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

ФУНКЦІЇ У PYTHON

Методичні вказівки до лабораторних робіт з дисципліни «Програмування»

Ужгород 2023

Функції у Python. (Методичні вказівки до лабораторних робіт з дисципліни «Програмування»). /Укладачі: А.Ю. Брила, М.М. Ломага, А.С. Вощепинець. – Ужгород, 2023.– 22 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).

ФУНКЦІЇ У PYTHON

Допоміжні алгоритми у Python

Функції є реалізацією допоміжних алгоритмів. Вони є основою структурного підходу до розробки алгоритмів. При цьому складна задача розбивається на більш простіші підзадачі (принцип *декомпозиції*), а кожна із підзадач розв'язується за допомогою відповідної функції.

Розглянемо загальний вигляд опису функції

Загальний вигляд	`def <ім'я функції> (<список формальних параметрів>):` `    <тіло функції (оператори)>`

Приклад	`def get_max(number_1, number_2):`
	`    if number_1>number_2:`
	`        return number_1`
	`    else:`
	`        return number_2`

Список формальних параметрів – це список позначень величини, які є необхідними для роботи функції. Так у наведеному прикладі для знаходження найбільшого серед двох чисел необхідно мати ці два числа `number_1`, `number_2`. Формально, ми могли б позначити ці величини довільними іменами, але на практиці потрібно намагатися позначати величини таким чином, щоб з назви було зрозуміло їх призначення. Зауважимо при цьому, що при розробці функції ми абстрагуємось від конкретних значень, для яких буде викликано дану функцію (застосовуємо принцип *абстракції*). Функцію потрібно розробляти так, щоб її можна було використати для довільних значень формальних параметрів.

У процесі свого виконання функція може обчислювати деякі величини, які є результатами роботи (у даному випадку найбільше серед двох чисел). Результати роботи функції можуть бути повернені у точку виклику за допомогою службового слова `return`

```
return <результат>
```

Якщо функція не повертає результату за допомогою `return`, то вважається, що функція повертає значення `None`, яке у логічному виразі приводиться до значення `False`.

Виклик функції здійснюється за її ім'ям. При цьому у дужках необхідно вказати список фактичних параметрів. Значення цих фактичних параметрів буде скопійовано у відповідні формальні параметри.

Загальний вигляд	<ім’я функції> (<список фактичних параметрів>)

Приклад	# ----- Опис функції -------
	def **get_max**(number_1, number_2):
	    if number_1>number_2:
	        return number_1
	    else:
	        return number_2
	# ------ Виклик фунції -------
	result_1=**get_max(2,6)**
	print("result_1=",result_1)
	a=8
	b=7
	result_2=**get_max(a,b)**
	print("result_2=",result_2)

Слід зазначити, що у Python можна повернути не тільки одне значення, а декілька значень розділених комою (кортеж значень)

Загальний вигляд	def <ім’я функції> (<список формальних параметрів>):
	    . . . . . .
	  return <величина 1>,<величина 2>,. . .,<величина N>

Приклад	Більше число помножити на 2, а менше – поділити на 2
	# ----- Опис функції -------
	def **numbers_processing**(number_1, number_2):
	    if number_1>number_2:
	        number_1 *=2
	        number_2 /=2
	    else:
	        number_2 *=2
	        number_1 /=2
	    return number_1,number_2
	# ------ Виклик фунції -------
	a=8
	b=7
	a , b = number_processing (a,b)
	print("a={0}, b={1}".format(a,b))

Зауважимо, що таким чином можна реалізовувати зміну декількох значень за допомогою функції (у функцію передаємо копію вихідних значень, а потім у основній програмі у змінні записуємо нові (змінені) значення).

Документування функцій

Для вставки довідкової інформації щодо функції необхідно одразу після заголовку функції вставити коментар, поміщений у потрійні лапки (одинарні чи подвійні)

Загальний вигляд	`def <ім'я функції> (<список формальних параметрів>):`
	`  ''' коментар до функції  '''`
	`  <тіло функції (оператори)>`

Приклад	`def get_max(number_1, number_2):`
	`  ''' returns the maximum of two numbers '''`
	`  if number_1>number_2:`
	`  return number_1`
	`  else:`
	`  return number_2`

Дану довідкову інформацію можна отримати шляхом виклику функції `help`

Загальний вигляд	`help (<ім'я функції>)`
Приклад	`help (get_max)`

На екрані	`Help on function get_max in module __main__:`
	
	`get_max(number_1, number_2)`
	`  returns the maximum of two numbers`

Локальні і глобальні змінні

Область видимості змінної – це частина програми, де до цієї змінної можна звернутись. Блок – це послідовність операторів тіла функції або всього файлу. Область видимості змінної обмежується блоком, у якому цю змінну було ініціалізовано деяким значенням а також усіма вкладеними у даний блоками.

Значення, які ініціалізовано в тексті програми (за межами функції) є *глобальними* змінними і можуть бути використані у будь-якому вкладеному блоці (будь-якій функції). Змінні, які ініціалізовані в межах функції є *локальними* змінними. Їх областю видимості є функція, у якій вони були ініціалізовані.

Приклад.

```
a=5                 ← глобальна змінна
def func():
    print(a)        ← звертання до глобальної змінної
    b=3             ← опис локальної змінної
    print(b)        ← виведення значення локальної змінної (3)
func()          ← виклик функції
print(b)        ← ПОМИЛКА !!! Змінної b не існує у даній області видимості
```

Слід також пам’ятати, що у функції можна перевизначити глобальну змінну (описати локальну з таким же іменем, як і глобальна). У цьому випадку значення глобальної змінної буде недоступним і без попереднього присвоєння локальній змінній значення цієї локальної змінної використовувати не можна.

Приклад.

```
a=5                 ← глобальна змінна
def func():
    print(a)        ← Помилка!!! Звертання до локальної змінної,
                      яку ще не ініціалізовано
    a=3             ← опис локальної змінної, яка перевизначає глобальну
```

Слід відмітити, що не важливо, у якому саме місці функції міститься ініціалізація локальної змінної. Можливо ініціалізація знаходиться у фрагменті функції, який ніколи не буде виконано. Тут головним є фактор наявності ініціалізації новим значенням.

Приклад.

```
a=5                 ← глобальна змінна
def func():
    print(a)        ← Помилка!!! Звертання до локальної змінної,
                      яку ще не ініціалізовано
    if False:
        a=3         ← опис локальної змінної, яка перевизначає глобальну
```

Зауважимо, що формальні параметри також вважаються локальними змінними функції.

Значення за замовчуванням

При описі функції можна задавати так звані «значення за замовчуванням». Це значення, які використовуються у випадку, коли під час виклику не було вказано значення відповідного фактичного параметру. Формальні параметри зі значенням за замовчуванням обов'язково повинні бути в кінці списку формальних параметрів після параметрів без значень за замовчуванням (параметрів, які обов'язково потрібно вказати).

Загальний вигляд	`def <ім'я функції> ( <обов'язкові форм. парам.>,`
	`  <парам.1> = <знач.1>,`
	`  <парам.2> = <знач.2>, . . .):`
	`  <тіло функції (оператори)>`

Приклад	`def func(a, b=0, c=0):`
	
	`  return a+b+2*c`

Приклад виклику	`print (func(2))` ← Результат 2 (a=2, **b=0, c=0**) `print (func(2,7))` ← Результат 9 (a=2, **b=7, c=0**) `print (func(2,7,4))` ← Результат 17 (a=2, **b=7, c=4**)

Зауважимо, що значення параметрів можна вказувати і з використанням іменованого присвоєння формальним параметрам фактичних. Для цього під час виклику функції необхідно явно вказати формальний параметр, якому присвоюється значення.

Загальний вигляд	`<ім'я функції>(<форм.парам.1> = <факт.парам.1>,`
	`  <форм.парам.2> = <факт.парам.2>, ...):`
	`  <тіло функції (оператори)>`

Приклад	`#----- Опис функції ----`
	`def func(a, b=0, c=0):`
	
	`  return a+b+2*c`
	`#------ Виклик функції ----`
	`print (func(2,c=4))` ← a=2, b=0, c=4
	`print (func(c=4,a=2))` ← a=2, b=0, c=4
	`print (func(b=8,a=2))` ← a=2, b=8, c=0

Передача довільної кількості параметрів

У Python також є можливість описати функції з довільною кількістю параметрів (під час виклику можна передавати довільну кількість фактичних параметрів). Для цього необхідно описати формальний параметр, перед яким стоїть знак «*».

Загальний вигляд	<ім'я функції>(. . . інші формальні параметри . . ., * <формальний_параметр_кортеж> , . . . інші формальні параметри . . .): <тіло функції (оператори)>

Приклад.

```
#------ Опис функції з довільної кількістю параметрів -----
def get_sum(*arguments):
    sum=0
    for x in arguments:
        sum+=x
    return sum
#------ Виклик функції з довільної кількістю параметрів -----
print (get_sum(2))                ← Результат 2
print (get_sum(2,7))              ← Результат 9
print (get_sum(2,3,67,5))         ← Результат 77
print (get_sum(11,4,22,6,45,76)) ← Результат 164
```

Зауважимо, що опису такого формального параметра можуть бути описані і інші формальні параметри.

Приклад. Створити функцію для підрахунку кількості входжень деякого заданого елемента (`search_element`) у послідовності елементів (`*arguments`).

```
#--------------- Опис ---------------
def count_of_elements(search_element, *arguments, last):
    count=0
    for x in arguments:
        if search_element==x:
            count+=1
    return count
#----------------- Виклик -------------
print (count_of_elements(2,4,2,6,5,2,3, 8))
print (count_of_elements('a','s', 'd','a','f','a','a'))
```

До того ж він може бути не тільки в кінці списку а й у середині списку формальних параметрів. При цьому під час виклику значення формальних параметрів, які описано після формального параметра – кортежа (для довільної кількості фактичних параметрів) необхідно задавати явно (через імена), або ж такі формальні параметри повинні мати значення за замовчуванням.

Передача довільної кількості іменованих параметрів

Для передачі довільної кількості іменованих формальних параметрів (у вигляді словника) необхідно описати формальний параметр, перед яким необхідно написати дві зірочки.

Загальний вигляд	<ім'я функції>(. . . інші формальні параметри . . ., ****** <формальний_параметр_словник> , . . . інші формальні параметри . . .): <тіло функції (оператори)>

Приклад	У заданому рядку вказані букви замінити на нові
	#-------- Опис функції ------------
	def g(row, ****kwargs**):
	    for letter in **kwargs**:
	        row=row.replace(letter,**kwargs**[letter])
	    return row
	#-------- Виклик функції ------------
	print(g("Hello", **o='y', e='o',l='b'**))
	#-------- Результат ------------
	Hobby

Приклад 2	#-------- Опис функції ------------ def g(****kwargs**): print(**kwargs**) #-------- Виклик функції ------------ g(a=11,s="Hello",m=3.5) #-------- Результат ------------ {'a': 11, 's': 'Hello', 'm': 3.5}

Використання одночасно і формального параметра-кортежа і формального параметра-словника

При описі функцій можна одночасно використовувати як формальний параметр-кортеж, так і формальний параметр-словник. При цьому можна у функцію передавати довільну кількість як неіменованих, так і іменованих параметрів.

Загальний вигляд	<ім'я функції>(. . . інші формальні параметри . . ., *** <формальний_параметр_кортеж>** , . . . інші формальні параметри . . ., **** <формальний_параметр_словник>** , . . . інші формальні параметри . . .): <тіло функції (оператори)>
Приклад	#----------- Опис функції ----------- def g(a, b, ***args**, name='default', ****kwargs**): print(a, b, args, name, kwargs) #----------- Виклик функції --------- g(1,2,3,4,5,6,7,name="Hello", s=11,w="World") #----------- Результат -------------- 1 2 **(3, 4, 5, 6, 7)** Hello **{'s': 11, 'w': 'World'}** ***args** ****kwarg**

Приклад. Визначити скільки з переданих елементів належать до наперед визначених множин

Приклад	#-------- Опис функції ------------ def g(*users_keys, ****users_groups**): return {group: len(set(users_groups[group])&set(users_keys))) for group in users_groups} #-------- Виклик функції ------------ print(g(11,2,3,44,12,33,**guests=(1,2,3,4,5),** **managers=(11,22,33,44,55)**)) #-------- Результат ------------ {'guests':2, 'managers': 3)}

ЛЯМБДА ВИРАЗИ

За допомогою лямбда виразів маємо можливість робити скорочені описи функцій. Особливо це є корисним при передачі функції як аргументу (наприклад при сортуванні списків).

Загальний вигляд	`lambda <список аргументів> : <результат функції>`

Приклад	`Функція для знаходження суми двох чисел`
	`#-------- Опис функції ------------`
	`f= lambda a,b:a+b`
	
	`#-------- Виклик функції ------------`
	`print(f(3,7))`
	
	`#-------- Результат ------------`
	`10`

Приклад 2	`Упорядкувати елементи списку чисел за останньою цифрою`
	`#----- Використання функції як аргументу ----------`
	`listV=[22,93,12,45,61]`
	`listV.sort(key= lambda item:item%10)`
	`print(listV)`
	`#-------- Результат ------------`
	`[61, 22, 12, 93, 45]`

РЕКУРСІЯ

Рекурсія − це такий спосіб організації обчислювального процесу, за якого функція звертається сама до себе. Такі звернення називаються *рекурсивними викликами*, а функція, що містить рекурсивні виклики, − *рекурсивною*.

Рекурсію використовують у ситуаціях, коли легко звести вихідну задачу до задачі того ж виду, але з іншими вихідними даними. Найпростішим прикладом такої задачі може стати обчислення факторіала. Так обчислення факторіала може бути здійснене у відповідності до наступного рекурентного правила:

$$n! = \begin{cases} (n-1)! \cdot n, \text{ якщо } n > 0; \\ 1, \text{ якщо } n = 0. \end{cases}$$

Реалізуємо це рекурентне правило у вигляді рекурсивної функції:

Приклад	`Функція для знаходження суми двох чисел`
	`#-------- Опис функції ------------`
	`def factorial(n):`
	`    if n==0:`
	`        return 1`
	`    else:`
	`        return factorial(n-1)*n`
	`#-------- Виклик функції ------------`
	`print(factorial(5))`
	`#-------- Результат ------------`
	`120`

Будь-яке рекурентне правило, яке визначає деяку рекурсію повинно містити умову зупинки, що називається *умовою завершення рекурсії*. Для випадку задачі обчислення факторіала *n*! умовою завершення рекурсії є умова *n*=0.

Питання для самоконтролю

1. *Що таке функція у Python?*
2. *Як створити функцію у Python?*
3. *Які типи аргументів можуть бути у функції?*
4. *Як викликати функцію у Python?*
5. *Як передати аргументи у функцію?*
6. *Що таке аргументи за замовчуванням у функції?*
7. *Як обробити виняткові ситуації у функції?*
8. *Що таке рекурсивна функція і як її використовувати?*
9. *Які є вбудовані функції у Python?*
10. *Як створити анонімну функцію (lambda-функцію)?*
11. *Які є особливості області видимості змінних у функціях?*

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Завдання 1.

1	За даними дійсними числами a і b обчислити $u = f(a,b) + f(2,a) + 2,$ де $f(x,y) = \begin{cases} x^3 + \sqrt{x^2 + y^4}, x > y, \\ \dfrac{x^2 - 2x + \sqrt{x}}{\sqrt[5]{x^3}}, \textit{ інакше.} \end{cases}$
2	За даними дійсними числами a і b обчислити $u = f(a,b) + f(2,a) + 2,$ де $f(,y) = \begin{cases} x^3 + \sqrt{x^2 + y^4}, x > 0\, i\, y > 0, \\ \dfrac{x^2 - 2x + \sqrt{x}}{\sqrt[5]{x^3}}, x > 0\, i\, y < 0, \\ \sin(x * y), \textit{ інакше.} \end{cases}$
3	За даними дійсними числами a і b обчислити $u = f(a) + f(2) + f(b),$ де $f(x) = \begin{cases} \sum_{i=1}^{5} x^i, x > 0, \\ \prod_{i=1}^{5} x^i, \textit{ інакше.} \end{cases}$
4	За даними дійсними числами a і b обчислити $u = \min\{f(a), f(b)\},$ де $f(x) = \begin{cases} \sin x + \sin x^2 + ... + \sin x^6, x > 3, \\ -\cos x + \cos\cos x + ... + (-1)^5 \underbrace{\cos ... \cos}_{5} x, \textit{ інакше.} \end{cases}$
5	За даними дійсними числами a і b обчислити $u = \max\{f(a), f(b)\},$ де $f(x) = \begin{cases} \sin x + \sin^3 x + \sin^5 x + ... + \sin^{17} x, x > 3, \\ 15 + tgx + tg\, tgx + ... + \underbrace{tg ... tg}_{5} x, \textit{ інакше.} \end{cases}$
6	За даними дійсними числами a і b обчислити $u = \max\{f(a), f(b)\} - f(3),$ де $f(x) = \begin{cases} 1 + \dfrac{x}{2!} + \dfrac{x^2}{3!} + .. + \dfrac{x^8}{8!}, x < 0, \\ 15 + \sqrt{\cos^6 x}, \ 0 \le x \le 5, \\ \min\{1, 2 * \sin(x)\}, \ x > 5. \end{cases}$

7	Дано дійсні числа x,y,z . Обчислити $u = \dfrac{\max\{x, z\} + \max\{x + y, xy\}}{\max^2\{x + y, xy\}}$.
8	Дано дійсні числа x,y,z . Обчислити $u = \dfrac{\max\{x, y, z\} + \max\{x + y, xy, 4z\}}{\max\{\max^2\{x + y, xy, x^2\}, 7, z^2\}}$.
9	Використовуючи підпрограму для знаходження коренів квадратного рівняння, знайти розв'язок наступної системи рівнянь $\begin{cases} ax^2 + 2x + 7 = 0, \\ bx^2 - 5x + 2 = 0. \end{cases}$ Числа $a, b \in R$.
10	Використовуючи підпрограму для знаходження найбільшого спільного дільника (НСД), знайти значення виразу S=(НСД(a,b)+ НСД(a,4))+ НСД(24,b)
11	Використовуючи підпрограму для знаходження коренів квадратного рівняння, знайти розв'язок наступної системи рівнянь $\begin{cases} ax^2 + 2x + 7 = 0, \\ bx^2 - 5x + 2 = 0. \end{cases}$ Числа $a, b \in R$.
12	Дано три дійсних числа: a, b, c. Використовуючи підпрограми для знаходження максимального та мінімального серед двох дійсних чисел знайти $\max(a,b) + (\max(a,b) + \min(b,c))^2$.
13	Обчислити значення виразу $f(2, a, 4) - 5 * f(a, b, -c)$, де $f(x,y,z) = \begin{cases} \lg(x + y - z), & \text{якщо } x + y - z > 10, \\ (\lvert x + y \rvert + 1)^z, & \text{якщо } z - 10 < x + y < z + 1, \\ x^2 + y^2 - z^3, & \text{якщо } x + y - z = 1, \\ \cos^2 x + \sin y - e^{2z+1}, & \text{в інших випадках.} \end{cases}$
14	Обчислити значення виразу $f(4) + 2f(a) - f(b)$, де $f(x) = \begin{cases} 0, & x < 23; \\ 1, & x = 23; \\ -1, & x > 23. \end{cases}$

Завдання 2.

1	Використовуючи підпрограму для знаходження скалярного добутку, обчислити значення виразу $s = 2\langle a,b\rangle$-$3\langle a,c\rangle$ де $a,b,c \in R^n$, $\langle x,y\rangle$ – скалярний добуток векторів.
2	Використовуючи підпрограму наближеного знаходження визначеного інтегралу за формулою лівих прямокутників, обчислити значення виразу $s=\int_a^3 \sqrt{4x+\sin\sqrt{x^3}}\,dx+\int_a^b \sqrt{4x+\sin\sqrt{x^3}}\,dx$
3	Використовуючи підпрограму наближеного знаходження визначеного інтегралу за формулою правих прямокутників, обчислити значення виразу $s=\int_a^3 \sqrt{4x+\sin\sqrt{x^3}}\,dx+\int_a^b \sqrt{4x+\sin\sqrt{x^3}}\,dx$
4	Використовуючи підпрограми для додавання векторів та множення вектора на число, знайти вектор $c = a - 3 * b + 2c$, де $a,b,c \in R^n$.
5	Використовуючи підпрограму визначення паралельності двох прямих на площині, визначити, скільки взаємно паралельних пар прямих є серед вказаних n прямих: $A_i x + B_i y + C_i = 0,\ i = 1,2,...,n$.
6	Використовуючи підпрограму визначення перпендикулярності двох прямих на площині, визначити, скільки взаємно перпендикулярних пар прямих є серед вказаних n прямих: $A_i x + B_i y + C_i = 0,\ i = 1,2,...,n$.
7	Два трикутники задано координатами вершин. Використовуючи підпрограму визначення належності точки внутрішності трикутника, з'ясувати, чи лежить один з трикутників у середині іншого.
8	Трикутник задано координатами своїх вершин на площині. Використовуючи підпрограму для знаходження кута між векторами на площині, встановити тип трикутника (гострокутний, прямокутний, тупокутний).
9	Дано послідовність натуральних чисель $a_1, a_2, ..., a_n$. Використовуючи підпрограму, яка дозволяє встановити, чи є послідовність із чотирьох чисел арифметичною прогресією, знайти кількість послідовно розміщених четвірок чисел, які утворюють арифметичну прогресію.
10	Дано послідовність натуральних чисель $a_1, a_2, ..., a_n$. Використовуючи підпрограму, яка дозволяє встановити, чи є послідовність із чотирьох чисел геометричною прогресією, знайти кількість послідовно розміщених четвірок чисел, які утворюють геометричну прогресію.
11	Використовуючи відповідні підпрограми, з'ясувати, що є більшим, середнє арифметичне чи середнє геометричне чисел $a_1, a_2, ..., a_n$.
12	Дано послідовність натуральних чисель $a_1, a_2, ..., a_n$. Використовуючи відповідні підпрограми знаходження суми та добутку цифр, знайти натуральне число, у якого найбільша сума цифр та найменший добуток цифр.
13	Дано послідовність натуральних чисель $a_1, a_2, ..., a_n$. Використовуючи підпрограму знаходження найбільшої та найменшої цифри, знайти число, у яке містить найбільшу цифру та число, яке містить найменшу цифру.
14	Використовуючи підпрограму для знаходження n-тового числа Фібоначчі. Обчислити значення виразу $s = f_3 + f_8$, де f_i – i-тове число Фібоначчі.

Завдання 3. Рекурсії

1	Використовуючи підпрограму для знаходження n-тового числа Фібоначчі. Обчислити значення виразу $s = f_3 + f_8$, де f_i – i-тове число Фібоначчі.
2	Використовуючи відповідну підпрограму знаходження g_i, обчислити значення виразу $$s = g_7 + g_9,$$ де $g_i = \sin(g_{i-1} + \cos(g_{i-2})),\ g_0 = 9, g_1 = 35.$
3	Нехай $x_0 = 1,\ x_i = x_{i-1} + 2i$, де $i = 1,2,\ldots$. Визначити x_{10}.
4	Нехай $x_0 = x_1 = 1,\ x_i = x_{i-1} + 2x_{i-2}$, де $i = 2,3,\ldots$. Визначити x_n.
5	Нехай $x_0 = x_1 = 1,\ x_i = x_{i-1} + x_{i-2}$. Визначити x_n.
6	Нехай $x_0 = 0, x_1 = 9,\ x_i = 2x_{i-1} + 3x_{i-2}$. Визначити x_n.
7	Нехай $x_0 = 0, x_1 = x_2 = 9,\ x_i = x_{i-1} + x_{i-2} + x_{i-3}$. Визначити x_n.
8	Нехай $x_0 = 0, x_1 = x_2 = 9,\ x_i = x_{i-1} + 4x_{i-3}$. Визначити x_n.
9	Нехай $x_0 = 0, x_1 = 7,\ x_i = x_{i-1}(1 + x_{i-2})$. Визначити x_n.
10	Нехай $x_0 = 0, x_1 = x_2 = x_3 = 7,\ x_i = \dfrac{x_{i-1}(1 + x_{i-2}) + x_{i-3}}{x_{i-4}}$. Визначити x_n.
11	Нехай $x_0 = 0, x_1 = x_2 = x_3 = 7,\ x_i = \dfrac{x_{i-1} + 4x_{i-2}(1 + x_{i-2}) + x_{i-3}}{\sqrt{x_{i-4}}} + x_{i-4}$. Визначити x_n.
12	Нехай $x_0 = x_1 = 1,\ x_i = \sin x_{i-1} + \dfrac{x_{i-2}}{\cos(x_{i-1})}$. Визначити x_n.
13	Нехай $x_0 = 2, x_1 = 1,\ x_i = \cos x_{i-1}^{x_{i-2}}$. Визначити x_n.
14	Нехай $x_0 = 2, x_1 = x_2 = 3,\ x_i = 7x_{i-1} - x_{i-2} * x_{i-3}$. Визначити x_n.

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.

ФУНКЦІЇ У PYTHON

Методичні вказівки до лабораторних робіт з дисципліни «Програмування»