структури даних». У підручниках з інформатики (профільний рівень) для 10-11 класах для вивчення таких питань використовується мова Python.

Під час вивчення розділу «Мова програмування та структури даних» здійснюється систематизація та узагальнення знань, отриманих з основ алгоритмізації

та програмування в 5-9 класах. Зокрема це стосується таких питань, як базові алгоритмічні структури та структури даних.

З алгоритмізацією та програмуванням у школі учні починають знайомитися ще в початкових класах. Аналіз підручників з інформатики (2016 року) показав, що в 5-6 класах учні опановують Scratch, а у 7-9 – класах ознайомлюються з середовищем Lazarus (Free Pascal) та мовою Python.

Щоб обрати певну мову програмування, треба визначити критерії, яким повинна вона відповідати.

Обрана мова для вивчення програмування в школі повинна відповідати таким вимогам [2, с.17-18]:

- транслятор мови повинен бути кросплатформенним і безкоштовним;
- мова повинна мати простий синтаксис, з одного боку, і потужні засоби, з іншого;
- програми повинні бути короткими і зрозумілими, форма запису повинна бути максимально звичною для людини;
- мати розвинуті бібліотеки, можливості розробки різноманітних додатків;
- повинна підтримувати стилі як структурного, так і функціонального та об'єктно орієнтованого програмування;
- мова повинна бути сучасною, не «мертвою», входити хоча б у 10 найбільш уживаних у реальному виробництві мов програмування;
- середовище розробки програм не повинно бути перевантажене додатковими функціями і можливостями, складність освоєння яких «затьмарює» і відсуває на другий план алгоритмічну складність проєкту. Обов'язковою також є наявність консольного транслятора;
- методична доцільність.

Таким вимогам відповідає мова Python, яку використовують не тільки з навчальною метою. Станом на січень 2022 року Python займає перше місце в рейтингу популярності мов програмування TIOBE Programming Community Index (https://www.tiobe.com/tiobe-index). Мова активно розвивається, має відкритий (Open Source) програмний код, поширюється за GPL-ліцензією, є реалізації під основні операційні системи сімейств Windows, Linux, Mac OS. Мову Python використовують у системному програмуванні, для створення вебсайтів, ігор, програм з графічним інтерфейсом, для роботи з базами даних, для наукових розрахунків, для програмування систем штучного інтелекту тощо.

Зупинимось на останній вимозі і водночас найбільш дискусійній – методичній доцільності.

Мова Python взагалі-то не нова. Проте сьогодні вона набуває широкого розповсюдження і використання. Тому виникає необхідність вивчення мови програмування Python як у закладах середньої, так і вищої освіти. Це вимагає розроблення відповідної методики навчання.

Охарактеризуємо певні особливості мови Python (у деяких випадках у порівнянні з C++).

*Відступи – частина синтаксису*

У мові Python відступи є обов'язковими. Їх використовують як операторні дужки (аналог фігурних дужок у С-подібних мовах та begin-end у Pascal), тобто для задання певного блоку операторів. Наприклад, деякі вчителі витрачають багато зусиль і часу для того, щоб навчити учнів правильно робити відступи для виділення операторного блоку (мова йде про Pascal). Те ж стосується і мов C/C++. Це підтверджується і дослідженням [9], де зазначається, що одним з негативних моментів вивчення мови С у середній школі є те, що учні погано форматують код, написаний цією мовою. Тому

програмний код важко читати. Натомість у Python всі оператори, які містяться у блоці повинні мати однакові відступи (як правило, за замовчуванням чотири пробіли виставляються автоматично), наприклад:

```
for a in range(2,8):        #зовнішній цикл
    for b in range(3,9,2): #внутрішній цикл
        s=a+b               #обчислення площі
        print('s=',s)       #виведення значення площі
```

Аналогічно такий запис буде і в операторі розгалуження:

```
if x>0:         #якщо x>0
    if a>=0:
        y=a*x  # гілка Так другого оператора if
    else:
        y=a+x  # гілка Ні другого оператора if
else:
    y=x        # гілка Так першого оператора if
print('y=',y)
```

Проте тут не все так легко. Можуть виникати труднощі у випадку копіювання тексту програми з одного документу і вставляння його в іншу програму (не завжди зберігається відповідне форматування). На це треба також звертати увагу учнів.

*Динамічна типізація*

У мові Python реалізована динамічна типізація змінних, тобто не потрібно їх оголошувати. Одна й та сама змінна у різних місцях програмного коду може набувати значень різних типів. Наприклад:

```
a=1                         # ціле число
a=2.3                       # дійсне число
a="Python"                  # рядок
a=[2,3,4,5,6]               # список
a=(1,"Python")              # кортеж
a={"Python":1,"Pascal":2} # словник
```

Виникає цілком слушне питання: динамічна типізація – це добре чи ні? На перший погляд, це надзвичайно зручно: не треба наперед визначати тип змінної, наперед продумувати, яких значень вона може набути під час виконання програми і т.д. У коді програми не потрібно писати ще один рядок (чи рядки), де будуть оголошуватися змінні. Проте не все так добре і просто.

Розглянемо такий програмний код. На Рис. 1,а наведено програмний код мовою Python. У ньому помилка не виявиться навіть під час виконання. Її навіть важко виявити візуально, оскільки цифра «1» і латинська літера «l» мають однаковий вигляд. При виконанні умови x>y змінна al набуде значення x-y, а значення a1 змінної не зміниться. Очевидно, що в мовах програмування зі статичною типізацією, наприклад, C++ (Рис. 1, б), цю помилку буде виявлено на етапі трансляції програми (оскільки змінну al не було оголошено).

```
a1=9
x=int(input('Input x: '))
y=int(input('Input y: '))
if x>y:
    al=x-y
else:
    a1=y-x
print(a1)
```

*а) Мова Python*

```
int a1, x, y;
a1 = 9;
cout << "Input x:" << endl;
cin >> x;
cout << "Input y:" << endl;
cin >> y;
if (x > y) al = x - y; else  a1 = y - x;
cout << "a1=" << a1 << endl;
```

*б) Мова C++*

*Рис. 1*

У даному випадку з метою запобігання помилки програмний код необхідно протестувати для кожної гілки в операторі розгалуження. Для цього необхідно витратити деякий час.

Серед програмістів, які пишуть на Python, відома фраза «We are all consenting adults here», яка перекладається приблизно так: «Ми всі тут дорослі і за згодою». Це можна тлумачити так: програмування мовою Python надає повну свободу дій, але треба відповідати за свої дії (або перевіряти чи контролювати їх.

*Типи даних.* Дані у мові Python зберігаються у вигляді об'єктів. Як зазначено вище, у мові Python реалізована динамічна типізація змінних. Мова Python містить як прості, так і складені типи даних, а також ті, що розробляються користувачем засобами самої мови.

Найбільш вживаними типами даних у мові Python є цілі числа (int), дійсні числа (float), комплексні числа (complex), рядки (str), логічні дані (bool), списки (list), словники (dict), кортежі (tuple), множини (set), типи структурованих елементів (функції, модулі, класи). Для визначення типу, до якого належить об'єкт, використовується функція type.

*Цілі числа.* Цілі числа можуть бути довільної довжини, вони обмежуються тільки доступною пам'яттю. Це пояснюється тим, що розряди числа зберігаються як окремі елементи послідовності, а вона може бути довільної довжини. Наприклад, необхідно обчислити 1000!. У випадку використання мов Pascal чи C++ цю задачу можна розв'язати, наприклад, з використанням масиву або рядкових величин, або з використанням файлів. Натомість цілі числа у мові Python завжди розглядаються як «довгі». Тому задачі на «довгу» арифметику втрачають свою підвищену складність і, як наслідок, зникають із завдань на олімпіадах з програмування. Тут згадується вислів: програмування мовою Python – це технологія, а не мистецтво.

*Логічний тип.* Як і в більшості мов програмування, для цього типу визначені дві константи: True та False. Для побудови логічних виразів використовуються операції порівняння та логічні операції (and, or, not). Логічні вирази, крім безпосередньої перевірки умови (наприклад, 2==3), використовуються з операторами розгалуження if та циклу з передумовою while.

Зупинимось на особливостях використання об'єктів логічного типу у мові Python.

Для порівняння двох чисел, зважаючи на їх тип, використовується функція is. Наприклад:

```
>>> 1 is 1.0
False
>>> 5 is 5.0
False
>>> 5 is 5
True
>>> 1 is True
False
```

Для перевірки наявності значення в послідовності використовується функція in (для словників наявність значення серед ключів). Наприклад:

```
>>> a=[1,2,3,4]
>>> 2 in a
True
>>> 5 in a
False
```

Для перевірки належності об'єкта до певного типу використовується функція isinstance. Наприклад:

```
>>> isinstance(2.3,float)
True
>>> isinstance(2.3,str)
False
```

*Дійсні числа.* Як і в будь-якій мові програмування, візуальною ознакою дійсного числа є наявність в його записі розділювача «.» (так званий формат чисел з плаваючою крапкою). Це дає змогу зберігати та опрацьовувати як дуже великі, так і дуже малі за модулем числа. Проте такий підхід має недоліки, пов'язані з точністю подання числа. Наприклад:

```
>>> a=0.1+0.2
>>> a==0.3
False
>>> print(a)
0.30000000000000004
```

Тому під час проведення обчислень з дійсними числами треба враховувати похибку обчислень (як це видно, не завжди можливо здійснювати точні порівняння дійсних чисел). Тому рекомендується вважати два дійсних числа a і b рівними:

1) якщо abs(a-b)<eps (eps як завгодно мале, наприклад, eps=0.00001 або eps=10**-10).

```
>>> abs(a-0.3)<10**-10
True
```

2) якщо результатом функції isclose() є значення True:

```
>>> a=0.1+0.2
>>> import math
>>> math.isclose(a,0.3)
True
```

*Зауваження.* Для виконання високоточних наукових обчислень або обчислень у сфері фінансів використання об'єктів типу float може призводити до помилок. У таких випадках, коли необхідно проводити обчислення з високою точністю, доцільно використовувати об'єкти модуля Decimal.

Об'єкти модуля Decimal значно повільніші, ніж float, оскільки вони реалізовані програмно, а float – апаратно (або на низькому рівні). Створювати об'єкти Decimal можна з об'єктів цілого (int) або дійсного (float) типу, рядка (str) чи кортежа (tuple). Проте не рекомендується створювати Decimal з float, оскільки в Decimal потрапляє вже округлене число (можливо, і неправильне). Доцільно використовувати для створення об'єкта Decimal цілі числа або рядки. Наприклад:

```
>>> import decimal
>>> a=decimal.Decimal(1.2)
>>> print(a)
1.199999999999999955591079014993738383054733276367 1875
>>> b=decimal.Decimal('1.2')
>>> print(b)
1.2
```

Зауважимо, що у випадку ділення чисел, де ділене і дільник цілі, частка отримується дійсного типу. Тому як аргумент функції Decimal не рекомендується також використовувати вирази, які містять операцію ділення.

Розглянемо інший приклад. У мові Python при конструюванні функції не вказується тип її параметрів. Як наслідок, при їх використанні можуть бути неочікувані проблеми. Тому необхідно або самому слідкувати за типами значень, які передаються

до функції, або у тілі функції робити перевірку на коректність переданих значень параметрів і, відповідно, робити про це повідомлення. Другий варіант нам видається доцільнішим. Також не перевіряється тип значення, яке повертає функція. Як приклад розглянемо створення функції для обчислення факторіала числа n.

Згідно з означенням факторіалом числа n називається добуток всіх натуральних чисел від 1 до n. Також прийнято, що 0!=1 і 1!=1. На рис. 2 наведено функції для обчислення n!, створені мовою Python та C++.

```python
def fact(n):
    d=1
    if n<0:
        return "Введіть невід'ємне ціле число"
    elif n==0 or n==1:
        return d
    while n>1:
        d*=n
        n-=1
    return d
```

*а) Мова Python*

```cpp
long int fact(int n)
{
    int d = 1;
    if (n >= 0)
    {
    if ((n == 0) or (n == 1)) return d;
     while (n > 1)
          {
              d *= n;
              n -= 1;
          }
     return d;
    }
    else  { cout << "Введіть невід'ємне ціле число. ERROR"; }
}
```

*б) Мова C++*

*Рис. 2*

Проаналізуємо функцію для обчислення факторіалу числа n, створену мовою Python (Рис. 2, а). Результат її буде коректним для всіх цілих невід'ємних чисел (разом з числами типу float, дробова частина яких дорівнює нулю). Проте якщо число n буде типу float з дробовою частиною відмінною від нуля, то буде отримуватися некоректний результат. Наприклад:

```
>>> print(fact(5.5))
324.84375
```

Очевидно, що такий результат не має сенсу, оскільки факторіал числа можна обчислити тільки для невід'ємних цілих чисел. Такого можна уникнути, додавши до тіла функції умову перевірки типу числа n: type(n)!=int. Тоді умова перевірки на коректність набуде вигляду:

```
if n<0 or type(n)!=int:
        return "Введіть невід'ємне ціле число"
```

Розглянемо тепер функцію для обчислення факторіала числа n, створену мовою C++. Дія її є аналогічною до функції, створеної мовою Python. Проте є одна відмінність, яка полягає в опрацюванні дійсного числа. На відміну від мови Python, у мові C++ статична типізація змінних. Тому згідно правил зведення типів змінна, яка оголошується цілим типом (int, long int, short) набуває значення цього типу, навіть якщо її задати числом дійсного типу. Наприклад, якщо ініціалізувати змінну так

```
int a=5.5;
```

то в програмному коді змінна **a** набуває значення 5, тобто a=5.

Тому, повертаючись до функції обчислення факторіала у мові C++, якщо параметр функції набуває значення дійсного типу, то він буде перетворюватися до цілого типу відкиданням дробової частини. Наприклад, якщо у функцію передати n=5.5, то обчислиться 5!, що також буде некоректним результатом:

```
cout << fact(5.5)<<endl;
Значення 5.5!=120
```

Такого некоректного результату можна уникнути так:

1) параметр n функції fact оголосити дійсного типу (наприклад, float);
2) у тілі функції ввести додаткову змінну (наприклад, tmp) цілого типу;

3) змінній tmp надати значення параметра n;

4) умову перевірки введення коректності даних записати так: ((n >= 0) and (tmp == n)).

*Компактність коду*

Однією з переваг мови Python є компактність програмного коду. Наприклад, розглянемо задачу обміну значеннями між двома змінними a і b (рис. 3).

```python
a=int(input('Input a: '))
b=int(input('Input b: '))
print('a=',a,' b=',b)
a,b=b,a
print('a=',a,' b=',b)
```

*а) мова Python*

```cpp
int a, b, tmp;
cout << "Input a:" << endl;
cin >> a;
cout << "Input b:" << endl;
cin >> b;
cout << "a=" << a << " b=" << b << endl;
tmp = a;
a = b;
b = tmp;
cout << "a=" << a << " b=" << b;
```

*б) мова C++*

*Рис. 3*

Слід зауважити, що в такому випадку (мова Python) також втрачається зміст завдання про обмін значеннями між двома змінними a і b, без використання додаткової змінної. Те саме стосується і знаходження мінімального чи максимального значення з двох змінних, без використання оператора розгалуження.

Або інший фрагмент коду для створення масиву (чи списку у Python), який містить сто п’ятірок (рис.4).

```python
>>> a=[5]*100
```

*а) мова Python*

```cpp
int a[100];
for (int i = 0; i < 100; i++)
    a[i] = 100;
```

*б) мова C++*

*Рис. 4*

*Масиви, списки, словники, кортежі, множини.*

Окремо слід зупинитися на масивах. Під масивом (для мов C++ чи Pascal) розуміють сукупність однотипних даних, тобто в масиві можна зберігати дані тільки одного типу. Опрацювання масивів виконується з використанням циклів. Стандартними алгоритмами опрацювання масивів є:

*для одновимірних*:

- введення та виведення;
- обчислення деяких характеристик (суми, добутку, середнього арифметичного тощо);
- пошук мінімального (максимального) елемента;
- пошук у впорядкованому та невпорядкованому масивах;
- перестановка (обмін) елементів масиву;
- вставка, видалення, циклічний зсув елементів масиву;
- сортування;

*для двовимірних*:

- введення та виведення;
- створення нового масиву за заданим алгоритмом;

– пошук елементів масиву за визначеним критерієм;
– визначення, чи відповідає масив або окремі його елементи заданій властивості;
– виконання певних операцій над елементами масиву (транспонування, множення матриць, переставляння рядків (стовпців) тощо).

Слід зауважити, що опрацювання двовимірних масивів здійснюється обходом його елементів за рядками або стовпцями.

У мові Python масиви та методи і функції їх опрацювання визначені у модулі array. Проте з масивами у Python можна працювати й без використання цього модуля (його, як правило, використовують рідко та з метою досягнення високої швидкості роботи). Тому рекомендують масив типу array замінити такими типами даних як список, кортеж чи словник. Найчастіше для подання масиву використовують список. Зауважимо, що поняття «масив» і «список» не є тотожними. У Python список – це не обов'язково набір однотипних елементів.

У списках можна змінювати значення елементів, видаляти та вставляти елементи, здійснювати пошук, впорядковувати його елементи, тобто можна реалізовувати всі базові алгоритми опрацювання масивів. Слід зауважити, що в Python містяться вбудовані функції та методи опрацювання списків (наприклад, пошук мінімального чи максимального елемента, сортування, вставлення та видалення елемента). Крім того, для опрацювання двовимірних списків (матриць) не обов'язково «обходити» всі його елементи: у мові Python є можливість застосовувати функції до кожного з рядків. Зауважимо, що матрицю у мові можна розглядати як список списків однакової довжини. Розглянемо приклад.

Приклад. Перевірити, чи число є паліндромом.

Програмна реалізація мовою Python наведено на рис.5.

```
n=int(input("Введіть натуральне число n: "))
tmp=n;
s=0
while tmp>0:
    rev=tmp%10
    s=s*10+rev
    tmp=tmp//10
if n==s:
    print("Число ", n," є паліандромом")
else:
    print("Число ", n," не є паліандромом")
```

*а) з використанням циклічної структури*

```
n=input("Введіть натуральне число:")
tmp=n
tmp=list(tmp)
tmp.reverse()
tmp="".join(tmp)
if n==tmp:
    print("Число є паліандромом")
else:
    print("Число не є паліандромом")
```

*б) з використанням списків*

```
n=input("Введіть натуральне число: ")
if n==n[::-1]:
    print("Число є паліндромом")
else:
    print("Число не є паліндромом")
```

*в) з використанням зрізів*

*Рис. 5*

Постає цілком слушне питання: методи та функції опрацювання масивів (чи списків, кортежів, словників та рядків у мові Python) доцільно вивчати до циклів чи після? Для таких мов програмування, як C++ чи Pascal, відповідь: після. Якщо вивчається мова Python, то не так все однозначно. Як видно (Рис. 5,б та Рис. 5,в),

програмний код для даної задачі можна скласти без використання циклічних структур. На нашу думку, для кращого засвоєння учнями методів та функцій опрацювання таких типів даних, як списки, кортежі та словники, їх доцільно вивчати до циклів. Зауважимо, що програмна реалізація даної задачі мовою C++ є подібною до реалізації мовою Python з використанням циклічних структур (зрозуміло з відповідними описами змінних та врахуванням особливостей синтаксису).

Приклад. Знайти мінімальний елемент квадратної матриці.

На рис. 6 наведено програмну реалізації мовами Python та C++.

```python
mas=[[1,2,3],[4,2,6],[5,7,8]]
n=len(mas)
a=[]
for i in range(n):
    mn=min(mas[i])
    a.append(mn)
print("Мінімальний елемент матриці: ",min(a))
```

*а) мова Python*

```cpp
const int n = 3;
int mas[n][n] = { {1,2,3},{4,2,6},{5,7,8} };
int mn;
mn = mas[0][0];
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        if (mas[i][j] < mn) mn = mas[i][j];
cout << "Мінімальний елемент матриці: " << mn << endl;
```

*б) мова C++*

*Рис. 6*

Проаналізуємо кожну з них. Для програмної реалізації мовою Python (Рис. 6,а) матрицю задаємо за допомогою двовимірного списку (змінна mas). Розмірність матриці (довжину списку) визначаємо за функцією len (змінна n). Для кожного рядка матриці (списку) знаходимо мінімальний елемент (змінна mn) і додаємо (метод append) його до списку (змінна a). Сформувавши в такий спосіб список a мінімальних елементів за рядками, знаходимо серед них мінімальний. Для цього використовується тільки один цикл. Зауважимо, що вхідна квадратна матриця може бути довільної розмірності (як приклад наведено матрицю розмірності 3×3), оскільки розмірність визначається вже після введеного списку.

Цей приклад мовою Python можна розв’язати значно простіше (з використанням функції map()):

```python
mas=[[1,2,3],[4,2,6],[5,-1,8]]
mn=list(map(min,mas)) #мінімальний по рядках
mn=min(mn)            #мінімальний елемент
print('Мінімальний елемент матриці: ', mn)
```

Проаналізуємо програмну реалізацію мовою C++. На відміну від мови Python, тут ініціалізовано масив (int mas[n][n]={{1,2,3},{4,2,6},{5,7,8}};), попередньо визначивши його розмірність за допомогою константи n (const int n=3;). Для знаходження мінімального значення припускаємо, що ним є елемент mas[0][0] (mn=mas[0][0]). Далі перевіряємо, чи є меншим поточний елемент від мінімального (if (mas[i][j]<mn) mn=mas[i][j];). Це виконується за допомогою двох циклів: внутрішнього і зовнішнього. Зауважимо: якщо визначати розмірність масиву програмно, то необхідно використовувати динамічні масиви.

Для деяких задач у мові Python замість списків можна використовувати і словники, наприклад, для дослідження задач з використанням графів.

Не можна замінювати поняття «масиви» багаторівневими списками, велика кількість алгоритмів, які реалізовані у стандартних методах та функціях, оскільки учні не будуть розуміти внутрішньої їх організації. Зупинимось на цьому детальніше. Загальний методичний підхід до цього можна запропонувати такий: спочатку вивчається і розуміється, як це працює, а потім використовуються відповідні функції (чи методи) мови Python. Наведемо приклади.

1. Обмін значеннями між двома змінними. Спочатку учням пропонується написати обмін значень між змінними a і b з використанням додаткової, а потім повідомляється, що у Python це можна зробити за допомогою конструкції (a, b)=(b, a).
2. Максимальне (мінімальне) значення з двох, трьох чисел.
3. Пошук максимального (мінімального) елемента в масиві.
4. Сортування масивів

Після ознайомлення учнів з фундаментальними алгоритмами надалі пропонується використовувати відповідні конструкції мови Python, зокрема max (min) та sort.

В учнів виникають цілком слушні запитання, серед яких слід виокремити такі:

– навіщо знати алгоритм обміну значення між двома змінними?
– навіщо знати алгоритми знаходження мінімального та максимального значення двох чи більше змінних?
– навіщо вивчати алгоритми сортування?

Тут можна запропонувати такий підхід. Учні реалізовують наведені алгоритми, а перевірку здійснюють вже з використанням вбудованих функцій, наприклад, min та max – для знаходження мінімального та максимального елемента відповідно, sort – сортування списків (масивів) тощо. Заборону на використання вбудованих функцій учень може подолати шляхом усвідомлення особливостей (тонкощів) роботи алгоритмів. А це вже завдання вчителя, який повинен намагатися це пояснити.

І ще один негативний момент, на який також треба звернути увагу. Учні можуть зіткнутися з тим, що після мови Python, другою мовою вивчення може бути «менш» високорівнева, наприклад C++, для якої вже не буде динамічної типізації змінних, не буде вбудованих функцій для опрацювання масивів (списків) тощо. Тобто в учнів можуть виникати труднощі переходу від вивчення однієї мови програмування до іншої, у даному випадку від Python до C++.

Тут мимоволі згадуються слова Б. Страуструпа – розробника C++: «Є два типи мов програмування – ті, які всі критикують, і ті, на яких ніхто не пише».

На нашу думку, не потрібно відмовлятися від ідеї вивчення «надто» високорівневих мов програмування (до яких належить і Python) у закладах загальної середньої освіти. Вивчення основ алгоритмізації та програмування з використанням мови Python учнями, які навчаються за профільним рівнем, за умови врахування методичних особливостей буде сприяти кращій підготовці програмістів-початківців, які мають різносторонній досвід у написанні програмного коду. Це пояснюється тим, що сучасні високорівневі мови програмування є простими, зручними для сприйняття та відлагодження.

Високорівневість мови змушує пізніше розбиратися з нюансами програмування, роботою з пам’яттю, процесами та іншими низькорівними концепціями. Завдання вчителя полягає в тому, щоб пояснити учням, що вони повинні розуміти, як працює той чи інший метод або функція, перш ніж їх використовувати. А для учнів, які не бачать себе програмістами, які мають низький рівень мотивації для ґрунтовного вивчення програмування, така мова, як Python, не дозволить перетворити навчання на «страшні муки», а буде зручним інструментом, який у майбутньому вони за потреби зможуть використовувати.

Мови вищого рівня дозволяють більш коротко реалізовувати складні обчислення, опрацьовувати різноманітні дані тощо.

Можна зустріти думку, що основи алгоритмізації і програмування не потрібно вивчати у школі, оскільки не всі учні в майбутньому стануть (чи бажають стати) програмістами. Так, не всі. Як і фізиками, хіміками, біологами, істориками чи філологами. Основне завдання змістової лінії основ алгоритмізації та програмування –

це формування логічного та алгоритмічного стилю мислення. Це, зокрема, досягається вивченням елементів структурного програмування. Вивчення основ об'єктно орієнтованого програмування сприятиме розвитку в учнів абстрактного мислення (побудова абстрактних моделей об'єктів реального світу). Тому в школі необхідно вивчати основи алгоритмізації та програмування.

## 4. ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

На нашу думку, програмувати мовою Python можуть усі цілеспрямовані учні. На сьогодні розділи з основ алгоритмізації та програмування вивчаються в усіх класах разом з іншими розділами інформатики. Отже, принцип неперервності навчання основ алгоритмізації та програмування не забезпечується. Це є надзвичайно складною проблемою. Можливим виходом з такої ситуації є виокремлення змістової лінії основи алгоритмізації і програмування в окремий курс. Тут також виникають питання: який зміст курсу, з якого класу починати вивчати, які мови програмування, скільки годин виділяти і т.д., чим замінити ці розділи в курсі інформатики.

На нашу думку, навчання основ алгоритмізації і програмування для учнів 5-9 класів можна здійснювати в такий спосіб. Для учнів 5-6 класів вивчати основні алгоритми з використанням мов Scratch та Python (модуль turple). У 7-9 класах перейти до ґрунтовнішого вивчення основ алгоритмізації та програмування з використанням мови Python. У 10-11 класах інформатичного профілю можна обрати інші мови, наприклад C++, C# чи Java. Зауважимо, що вибір залежить від педагога і наявного навчально-методичного та матеріально-технічного забезпечення. Наприклад, для більшої зацікавленості учнів можна обрати Java, оскільки з її використанням можна створювати застосунки, які працюють під управлінням Android. Хоча зауважимо, що і з використанням мови Python можна створювати достатньо цікаві для учнів проєкти.

Найкращий спосіб сформувати власну думку про мову програмування – це випробувати її в дії. Тому вибір мови програмування залежить від закладу загальної середньої освіти чи вчителя. Власне програмами з інформатики не вказується вивчення якоїсь конкретної мови програмування. Водночас шкільні підручники з інформатики 2020 року випуску орієнтовані на вивчення мови Python. Це стосується як 7-9 класів, так 10-11 класів, які навчаються за профільним рівнем. Тому необхідні нові методичні розробки, які б якнайкраще враховували особливості мови Python, зокрема динамічну типізацію та «надто велику» високорівневість.

Подальші дослідження будуть спрямовані на розроблення методики навчання основ алгоритмізації та програмування з використанням мови Python у класах інформатичного профілю.

Усі демонстраційні приклади виконані в середовищі IDLE (version 3.7.8, мова Python) та Visual Studio 2019 (мова C++).

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] "Яку мову програмування вивчати у школі", *Комп'ютер у школі та сім'ї,* №7, с.14-18, 2013.
[2] "Яку мову програмування вивчати у школі", *Комп'ютер у школі та сім'ї*, №8, с.9-18, 2013.
[3] M. Saeli, J. Perrenet, W. Jochems, and B. Zwaneveld, "Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective", *Informatics in Education,* vol.10, n.1, pp.73-88, 2011. doi: https://doi.org/10.15388/infedu.2011.06.
[4] Y. Lee, and J. Cho, "The Influence of Python Programming Education for Raising Computational Thinking", *International Journal of u- and e- Service, Science and Technology*, vol.10, n.8, pp.59-72, 2017. doi: https//doi.org/10.14257/ijunesst.2017.10.8.06.

[5] I. Yoon, J. Kim, and W. Lee, "The analysis and application of an educational programming language (RUR-PLE) for a pre-introductory computer science course", *Cluster Computing*, vol. 19, n. 1, pp.529-546, 2016. doi: https://doi.org/10.1007/s10586-016-0540-6.
[6] M. Ateeq, H. Habib, A. Umer, and M. U. "Rehman, "C++ or Python? Which One to Begin with: A Learner's Perspective," in *International Conference on Teaching and Learning in Computing and Engineering (LaTiCE)*, Kuching, Malaysia, 2014, pp. 64-69. doi: https://doi.org/10.1109/LaTiCE.2014.20.
[7] C. Lo Y. Lin, and C. Wu, "Which Programming Language Should Students Learn First? A Comparison of Java and Python," in *International Conference on Learning and Teaching in Computing and Engineering (LaTiCE)*, Taipei, Taiwan, 2015, pp. 225-226. doi: https://doi.org/10.1109/LaTiCE.2015.15.
[8] J-P. Pellet, A. Dame, and G. Parriaux, "How beginner-friendly is a programming language? A short analysis based on Java and Python examples", in *12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, Larnaca, Cyprus, 2019, pp.48-56.
[9] E. Mészárosová. "Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools?", *International Journal of Information and Communication Technologies in Education*, 4(2), pp.5-14, 2015. doi: https://doi.org/10.1515/ijicte-2015-0005.
[10] J. Monsálvez. "Python como primer lenguaje de programación textual en la enseñanza secundaria", *Educationn the Knowledge Society*, vol 18, n. 2, pp.147-162. 2017. doi: https://doi.org/10.14201/eks2017182147162.
[11] B. Fagan, and B. Payne, "Learning to Program in Python – by Teaching It!", in *Proceedings of the Interdisciplinary STEM Teaching and Learning Conference*, Vol. 1, pp.99-107, 2017. doi: https://doi.org/10.20429/stem.2017.010109.
[12] P. H. Shevchuk, «Основні підходи добору мови та середовища програмування як засобів навчання», *ITLT*, вип. 17, вип. 3, Вер 2010. doi: https://doi.org/10.33407/itlt.v17i3.251.
[13] V. M. Bazurin, «Середовища програмування як засіб навчання учнів основ програмування», *ITLT*, вип. 59, вип. 3, с. 13-27, Чер 2017. doi: https://doi.org/10.33407/itlt.v59i3.1601.
[14] O. V. Semenykhina, та Rudenko Y. O., «Проблеми навчання програмувати учнів старших класів та шляхи їх подолання», *ITLT*, вип. 66, вип. 4, с. 54-64, Вер 2018. doi: https://doi.org/10.33407/itlt.v66i4.2149.
[15] О. Маловічко, та С. Конюхов, "Застосування спеціалізованого педагогічного програмного комплексу у процесі вивчення програмування у восьмому класі", *Ukrainian Journal of Educational Studies and Information Technology*, 5 (4), с. 38-55, 2017. doi: https://doi.org/10.32919/uesit.2017.04.04.
[16] Л. Міцкан, Т. Вербицька, та В. Базурін, "Порівняльний аналіз мов Python і Free Pascal як перших мов програмування для учнів 8 класу", *Актуальні питання природничо-математичної освіти*, № 2 (10), с. 130-139, 2017.
[17] В. Д. Руденко, та О. О. Жугастров, *Основи алгоритмізації і програмування*. Харків, Україна: Вид-во «Ранок», 2019.
[18] В. Д. Руденко, Н. В. Речич, та В. О. Потієнко, *Інформатика (профільний рівень): підруч. для 10 кл. закл. загал. серед. освіти*. Харків, Україна: Вид-во «Ранок», 2018.
[19] Інформатика. Навчальна програма вибірково-обов'язкового предмету для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту). [Електронний ресурс]. Доступно: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/informatika-standart-10-11.docx. Дата звернення: Груд. 19, 2021.
[20] Інформатика для 10-11 класів (профільне навчання). [Електронний ресурс]. Доступно: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx. Дата звернення: Груд. 19, 2021.

*Матеріал надійшов до редакції 08.02.2022 р.*

## PYTHON AS A TOOL FOR LEARNING BASICS OF ALGORITHMIZATION IN GENERAL SECONDARY EDUCATION INSTITUTIONS

**Taras P. Kobylnyk**
PhD of Pedagogical Sciences
Associate Professor at Informatics and Information Systems Department
Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine
ORCID ID 0000-0002-2703-7570
*kobylnyktaras@gmail.com*

**Oksana V. Sikora**
PhD of Technical Sciences, Head of the Department of Informatics and Information Systems, Associate Professor at Informatics and Information Systems
Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine
ORCID ID 0000-0002-4043-778X
*sikora60@ukr.net*

**Volodymyr B. Zhydyk**
Senior Lecturer at Informatics and Information Systems Department
Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine
ORCID ID 0000-0002-9876-0053
*Zhvb63@gmail.com*

**Oleksandra V. Sharan**
PhD of Pedagogical Sciences, Associate Professor at the Mathematics, Informatics and Teaching Methods in Primary School Department
Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine
ORCID ID 0000-0003-3198-8026
*sharan_oleks@ukr.net*

**Abstract**. Today, there is a tendency to expand programming, and programming languages are becoming simpler and more convenient. As a result, the number of programmers will increase. At the same time, the level of deep understanding of the basics of algorithmization will decrease. This is inevitable, but it is not a reason to abandon the study of modern high-level programming languages. The problem of choosing a programming language for learning the basics of algorithmization at the school is actual and complex. Based on the analysis of scientific publications and personal experience, the criteria for choosing a programming language as a means of learning the basics of algorithmization are determined: modernity, freeness, clarity, and conciseness. The choice of Python as a means of teaching the basics algorithmization is substantiated and certain of its characteristics are analyzed, which are illustrated by specific examples. To better understand the features of the Python language, the software implementation of algorithms is presented in parallel with the C ++ language. However, learning Python as the first programming language may raise some concerns. There are a dynamic typing of variables and a "too high" high-level of language. The main problem with learning the Python programming in schools is the lack of developed methods of teaching it, unlike Pascal. It is necessary to clarify what topics, and algorithms need to be studied in schools and how to form a sequence of topics and choose the appropriate teaching methods. It is better to focus on learning algorithms and developing skills for solving various problems that they will be able to use in practice in the future. Therefore, the main emphasis should not be on learning a particular programming language, but on learning the basics of algorithmization, and using a programming language as an aid. It is advisable to choose a programming language that is easy to read and easy to write. This will help increase the level of confidence of students who do not have very good knowledge of the basics of algorithmization and who in the future do not link their professional activities with programming. Further research will be aimed at developing methods for teaching the basics of algorithmization and programming using Python in computer science classes.

**Keywords:** Python language; institution of general secondary education; basics of algorithmization, programming.

## REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] "What programming language to learn at school", *Computer at School and Family*, no.7, 2013. pp.14-18 (in Ukrainian)
[2] "What programming language to learn at school", *Computer at School and Family*, no.8, pp.9-18, 2013. (in Ukrainian)
[3] M. Saeli, J. Perrenet, W. Jochems, and B. Zwaneveld, "Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective", *Informatics in Education*, vol.10, n.1, pp.73-88, 2011. doi: https://doi.org/10.15388/infedu.2011.06. (in English)

[4] Y. Lee, and J. Cho, "The Influence of Python Programming Education for Raising Computational Thinking", *International Journal of u- and e- Service, Science and Technology*, vol.10, n.8, pp.59-72, 2017. doi: https://doi.org/10.14257/ijunesst.2017.10.8.06. (in English)
[5] I. Yoon, J. Kim, and W. Lee, "The analysis and application of an educational programming language (RUR-PLE) for a pre-introductory computer science course", *Cluster Computing*, vol. 19, n. 1, pp.529-546, 2016. doi: https://doi.org/10.1007/s10586-016-0540-6. (in English)
[6] M. Ateeq, H. Habib, A. Umer, and M. U. "Rehman, "C++ or Python? Which One to Begin with: A Learner's Perspective," in *International Conference on Teaching and Learning in Computing and Engineering (LaTiCE)*, Kuching, Malaysia, 2014, pp. 64-69. doi: https://doi.org/10.1109/LaTiCE.2014.20. (in English)
[7] C. Lo Y. Lin, and C. Wu, "Which Programming Language Should Students Learn First? A Comparison of Java and Python," in *International Conference on Learning and Teaching in Computing and Engineering (LaTiCE)*, Taipei, Taiwan, 2015, pp. 225-226. doi: https://doi.org/10.1109/LaTiCE.2015.15. (in English)
[8] J-P. Pellet, A. Dame, and G. Parriaux, "How beginner-friendly is a programming language? A short analysis based on Java and Python examples", in *12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, Larnaca, Cyprus, 2019, pp.48-56. (in English)
[9] E. Mészárosová. "Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools?", *International Journal of Information and Communication Technologies in Education*, 4(2), pp.5-14, 2015. doi: https://doi.org/10.1515/ijicte-2015-0005. (in English)
[10] J. Monsálvez. "Python as a First Textual Programing Language in Secondary Education", *Educationn the Knowledge Society*, vol 18, n. 2, pp.147-162. 2017. doi: https://doi.org/10.14201/eks2017182147162. (in Spanish)
[11] B. Fagan, and B. Payne, "Learning to Program in Python – by Teaching It!," *Proceedings of the Interdisciplinary STEM Teaching and Learning Conference*, Vol. 1, pp.99-107, 2017. doi: https://doi.org/10.20429/stem.2017.010109. (in English)
[12] P. H. Shevchuk, "Criteria of language and programming environment selection for use in the capacity of educational aids", ITLT, vol. 17, no. 3, Sep. 2010. doi: https://doi.org/10.33407/itlt.v17i3.251. (in Ukrainian)
[13] V. M. Bazurin, "Programming environments as a means of teaching pupils to programming basics", ITLT, vol. 59, no. 3, pp. 13-27, Jun. 2017. https://doi.org/10.33407/itlt.v59i3.1601. (in Ukrainian)
[14] O. V. Semenykhina and Rudenko Y. O., "Problems of educating to programming of students and way of their overcoming", *ITLT*, vol. 66, no. 4, pp. 54-64, Sep. 2018. doi: https://doi.org/10.33407/itlt.v66i4.2149. (in Ukrainian)
[15] O. Malovichko, and S. Koniukhov, "The use of specialized pedagogical software in the study of programming in the eighth grade", *Ukrainian Journal of Educational Studies and Information Technology*, 5 (4), 2017, pp. 38-55 (in Ukrainian)
[16] L. Mitskan, T. Verbytska, and V. Bazurin, "Comparative analysis of Python and Free Pascal languages as the first programming languages for 8th grade students", *Topical issues of natural science and mathematics education*, no.2 (10), pp.130-139, 2017. (in Ukrainian)
[17] V. D. Rudenko, and O. O. Zhuhastrov, *Basics of Algorithms and Programming*. Kharkiv, Ukraine, Vyd-vo «Ranok», 2019, 192 p. (in Ukrainian)
[18] V. D. Rudenko, N. V. Rechych, and V.O. Potiienko, *Informatics* (*profile level*): *a Textbook for 10th Grade Secondary Schools*. Kharkiv, Ukraine, Vyd-vo «Ranok», 2018, 256 p. (in Ukrainian)
[19] Informatics for 10-11 grades (profile education). [Online]. Available: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx. Accessed on: Dec. 19, 2021. (in Ukrainian)
[20] Informatics. The curriculum of an elective compulsory course for students in grades 10-11 of general educational institutions (standard level). [Online]. Available: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/informatika-standart-10-11.docx. Accessed on: Dec. 19, 2021. (in Ukrainian)