МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
Кафедра системного аналізу та теорії оптимізації

А.Ю.Брила, М.М. Ломага, А.С. Вощепинець

АРИФМЕТИЧНІ ВИРАЗИ У PYTHON. ЛІНІЙНІ АЛГОРИТМИ

Методичні вказівки до лабораторних робіт з дисципліни «Програмування»

Ужгород 2023

Арифметичні вирази у Python. Лінійні алгоритми. (Методичні вказівки до лабораторних робіт з дисципліни «Програмування»). /Укладачі: А.Ю. Брила, М.М. Ломага, А.С. Вощепинець. – Ужгород, 2023.– 19 с.

Навчальний матеріал методичних вказівок призначений для аудиторної і самостійної підготовки студентів при вивченні дисципліни «Програмування».

Основна мета самостійної роботи студента – закріплення теоретичних відомостей, які викладаються на лекціях, та вміння їх застосувати, розв'язуючи задачі, що виникають на практиці. Завдання розроблених методичних матеріалів полягає в чіткій, цілеспрямованій допомозі студентам в організації самостійної підготовки до практичних занять з дисципліни «Програмування».

Методичні вказівки призначені для студентів різних напрямків підготовки.

Рецензенти:

к.ф.-м.н., доц. Погоріляк О.О.,

к.ф.-м.н., доц. Млавець Ю.Ю.

Рекомендовано до друку:
Кафедрою системного аналізу та теорії оптимізації (Протокол №10 від 18 травня 2023 року);
Науково-методичною комісією факультету математики та цифрових технологій, (Протокол № 9 від 23 травня 2023 року);
Вченою радою факультету математики та цифрових технологій ДВНЗ "Ужгородський національний університет", (Протокол №9 від 25 травня 2023 року).

МОВА ПРОГРАМУВАННЯ. ПОНЯТТЯ ПРОГРАМИ

Мова програмування – фіксована система позначень та правил для опису структур даних та алгоритмів, призначених для виконання обчислювальними машинами.

З формальної точки зору ***мова програмування*** – це набір вихідних символів *(алфавіт)* разом із системою правил утворення з цих символів формальних конструкцій *(синтаксис)* та системою правил їх тлумачення (інтерпретації) *(семантика),* за допомогою яких описуються алгоритми. **Програма –** це алгоритм записаний за допомогою мови програмування.

Алфавіт, синтаксис та семантика — це три основні складові частини будь-якої мови програмування. До **алфавіту** мови програмування, як правило, входять літери латинського алфавіту, арабські цифри, знаки арифметичних операцій, розділові знаки, спеціальні символи. Із символів алфавіту будують послідовності, які називають *словами (лексемами).* Кожне слово у мові програмування має своє змістове призначення. **Правила синтаксису** пояснюють, як потрібно будувати ті чи інші мовні повідомлення для опису всіх понять мови, здійснення описів та запису вказівок. **Правила семантики** пояснюють, яке призначення має кожен опис та які дії повинна виконати обчислювальна машина під час виконання кожної із вказівок. Вказівки на виконання конкретних дій називають ще **командами** або **операторами** мови.

Усі слова або ж лексеми, поділяють на

- *службові* (зарезервовані або ж ключові) слова;
- *стандартні ідентифікатори;*
- *ідентифікатори користувача;*
- *знаки операцій;*
- *літерали;*
- *розділові знаки*.

Службові слова мають наперед визначене призначення і використовуються для формування структури програми, здійснення описів, позначення операцій, формування керуючих конструкцій (вказівок). Наприклад, службовими словами для мови Python є: 'and', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'yield'.

Імена (позначення) для програмних об'єктів (типів даних, констант, змінних, функцій і т. п.) формують у вигляді ідентифікаторів. Правильний *ідентифікатор* — це послідовність латинських літер, цифр і символу підкреслення, яка розпочинається з латинської літери або символу нижнього підкреслювання.

Стандартні ідентифікатори використовуються як імена для стандартних (передбачених авторами мови програмування) типів даних, констант, підпрограм (зокрема, стандартних математичних функцій). Приклад: int, long, float, string, bool, complex та ін.

Ідентифікатори користувача є іменами тих програмних об'єктів (констант, змінних, функцій тощо), які створює сам користувач. Не можна в якості

ідентифікаторів користувача використовувати службові слова та стандартні ідентифікатори. Не можна також використовувати імена, які починаються і закінчуються двома підкресленнями.

Коментар – невиконувана частина тексту програми, що ігнорується компілятором і служить для вставки деяких поміток у програмі тільки для програміста. Коментарі бувають однорядковими та багаторядковими. Однорядковий коментар починається з символу «#» і закінчується у кінці рядка. Тобто всі символи до кінця рядка вважаються коментарем.

тип літерала	*опис*	*приклади*
цілочисловий	у десятковій системі числення – звичне нам ціле число	125, -89, 108
	у вісімковій системі числення – починається з 0	023, 075, 0416;
	у шістнадцятковій системі числення – починається з 0x або 0X	0x4A, 0x6FE2, 0xABC
дійсного типу	у форматі з фіксованою крапкою – у записі числа є крапка, що розділяє цілу і дробову частити	25.69, 2.0, 145.058
	у форматі з плаваючою крапкою – у записі використано символ «e», що розділяє мантису від порядку	3.5e5, 3e12, 6.78e2
символьний	заданий у явному вигляді рядок тексту поміщається у одинарні чи подвійні лапки	"Hello", 'Hello'
	прості ескейп-послідовності – службові символи починаються з символу «\»	\n -перехід на новий рядок, \t -горизонтальна табуляція, \' - апостороф, і т. д.
	ескейп-послідовності Unicode – символи «\u», за якими вказують код символу з чотирьох цифр у шістнадцятковій системі числення	'\u0123' , '\u3A58 '
логічний	може приймати два значення	True, False
порожньої адреси (нульовий літерал)	використовується у випадку, коли покажчик не містить жодної адреси	None

Приклад.

```
# Це однорядковий коментар
```

Багаторядковий коментар може бути вставлений як багаторядковий текст і міститься всередині потрійних одинарних чи подвійних лапках. Використовується при документуванні функцій.
Приклад.

```
'''Це
   багаторядковий
   коментар'''
```

```
"""Це
   багаторядковий
   коментар"""
```

Літерал – це явно вказане значення декого типу.

Розділові знаки служать для відокремлення однієї лексеми від іншої. Ними є пробіл, табуляція, символ нового рядка, крапка з комою, коментарі.

ВЕЛИЧИНА. ТИП ВЕЛИЧИНИ

У своїй роботі програміст має справу з таким поняттям, як величина. З точки зору програмування ***величини*** – це дані, що обробляються програмами. *Дані* — це інформація, введена у пам'ять комп'ютера або підготовлена до введення.

Носіями даних у програмах є *константи, змінні* (значення яких зберігається в оперативній пам'яті) та *файли* (на зовнішніх носіях інформації).

Константи — це величини, значення яких у процесі виконання програми не змінюється. *Змінні* — це величини, значення яких у процесі виконання програми можуть змінюватися. Імена констант і змінних, як і інших програмних об'єктів, записують у формі ідентифікаторів. Кожна змінна і константа належать до визначеного типу.

Тип даних – це сукупність властивостей певного набору даних, від яких залежать: діапазон значень, якого можуть набувати ці дані, а також сукупність операцій, які можна виконувати над цими даними.

З іншого боку **тип даних** – це описання того, яку структуру, розмір мають комірки оперативної пам’яті при зберіганні відповідного елемента даних.

Елемент даних певного типу – це комірка або комірки оперативної пам’яті, що мають фіксовану адресу, розряди яких розшифровуються згідно описання даного типу даних.

У зв’язку з цим, можна дати інше означення константи та змінної.

Якщо елемент даних не може змінювати свого значення, тобто завжди містить одне і те ж саме значення, то відповідний ідентифікатор називається *константою даного типу.* Якщо елемент даних певного типу може змінювати своє значення під час виконання програми, то ідентифікатор, що зв’язаний з цим елементом даних називається змінною відповідного типу. *Значення змінної* – це елемент даних, з якими ця змінна пов’язана.

Отже, у програмах змінна характеризується такими ознаками: *іменем, типом* і *значенням.*

Типи величин у PYTHON

У мові Python усі дані є об’єктами. Імена змінних є лише посиланнями на ці об’єкти. Типи даних можна поділити на основні (вбудовані) та типи, які одержуємо при відключенні додаткових бібліотек.

До основних типів даних можна віднести:

1. None (невизначене значення змінної).
2. Логічний тип даних (bool).
3. Числові типи даних :
    1. int – ціле число;
    2. float – дійсне число;
    3. complex – комплексне число.
4. Списки :

    1. list – список;
    2. tuple – кортеж;
    3. range – диапазон.
5. Рядки:
    1. str.
6. Бінарні списки:
    1. bytes – байти
    2. bytearray – масиви байтів
    3. memoryview – специальні об’єкти для доступу до внутрішнім даним
7. Множини:
    1. set – множина;
    2. frozenset – незмінювана множина.
8. Словники:
    1. dict – словник.

Опис змінних

Як було сказано раніше, у мові Python усі дані є об’єктами. Кожен об’єкт має такі атрибути:

- ідентифікатор (що однозначно ідентифікує об’єкт у пам’яті (адреса об’єкта));
- значення;
- тип.

Імена змінних є лише посиланнями на ці об’єкти. Ідентифікатор змінної повинен відповідати правилам:

- може складатися з літер, символу підкреслення (_), цифр;
- не може починатися з цифр;
- не повинен співпадати з ключовими (зарезервованими) словами та стандартними ідентифікаторами;
- не може починатися і закінчуватися двома символами підкреслювання.

Правила оформлення ідентифікаторів можна знайти за посиланням: https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html#id19

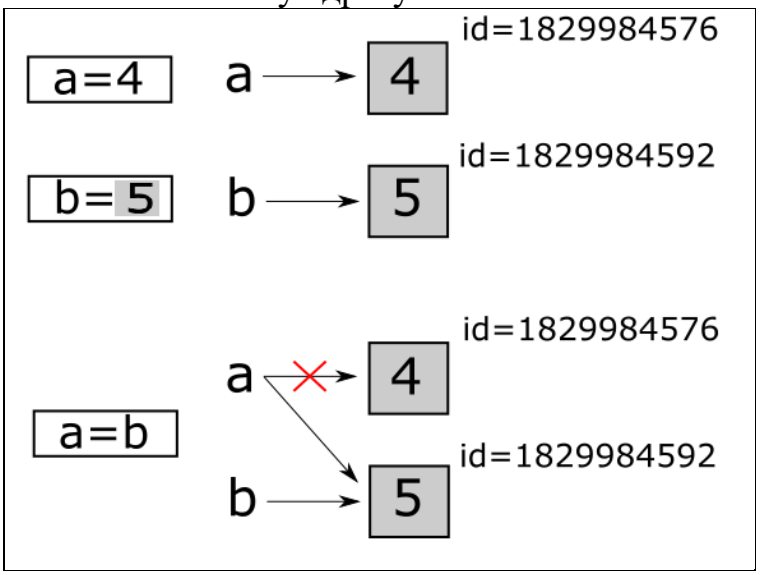
Оператор присвоєння “ = ” зв’язує посилання на об’єкт з об’єктом, який знаходиться в пам’яті

Загальна форма опису змінної	Приклад

<ідентифікатор змінної> = <значення>	`a = 4`
	`b = 5`
	`d = “Hello”`

При виконанні оператора присвоєння можна виділити такі кроки:

1) обчислюється вираз, що знаходиться справа від знаку “=” (це значення має певну адресу в оперативній пам’яті);
2) адреса обчисленого значення зберігається у змінній, яка знаходиться зліва (при цьому:
    - якщо такої змінної ще не існує, то вона буде створена;
    - якщо змінна з вказаним іменем вже існує, то змінюємо її значення на нову адресу обчисленого значення.



Опис констант

Загалом констант як таких не передбачено. Але за прикладом інших мов програмування константи умовно можуть бути описані як змінні, але з ідентифікаторами, які складаються лише з великих літер і знаків нижнього підкреслювання.

Загальна форма опису змінної	Приклад
<ідентифікатор константи>= <значення>	`RATE = 4` `MAX_VALUE = 5`

Саме лише назва пояснює іншим програмістам, що це константа і її не можна змінювати.

Перетворення типів

Розрізняють два види перетворень змінних – *явне* та *неявне*.

Неявне перетворення типів використовується в тому випадку, коли дане перетворення є «природним». Наприклад, перетворюється величина типу int в величину типу float.

Приклад.

```
f= 2 + 3.5 // int + float = результат типу float
```

Або ж, наприклад, при порівнянні значень 2 (ціле число int) і 2.0 (дійсне число float) обидва будуть приведені до дійсного типу. Таке перетворення є

природним, оскільки обидва типи є числовими, причому цільовий тип має більший діапазон представлення. При проведенні неявного перетворення немає необхідності вказувати цільовий тип (тип до якого здійснюється перетворення). Тип результату при цьому визначається типом найбільш «складного» із значень, що входять до цього виразу.

Явне перетворення типів вимагає використання спеціальних функцій. Розглянемо деякі з них

Функція	Призначення	Приклад
`float`	перетворення рядка в дійсне число	>>> float("3.5") 3.5

`int`	перетворення рядка в ціле число	>>> int("24")
		24
		>>> int(3.5)
		3

`str`	перетворення у рядок	>>> str(45.78) '45.78'
`chr`	повертає символ з вказаним кодом	>>> chr(65) 'A'
`ord`	повертає код символа	>>> ord('A') 65

Введення/виведення даних у консольних програмах

Виведення даних

Для виведення даних може бути використана функція `print`

```
print( список значень, sep=символ розділювач, end=кінцевий символ )
```

Загальна форма	Приклад	На екрані
print(список значень) (за замовчуванням значення у списку відділяються пробілами)	print("Hello", 25)	Hello 25
	print(18, 25, 10)	18 25 10
	print("Загальна сума = ", 10)	Загальна сума = 10

Необов'язковий параметр `sep` використовується у випадку, коли потрібно змінити символ розділювач при виведенні з пробіла на якийсь інший.

Загальна форма	Приклад	На екрані
print(список значень , **sep**=символ розділювач **)**	print("Total",25,sep='-')	Total-25
	print(18, 25, **sep=','**)	18,25

За замовчуванням після виведення інформації за допомогою print курсор переводиться на наступний рядок (тому, що за замовчуванням кінцевий символ у параметрі end='\n'). Ми можемо змінити значення цього необов'язкового параметра end.

Загальна форма	Приклад	На екрані
print(список значень , **sep**=символ розділювач)	print(11,22,end=' ') print(33,44)	11 22 33 44

Форматований вивід даних

Форматований вивід даних дозволяє здійснювати контрольований вивід даних. Наприклад, можемо вказати кількість позицій для виведення, кількість знаків після коми та ін. Такий вивід здійснюється з використанням метода str.format

```
print( "рядок-шаблон".format (список форматованих значень))
```

Використання цього методу дозволяє здійснити вивід ґрунтуючись на рядках шаблонах (рядках, у яких у певних зазначених місцях (у фігурних дужках вказуємо номер значення) вставляються відповідні значення із списку виводу).

Приклад.

Команда	print("Price ={0}, total={1}" . format(5, 78))
Результат	Price =5, total=78

Приклад.

Команда	print("a=({0}; {1}; {2})".format(11, 22,33))
Результат	a=(11; 22; 33)

Розглянемо приклади форматованого виводу числових даних.

Форматований вивід цілих чисел

При виведенні цілочислових даних є можливість зазначити кількість позицій для виведення.

```
{номер значення : кількість_позицій d}
```

Літера «**d**» в кінці означає, що сюди буде вставлено ціле число. Якщо у цілому числі при цьому кількість цифр є меншою, то інші позиції будуть заповнюватися пробілами.

Приклад.

Команда	`print("a={0:5d}".format(11))`
Результат	`a=    11  #оскільки в 11 лише 2 символи, то додано 3 пробіли`

Форматований вивід дійсних чисел

При виведенні дійсних чисел є можливість зазначити загальну кількість позицій для виведення та кількість знаків після коми

```
{номер_значення : кількість_позицій . кількість_знаків_після_коми f}
```

Літера «**f**» в кінці означає, що сюди буде вставлено дійсне число. При цьому:

- якщо у числі при цьому кількість символів є меншою, то інші позиції будуть заповнюватися пробілами;
- якщо після коми більше символів, то значення для виведення буде округлено до вказаної кількості знаків після коми;
- якщо знаків після коми менше, то вони доповняються нулями.

Приклад.

Команда	`print("a={0:7.2f}".format(35.78543))`
Результат	`a=  35.79`

Приклад.

Команда	`print("a={0:7.2f}".format(35.7))`
Результат	`a=  35.70`

Якщо загальна кількість позицій для виведення є наважливою, то можна вказати тільки кількість знаків після коми.

Приклад.

Команда	`print("a={0:.2f}".format(35.3563))`
Результат	`a=35.36`

Введення даних

Введення даних може бути здійснено за допомогою функції `input`

```
input(" текст повідомлення ")
```

`Текст повідомлення` – це текст, який виводиться під час введення і пояснює користувачу, які саме дані необхідно ввести. Результатом введення є введений користувачем рядок тексту. Введення закінчується натисненням на клавішу `ENTER`.

Приклад.

Команди	`user_name=input("Введіть своє ім'я: ")` `print("Привіт {0}". format (user_name))`
Результат	`Введіть своє ім'я: Іван` `Привіт Іван`

Введення цілочислових даних

Оскільки функція `input` дозволяє ввести тільки символьне представлення даних (рядок тексту), то для введення цілочислових даних необхідно виконати перетворення рядка у текст за допомогою функції `int`.

```
int(input("текст повідомлення"))
```

Приклад. Знайти суму двох цілих чисел

Команди	`number_1=int(input("Введіть перше число: "))`
	`number_2=int(input("Введіть друге число: "))`
	`sum=number_1+number_2`
	`print("Сума = {0}". format (sum))`

Результат	`Введіть перше число: 5`
	`Введіть друге число: 7`
	`Сума = 12`

Введення дійсних чисел

Аналогічно до введення цілих чисел, при введенні дійсних чисел результат введення функції `input` необхідно перетворити у дійсне число за допомогою функції `float`.

```
float(input("текст повідомлення"))
```

Приклад. Знайти площу прямокутника

Команди	a=**float**(input("Введіть довжину першої сторони: ")) b=**float**(input("Введіть довжину другої сторони: ")) s=a*b print("Площа = {0:.2f}". format (s))
Результат	Введіть довжину першої сторони: 3.5 Введіть довжину другої сторони: 2.8 Площа = 9.80

АРИФМЕТИЧНІ ТА ЛОГІЧНІ ВИРАЗИ

Вирази

Виразом називають послідовність операцій, операндів і розділових знаків, що задають деякі обчислення. В залежності від значення, яке одержується в результаті цих обчислень, вираз поділяють на арифметичні та логічні вирази.

Арифметичні вирази

Арифметичним виразом називають вираз, в результаті обчислення якого одержуємо числове значення. При цьому можуть використовуватися бінарні та унарні операції арифметичні операції.

Бінарні операції

Операція	Позначення	Приклад
+	додавання	z=x+y
–	віднімання	z=x-y
–	унарний мінус	z= -y
*	множення	z=x*y
/	ділення	z=x/y
//	ділення націло	z=x//y
%	остача від ділення	z=x%y
**	піднесення до степеня	z=x**y

Побітові операції

У мові є можливість здійснювати побітові операції над розрядами аргументів

Операція	Позначення	Приклад

&	побітове «і»	x=10; //x=$1010_{(2)}$
		y=7; //y=$0111_{(2)}$
		z=x&y //
		z=2=$0010_{(2)}$

|	побітове «або»	x=10; //x=$1010_{(2)}$
		y=7; //y=$0111_{(2)}$
		z=x|y
		//z=15=$1111_{(2)}$

^	Побітове «виключаюче або»	x=10; //x=$1010_{(2)}$
		y=7; //y=$0111_{(2)}$
		z=x^y
		//z=13=$1101_{(2)}$

Бінарні операції зсуву

Операція	Позначення	Приклад

<<	зсув розрядів вправо
	(змінна) =(змінна)>>(кільк. розрядів)
		i=4; //i=100(2)
		i=i<<2; // i=16=10000(2

>>	зсув розрядів вліво
	(змінна) =(змінна)<<(кільк. розрядів)
		i=4; //i=100(2)
		i=i>>1; //i=2=10(2))

Операції присвоєння

Якщо при виконанні бінарної операції результат зберігається у змінній, що є першим аргументом, то можна використати так звані операції присвоювання.

Операція	Операція	Аналог з бінарними операціями
+=	i = 5; i += 3; # i=8	i = 5; i=i+3; # i=8
-=	i = 5; i -= 3; # i=2	i = 5; i=i-3; # i=2
*=	i = 5; i *= 3; # i=15	i = 5; i=i*3; # i=15
/=	i = 6; i /= 3; # i=2	i = 6; i=i/3; # i=2
//=	i = 7; i //= 3; # i=2	i = 7; i=i//3; # i=2
**=	i = 2; i **= 3; # i=8	i = 2; i=i**3; # i=8
>>=	i = 5; i >>= 1; # i=2	i = 5; i=i>>1; # i=2
<<=	i = 5; i <<= 1; # i=10	i = 5; i=i<<1; # i=10

&=	i = 5;
	j = 7;
	i &= j; # i=5
		i = 5;
		j = 7;
		i=i&j; # i=5

^=	i = 5;
	j = 7;
	i ^= 1; # i=2
		i = 5;
		j = 7;
		i=i^j; # i=2

|=	i = 5;
	j = 7;
	i |= 1; // i=7
		i = 5;
		j = 7;
		i=i|j; // i=7

Математичні функції

Для використання математичних функцій необхідно підключити модуль `math`

```
import math
```

Наведемо деякі математичні функції

`round(x)`	округлення до найближчого цілого
`round(x, n)`	округлення числа x до n знаків після коми
`floor(x)`	округлення числа вниз (найбільше ціле число, що менше або рівне за дане) floor(1.5) == 1, floor(-1.5) == -2
`ceil(x)`	округлення числа вверх (наймене ціле число, що більша або рівне за за дане) ceil(1.5) == 2, ceil(-1.5) == -1
`abs(x)`	модуль числа
Корені, логарифми	
`sqrt(x)`	квадратний корінь
`log(x)`	логарифм натуральний (виклик log(x, b) дозволяє знайти логарифм за основою b.
`e`	константа e = 2,71828...
Тригонометрія	
`sin(x)`	синус
`cos(x)`	косинус
`tan(x)`	тангенс
`asin(x)`	арксинус
`acos(x)`	арккосинус
`atan(x)`	арктангенс
`degrees(x)`	перетворює кут з радіанної міри у градусну
`radians(x)`	перетворює кут з градусної міри у радіанну
`pi`	константа π = 3.1415...

Питання для самоконтролю

1. *Що називають арифметичним виразом?*
2. *Що називають логічним виразом?*
3. *Які операції може містити арифметичний вираз?*
4. *Які операції може містити логічний вираз?*
5. *Назвіть основні математичні функції, які доступні у мові Python.*
6. *Який пріоритет операцій у мові Python.*

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

1	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Скласти програму для знаходження площі трикутника за формулою Герона.
2	Обчислити площу трикутника, якщо трикутник задано довжинами сторін.
3	Обчислити площу трикутника, якщо трикутник задано довжиною однієї з сторін та висотою, опущеною на неї.
4	Обчислити площу трикутника, якщо трикутник задано двома сторонами та кутом між ними.
5	Обчислити площу та периметр квадрата, якщо задано довжину сторони цього квадрата.
6	Обчислити площу та периметр квадрата, якщо задано довжину діагоналі цього квадрата.
7	Обчислити площу та периметр прямокутника, довжини сторін якого задаються.
8	Обчислити площу та периметр ромба, якщо задано довжину сторін та один з кутів.
9	Обчислити площу та периметр рівнобічної трапеції, для якої задано довжини основ та висоту..
10	Трикутник задається координатами своїх вершин на площині: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Скласти програму для знаходження периметра.
11	Дано два вектори $x, y \in R^3$. Знайти косинус кута між ними.
12	Дано : $x, y \in R^3$. Знайти $z = \langle x, y \rangle x$ де $\langle x, y \rangle$ – скалярний добуток векторів x та y.
13	Дано : $x, y \in R^3$. Знайти $z = 2x + 3y$.
14	Дано два дійсних числа. Знайти суму, добуток, середнє арифметичне та середнє геометричне цих чисел.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.python.org/3/tutorial/index.html.

2. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Чернігів: ФОП Баликіна С.М., 2020. 180 с.

3. Яковенко А. В. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки", спеціалізації "Інформаційні технології в біології та медицині". – Київ : КПІ ім. Ігоря Сікорського, 2018. – 195 с.

ЗМІСТ

Укладачі: к. ф.-м. н., доц. Брила А.Ю.,
ст.викл., Ломага М.М.,
к. ф.-м. н., Вощепинець А.С.

Рецензенти: к.ф.-м.н., доц. Погоріляк О.О.,
к.ф.-м.н., доц. Млавець Ю.Ю.

АРИФМЕТИЧНІ ВИРАЗИ У PYTHON. ЛІНІЙНІ АЛГОРИТМИ

Методичні вказівки до лабораторних робіт з дисципліни «Програмування»