Сергій Матвійчук

АБЕТКА ПРОГРАМУВАННЯ на Python

збірник задач вміщує пояснення алгоритмів та коди програм до 200 окремих задач з сайту **e-olymp.com**

2020

Зміст

Близько року тому на сайті **e-olymp.com** з'явилась добірка завдань початкового рівня під загальною назвою "Абетка програмування" це номери 8800..9000. Коли кількість задач на сайті сягає декількох тисяч, то знайти потрібний набір, тим більш початкового рівня досить проблематично. Задачі з "Абетки" розміщені методологічно і відповідно до шкільної програми, щоб допомогти тим, хто робить перші кроки в "школі молодого бійця". Тисячі зарахованих і незарахованих спроб на цих завданнях від користувачів сайту свідчать про популярність і корисність завдань.

Завдання "Абетки" створювались без попередньої підготовки, тобто умови задач, тести, коди для перевірки писалися прямо на сайт. Може дещо поспішно, але так вже вийшло. Тому постала необхідність в посібнику, де можна прочитати умови завдань в хронологічному порядку, мати необхідний довідковий матеріал та для прикладу розв'язання деяких з них.

Найближчим часом всі завдання "Абетки" були перенесені у Word і до Вашої уваги посібник або збірник задач "Абетка програмування" тільки з "вогню-полум'я". Записані всі двісті задач в одному з двох варіантів:

- номер з сайту, назва, умова та приклад тесту;
- номер з сайту, назва, умова, приклад тесту та коротке пояснення і код програми.

Задачі розбиті (майже завжди) на групи по дванадцять завдань, групи відповідно об'єднані в розділи "Лінійні програми**"**, "Розгалуження**"** "Цикл **while**", "Цикл **for**", "Масиви", "Рядки". Кожен розділ має коротенький теоретичний вступ та довідковий матеріал по **Python** – скопійовано з книги «Практикум програмування Python / C++ на e-olymp.com». Нічого зайвого, мінімальна кількість символів і аркушів, їх до речі 75. Розраховано, щоб вчитель мав змогу роздрукувати посібник в необхідній кількості для учасників навчального процесу.

Користуватися посібником можна в декількох варіантах:

- просто розв'язувати завдання, не використовуючи сайт **e-olymp**;
- розв'язувати завдання, використовуючи для перевірки програм сайт;
- отримати статус керівника груп на сайті **e-olymp** , проводити підсумкові і тренувальні роботи автоматизованими засобами, що особливо актуально в наш нелегкий час дистанційного навчання і карантину.

Бажаю успіхів. Надіюся, що ця книга буде Вам корисною.

Лінійні програми

Розпочнемо з вправ, які традиційно називають лінійними програмами – тобто не потребують особливих методів і прийомів програмування, а мають три чітко окреслених блоки, які послідовно виконуються в програмі – це ввід даних, обчислення та вивід результатів. Зазвичай, в таких програмах проблематичною є друга, тобто математична частина. Зрозуміло, що кожного разу будуть різні формули, їх можна придумати використовуючи свій математичний досвід або глянути в Інтернеті. Потрібно не забувати, що Ви розробник програми і невірно вибрана математична модель буде давати неправильні результати або не працювати взагалі. Короткі коментарі по інструкціях в лінійних програмах, тільки тези:

- для друку чисел, тексту і значень змінних в мові Python використовується оператор **print**();
- щоб вивести текст, наприклад **Hello!**, його записують в круглих дужках і в лапках після **print**, ось так **print**("**Hello!**");
- щоб вивести числове значення, в круглих дужках після **print**() записується константа, ім'я змінної або арифметичний вираз;
- щоб вивести декілька об'єктів, їх записують аргументами в інструкції **print**() через кому;
- імена змінних в Python відповідають загальноприйнятим правилам в мовах програмування, тобто позначаються буквами і цифрами (перша буква), але великі і маленькі букви розрізняються;
- типи змінних попередньо прописувати не потрібно;
- під час виконання програми одна та ж сама змінна може приймати значення різних типів в залежності від інструкцій;
- для перетворення типів значень змінних використовуються функції **int**(), **float**(), **bool**(), **str**();
- ввід даних в Python виконує оператор **input**(), причому введене значення рахується текстовим рядком;
- для введення чисел використовуються інструкції **int**(**input**()) для цілих або **float**(**input**()) для дійсних;
- вирази в командах присвоювання і друку використовують загальноприйнятий порядок дій, арифметичні операції і математичні функції;
- схема лінійного алгоритму - перш за все потрібно ввести дані, потім обчислити і тільки на кінець вивести результати.

Якщо поки не все зрозуміло, не біда. Можете сміливо приступати до розгляду конкретних задач, а тези перечитаєте ще раз пізніше.

Основні арифметичні операції	
додавання	`+`
віднімання	`-`
множення	`*`
ділення	`/`
піднесення до степені	`**`
ділення націло	`//`
остача від ділення націло	`%`
Основні арифметичні функції	
модуль аргументу **x**	`abs(x)`
перетворення рядка **x** до цілого	`int(x)`
перетворення рядка **x** в системі числення **n** до цілого	`int(x,n)`
перетворення аргументу до дійсного	`float(x)`
округлення аргументу до **n** знаків після коми	`round(x,n)`
перетворення цілого числа в двійковий рядок	`bin(x)`
перетворення цілого числа в вісімковий рядок	`oct(x)`
перетворення цілого числа в шістнадцятьковий рядок	`hex(x)`
повертає частку і остачу від ділення **a** на **b**	`divmod(a,b)`
повертає **a** в степені **b**	`pow(a,b)`
Основні команди в лінійних програмах	
ввести текстовий рядок з консолі в	`x = input()`
ввести ціле число в **X**	`x = int(input())`
ввести дійсне число в **X**	`x = float(input())`
ввести декілька цілих чисел в **X**, **Y**	`x,y = map(int,input().split())`
ввести декілька дійсних чисел в **X**, **Y**	`x,y = map(float,input().split())`
вивести значення **X**	`print(x)`
вивести значення **X**, **Y**	`print(x,y)`

Лінійні програми. Початок.

8800 Hello, Python!

Виведіть повідомлення **Hello,Python!**

Hello, Python!

Текст виводить команда **print**, потрібно уважно ставити дужки і лапки.	`print("Hello, Python!")`

8801 Наступне число

Прочитати ціле число та вивести наступне за ним число.

10	11

Програма читає з консолі ціле число **n** і виводить на екран значення **n+1**.	`n=int(input())` `print(n+1)`

8802 Попереднє число

Програма повинна прочитати з консолі ціле число та вивести попереднє до нього число.

10	9

8803 Хто крайній?

Оленка є передостанньою у списку учнів свого класу. Знайти кількість учнів, якщо відомо номер Оленки.

20	21

8804 Сума двох цілих

На вході програми маємо два цілих числа, кожне в окремому рядку. На вихід потрібно подати суму заданих чисел.

12 9	21

Програма читає з консолі два цілих числа **n** та **m**, кожне з окремого рядка та виводить на екран їх суму.	`n=int(input())`
	`m=int(input())`
	`print(n+m)`

8805 Сума двох цілих 2

На вході програми маємо два цілих числа, записані в одному рядку через пропуск. На вихід потрібно подати суму заданих чисел.

12 9	21

Аналогічно попередньому завданню. Варто звернути увагу на особливості введення двох цілих чисел, записаних в одному рядку через проміжок.	`a,b=map(int,input().split())` `print(a+b)`

8806 Кількість учнів

В класі навчається **a** хлопчиків і **b** дівчат. Скільки всього учнів в класі?

12 9	21

8807 Протилежне число

Програма повинна прочитати з консолі ціле число та вивести протилежне до нього число.

10	-10

8808 Різниця двох цілих

На вході програми маємо два цілих числа, кожне в окремому рядку. На вихід потрібно подати різницю між першим і другим числом.

21 9	12

8809 Марафон

Змагання з бігу розпочали **a** учасників, але **b** з них зійшли з дистанції. Скільки бігунів фінішували?

12 9	3

8810 Шкільний концерт

На шкільному концерті **a** учнів співали, **b** - танцювали, а **c** – співали й танцювали. Скільки було учасників всього?

13 9 5	17

Від суми двох перших чисел потрібно відняти третє, бо значення **c** врахували двічі.	`a,b,c=map(int,input(`
	`).split())`
	`print(a+b-c)`

8811 Добуток двох цілих

На вході програми маємо два цілих числа, записані в одному рядку через пропуск. На вихід потрібно подати добуток заданих чисел.

3 7	21

Лінійні програми. Обчислення.

8812 Периметр і площа

Знайти периметр і площу прямокутника, якщо відомі довжини його сторін.

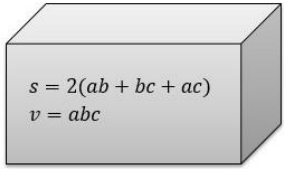
$$p = 2(a + b)$$
$$s = ab$$

3 7	20 21

8813 Площа поверхні та об’єм

Знайти площу поверхні та об'єм прямокутного паралелепіпеда, якщо відомі його виміри.

2 3 4	52 24



Алгоритм очевидний:
– вводимо a, b, c
– обчислюємо s, v
– виводимо s, v
	`a,b,c=map(int,input().split())`
	`s=2*(a*b+b*c+a*c)`
	`v=a*b*c`
	`print(s,v)`

8814 Периметр і площа 2

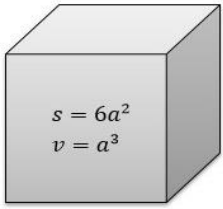
Знайти периметр і площу квадрата, якщо відома довжина його сторони a.

$$p = 4a$$
$$s = a^2$$

3	12 9

8815 Площа поверхні та об'єм 2

Знайти площу поверхні та об'єм куба, якщо відома довжина його ребра.



3	54 27

8816 Степінь числа

На вході програми маємо два натуральних числа **a** та **n**, кожне в окремому рядку. На вихід потрібно подати a^n.

2 10	1024

8817 Кількість n-значних чисел

На вході програми маємо натуральне число **n**. Вивести кількість **n**-значних натуральних чисел.

1	9

Кількість **n**-значних натуральних чисел можна знайти за формулою $9*10^{n-1}$. У найстаршому розряді може будь-яка цифра крім нуля, тобто дев'ять різних і всі десять цифр в інших розрядах.	`n=int(input())` `print(9*10**(n-1))`

8818 Кількість непарних n-значних чисел

На вході програми маємо натуральне число **n**. Вивести кількість непарних **n**-значних натуральних чисел.

1	5

8819 Кількість парних n-значних чисел

На вході програми маємо натуральне число **n**. Вивести кількість парних **n**-значних натуральних чисел.

1	4

Очевидно, що парних буде половина від загальної кількості чисел.	`n=int(input())-1` `print(9*10**n//2)`

8820 Кількість n-значних чисел 2

На вході програми маємо натуральне число **n**. У відповідь потрібно вивести кількість **n**-значних натуральних чисел, у записі яких використано тільки непарні цифри.

1	5

8821 Кількість n-значних чисел 3

На вході програми маємо натуральне число **n**. У відповідь потрібно вивести кількість **n**-значних натуральних чисел, у записі яких використано тільки парні цифри.

1	4

Найстарший розряд кожного з таких чисел можна заповнити однією з чотирьох цифр - 2,4,6,8 (цифра 0 не підходить). Для наступних розрядів підходять п'ять цифр. Отже, актуальна формула $\mathbf{4*5^{n-1}}$.	`n=int(input())` `print(4*5**(n-1))`

8822 Кількість n-значних чисел 4

На вході програми маємо натуральне **n**. У відповідь потрібно вивести кількість **n**-значних натуральних чисел, в записі яких немає жодної цифри **6**.

1	4

Якщо цифру **6** не використовувати, то очевидно, що найстарший розряд вибираємо з **8** цифр (всі крім **0** і **6**). Наступні розряди мають по **9** варіантів. Отже, дійсна формула $\mathbf{8*9^{n-1}}$.

8823 Кількість n-значних чисел 5

На вході програми маємо натуральне **n**. У відповідь потрібно вивести кількість **n**-значних натуральних чисел, в записі яких є хоча б одна цифра **7**.

1	1
2	18

Очевидно, має місце алгоритм :

- кількість всіх **n**-значних натуральних чисел дорівнює $\mathbf{9*10^{n-1}}$;
- кількість **n**-значних чисел, що не містять **7** дорівнює $\mathbf{8*9^{n-1}}$;
- якщо від першого результату відняти другий, то отримаємо відповідь до задачі (до речі, для будь-якої іншої ненульової цифри результат буде аналогічний).

8824 Знайти число

Знайти ціле число **n**, якщо відомі попереднє та наступне до нього числа, записані у довільному порядку через пропуск.

6 8	7

Будь-яке ціле число дорівнює середньому арифметичному своїх сусідів. І ще, виконуючи переніс команди в наступний рядок, достатньо розірвати пару відповідних дужок.	`a,b=map(int, input().split( ))` `print((a+b)//2)`

Обчислення виразів і функцій.

8825 Значення змінної 1

Обчислити значення змінної **y** при заданому дійсному значенні змінної **x**.

$$y = x^3 - \frac{5x^2}{7} + 9x - \frac{3}{x} + 1$$

1	7.286

Для введення дійсного числа використовуємо функцію **float**. Далі, надаємо змінній **y** значення виразу у стилі **Python** та округлюємо результат до тисячних, функція **round** з параметром **3**.	`x=float(input())`
	`y=x**3-5*x*x/7+9*x-3/x+1`
	`print(round(y,3))`

8826 Значення змінної 2

Обчислити значення змінної y при заданому дійсному значенні змінної x.

$$y = x - \frac{x^2 + 4}{2} + x^3 - \frac{3}{x + 7}$$

1	-0.875

8827 Значення змінної 3

Обчислити значення змінної **y** при заданому дійсному значенні змінної **x**.

$$y = \frac{x^2 + 3x - 4}{2x - 3} - \frac{x + 2}{x^2 - 5x + 7}$$

1	-1.000

8828 Значення змінної 4

Обчислити значення змінної **y** при заданому дійсному значенні змінної **x**.

$$y = \frac{2x - 1}{x^2} + \frac{\sqrt{x^2 + 1}}{2}$$

1	1.707

8829 Значення змінної 5

Обчислити значення змінної **y** при заданому дійсному значенні змінної **x**.

$$y = \frac{2x}{\sqrt{x^2+1}} - \frac{\sqrt{x^2+1}}{x^3}$$

1	0.000

Для обчислення квадратного кореня імпортуємо функцію **sqrt** модуля **math**.	`from math import sqrt`
	`x=float(input())`
	`y=(x*2)/sqrt(x**2+1`
	`)-sqrt(x**2+1)/x**3`
	`print(round(y,3))`

8830 Значення змінної 6

Знайти значення змінної **y** при заданому дійсному значенні змінної **x**.

$$y = \frac{\sqrt{x^4+1}}{x^2} - \sqrt{\frac{x^2}{x^2+1}}$$

1	0.707

8831 Значення виразу 1

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$2x^2 - 4xy + 3y^2 + \frac{x+y}{7}$$

1.000 -2.000	21.857

8832 Значення виразу 2

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$\frac{x^2-2xy+4y^2}{x+5} + \frac{3x^2-y^2}{y-7}$$

1.000 -2.000	3.611

Дійсні числа, записані в рядку через проміжок вводяться в програму аналогічно, як і цілі. Ітератор **map** застосовує функцію **float** до всіх значень введених з рядка даних.

```
x,y=map(float,input().split())
z=(x**2-2*x*y+4*y**2)/(x+5)+(3*x**2-y**2)/(y-7)
print(round(z,3))
```

8833 Значення виразу 3

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$\frac{2x+y}{x^2-xy+4y^2}+\frac{2x^2-xy+y^2}{x+4y}$$

1.000 -2.000	-1.143

8834 Значення виразу 4

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$\frac{2xy}{\sqrt{x^2+y^2}}-\frac{(x+y-1)^2}{xy}$$

1.000 -2.000	0.211

8835 Значення виразу 5

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$\frac{\sqrt{x^2+y^2}}{(x-y)^2}-\frac{2xy}{\sqrt{x^2+y^2}}$$

1.000 -2.000	2.037

8836 Значення виразу 6

Знайти значення виразу при заданих дійсних значеннях змінних **x** та **y**.

$$\frac{(x-y)^2}{\sqrt{x^2+y^2-1}}+\frac{\sqrt{x^2+y^2-1}}{2xy}$$

1.000 -2.000	4.000

Цей код демонструє, як вдало, застосувавши додаткову змінну можна суттєво зменшити саму програму. Не забуваємо про функцію **sqrt**.	`from math import sqrt`
	`x,y=map(float,input().split())`
	`w=sqrt(x**2+y**2-1)`
	`z=(x-y)**2/w+w/(2*x*y)`
	`print(round(z,3))`

Ділення цілих чисел.

8837 Частка та залишок

На вході програми маємо два натуральних числа **a** і **b**, записані в одному рядку через пропуск. Потрібно вивести частку та залишок при цілочисельному діленні **a** на **b**.

27 10	2 7

Знайомим способом вводимо цілі числа з одного рядка, обчислюємо та виводимо частку (**//**) і залишок (**%**) .	`a,b=map(int,input().split())` `print(a//b,a%b)`

8838 Петрик і пиріжки

Ще на першій великій перерві у Петрика було **m** гривень, але вже на другій перерві він так зголоднів, що вирішив весь капітал витратити на смачні пиріжки. Скільки пиріжків зможе купити Петрик, якщо їх продають по **n** гривень?

27 10	2

8839 Петрик і пиріжки 2

По сюжету аналогічному попередній задачі потрібно знайти скільки грошей лишилося у Петрика.

27 10	7

8840 Цифра

На вході програми маємо натуральне число **n**. Виведіть крайню справа цифру (розряду одиниць) числа **n**.

27	7

8841 Цифра 2

На вході програми маємо натуральне число **n** (**n>9**). Виведіть передостанню справа цифру (розряду десятків) числа **n**.

527	2

Видаляємо в числі **n** цифру одиниць (**//10**) і знаходимо залишок від ділення (**%10**).	`n=int(input())` `print(n//10%10)`

8842 Цифра 3

На вході програми маємо натуральне число **n** (**n>99**). Виведіть третю справа цифру (розряду сотень) числа **n**.

7512	5

Видаляємо в числі **n** цифри одиниць і десятків (**//100**), знаходимо залишок від ділення (**%10**).	`n=int(input())` `print(n//100%10)`

8843 Видалити цифру

На вході програми маємо натуральне число **n** (**n>9**). Видаліть крайню справа цифру (розряду одиниць) числа **n**.

512	51

8844 Видалити цифру 2

На вході програми маємо натуральне число **n** (**n>9**). Потрібно видалити передостанню справа цифру (розряду десятків) числа **n**.

7512	752

В змінну **a** обчислюємо число, яке стоїть перед цифрою десятків і множимо його на **10**, в нашому випадку **a=7512//100*10=75*10=750**. Далі в змінну **b** знаходимо цифру одиниць **b=2** та виводимо суму **a+b**.	`n=int(input())`
	`a=n//100*10`
	`b=n%10`
	`print(a+b)`

8845 Видалити цифру 3

На вході програми маємо натуральне число **n** (**n>99**). Видаліть третю справа цифру (розряду сотень) числа **n**.

4625	425

8846 Обмін цифр

На вході програми маємо натуральне число **n** (**n>9**). В числі **n** потрібно обміняти місцями цифри десятків і одиниць.

512	521

В змінну **a** знаходимо цифру десятків, а в змінну **b** - цифру одиниць. Обмін виконуємо за формулою **n-10*a+10*b+a-b=n+9*(b-a)**. У нашому випадку **512+9*(2-1)=521**.	`n=int(input())`
	`a=n//10%10`
	`b=n%10`
	`print(n+9*(b-a))`

8847 Обмін цифр 2

На вході програми маємо натуральне число **n** (**n>99**). В числі **n** потрібно обміняти місцями цифри сотень і одиниць.

3215	3215

8848 Обмін цифр 3

На вході програми маємо натуральне число **n** (**n>99**). В числі **n** потрібно обміняти місцями цифри сотень і десятків.

7512	7152

В змінну **a** знаходимо цифру сотень, а в змінну **b** - цифру десятків. Формула для обміну буде **n=100*a+100*b+10*a-b*10=n+90*(b-a)**.	`n=int(input())`
	`a=n//100%10`
	`b=n//10%10`
	`print(n+90*(b-a))`

Ділення цілих чисел 2

8849 Подвоєння

Подвоїти кожну цифру заданого двоцифрового числа.

27	2277

В змінну **a** знаходимо цифру десятків, в змінну **b** - цифру одиниць. Формула подвоєння **1000*a+100*a+10*b+b=1100*a+11*b**.	`n=int(input())`
	`a=n//10`
	`b=n%10`
	`print(a*1100+b*11)`

8850 Сума цифр

Знайти суму цифр заданого трицифрового числа.

512	8

8851 Число навпаки

Записати чотиризначне натуральне число в зворотному порядку.

1234	4321

В змінні **a, b, c, d** послідовно знаходимо цифри заданого числа, використавши відомі команди. Формула оберненого числа: **a+10*b+100*c+1000*d**.	`n=int(input())`
	`a=n//1000`
	`b=n//100%10`
	`c=n//10%10`
	`d=n%10`
	`m=a+b*10+c*100+d*1000`
	`print(m)`

8852 Видалити цифри

У п’ятизначному натуральному числі видалити цифри парних розрядів, тобто другу і четверту.

12345	135

8853 Видалити цифри 2

У п’ятизначному натуральному числі видалити цифри непарних розрядів, тобто першу, третю і п’яту.

12345	24

В змінні **a, b** знаходимо цифри парних розрядів, тобто другу й четверту. Далі складаємо з цифр двоцифрове число заново за формулою: **10*a+b**.	`n=int(input())`
	`a=n//1000%10`
	`b=n//10%10`
	`print(a*10+b)`

8854 Число навпаки 2

Записати п’ятизначне натуральне число в зворотному порядку.

12345	54321

8855 Знайти число 1

Від тризначного числа **N** відняли його останню цифру. Коли результат розділили на 10, а до частки зліва приписали останню цифру числа **N**, то отримали число **X**. Знайти число **X**, якщо відомо значення **N**.

123	312

Умова задачі описує циклічний зсув цифр числа вправо. Отже, записуємо в **b** цифру одиниць та робимо зсув цифр числа **N**, розділивши його на **10**. Далі в розряд сотень дописуємо цифру **b**.	`n=int(input())`
	`a=n//10`
	`b=n%10`
	`print(b*100+a)`

8856 Знайти число 2

В трицифровому числі **N** закреслили першу цифру. Коли результат помножили на **10**, а до добутку додали першу цифру числа **N**, то отримали число **X**. Знайти число **X**, якщо відомо значення **N**.

123	231

8857 Знайти число 3

В трицифровому числі **N** видалили другу цифру. Коли результат помножили на **10**, а до добутку додали другу цифру числа **N**, то отримали число **X**. Знайти число **X**, якщо відомо значення **N**.

123	132

Спочатку в змінні **a, b, c** послідовно знаходимо цифри заданого числа. Далі цифра сотень **a** лишається на своєму місці, а цифри **b** і **c** обмінюються. Формула: **100*a+10*c+b**.	`n=int(input())`
	`a=n//100`
	`b=n//10%10`
	`c=n%10`
	`print(a*100+c*10+b)`

8858 Знайти число 4

В трицифровому числі **N** закреслили останню цифру. Коли в утвореному двозначному числі переставили цифри, а потім приписали до них зліва останню цифру числа **N**, то отримали число **X**. Знайти число **X**, якщо відомо значення **N**.

123	321

8859 Знайти число 5

В трицифровому числі **N** видалили другу цифру. Коли до утвореного двохзначного числа приписали зліва другу цифру числа **N**, то отримали число **X**. Знайти число **X**, якщо відомо значення **N**.

123	213

8860 Знайти суму

Для заданого трицифрового числа **N** знайти суму тризначних чисел, утворених всіма можливими перестановками цифр числа **N**, включаючи тотожну.

123	1332

В змінні **a, b, c** послідовно знаходимо цифри заданого числа. Легко перевірити, що різних перестановок цифр тризначного числа буде шість, отже кожна цифра побуває в кожному розряді по два рази. Якщо все додати і спростити, то маємо (**a+b+c**)***222**.	`n=int(input())`
	`a=n//100`
	`b=n//10%10`
	`c=n%10`
	`print((a+b+c)*222)`

Розгалуження

Всі програми з попереднього розділу були лінійними, команди в яких виконувались безальтернативно – підряд, як рядки в пісні. Але зазвичай, розв'язуючи інші задачі, програма в залежності від вхідних даних повинна вміти зробити вибір: значення парне чи непарне; число менше, рівне або більше нуля; пора року весна, літо, осінь або зима, ось тоді в пригоді стане команда розгалуження. Така конструкція традиційно має засіб для перевірки логічних тверджень і перехід до виконання визначених команд у позитивному і протилежному випадках.

В кожній мові програмування розгалуження має свої особливості запису, але основний зміст від цього не змінюється. В мові Python команда розгалуження або вибору реалізується службовими словами **if** – **elif – else**. Спочатку записується частина **if** з умовним виразом, далі можуть йти декілька секцій **elif** і насамкінець необов'язкова секція **else**. Загальна форма розгалуження на Python:

```
if    умова1 :          або    if    умова1 : блок1
      блок1                    elif  умова2 : блок2
elif  умова2 :                 else        : блок3
      блок2
else  :
      блок3
```

Істинність умов перевіряється в порядку, як вони записані в команді. Відповідний блок команд виконується тільки у випадку, якщо його умова істина (**True**), а наступна перевірка умов припиняється. Блок команд секції **else** буде задіяний у випадку, якщо всі попередні умови виявилися хибними (**False**).

Для запису умов використовуються знаки відношення між величинами і службові слова **and**, **or** і **not**. Якщо потрібно, щоб декілька простих умов виконувались одночасно – з'єднуємо їх операцією **and**; **or** використовуємо, тоді коли достатньо виконання однієї з простих умов, а **not** – щоб записати, що умова не виконується.

Зауважимо також, що існує зв'язок між логічними значеннями умов і цілими числами, втім для такої емуляції достатньо лише двох чисел, отже **int**(**True**)=**1; int**(**False**)=**0; bool**(**1**)= **True; bool**(**0**)= **False**.

Запис відношень між величинами	
дорівнює	`==`
не дорівнює	`!=`
більше	`>`
менше	`<`
більше рівне	`>=`
менше рівне	`<=`
логічне «І»	`and`
логічне «АБО»	`Or`
логічне заперечення	`not`
Функції, що пов'язані з розгалуженням	
повертає True, якщо всі елементи True	**`all`**(послідовність)
повертає True, якщо хоча б один елемент True	**`any`**(послідовність)
повертає найбільший елемент	**`max`**(послідовність)
повертає найменший елемент	**`min`**(послідовність)
перетворення аргументу до логічного типу	**`bool(x)`**
створення списку об'єктів	**`list(x)`**

Задачі з розгалуженням

8861 Модуль числа

Програма читає з консолі ціле число та виводить його модуль.

7	7
-1	1

8862 Знак числа

Програма повинна прочитати з консолі ціле число та вивести **-1**, **0** або **1**, якщо введене значення від'ємне, нульове і додатне, відповідно.

7	1
-5	-1
0	0

На початок вводимо ціле значення **n**, а далі просто переписуємо умову задачі в стилі **Python**.	`n=int(input())`
	`if n>0 : print(1)`
	`elif n==0 : print(0)`
	`else : print(-1)`

8863 Парність числа

Програма повинна прочитати з консолі ціле число та вивести **1**, якщо число парне і **0** у протилежному випадку.

7	0
-2	1

8864 Числа одного знаку

На вході програми маємо два цілих ненульових числа **n** і **m**, записаних в одному рядку через пропуск. Програма повинна вивести **1**, якщо числа **n** і **m** одного знаку і **0** у протилежному випадку.

7 4	1
-2 5	0

Вводимо значення **n**, **m** і пригадаємо, що добуток двох чисел одного знаку завжди додатній.	`n,m=map(int,input().split())`
	`if n*m>0 : print(1)`
	`else : print(0)`

8865 Однакова парність

На вході програми маємо два цілих числа **n** і **m**, записаних в одному рядку через пропуск. Програма повинна вивести **1**, якщо числа **n** і **m** мають однакову парність, і **0** у протилежному випадку.

7 4	0
-2 8	1

Використаємо той факт, що сума двох чисел з однаковою парністю завжди парна. В стилі **Python** це виглядає ось так.	`n,m=map(int,input().split())`
	`if (n+m)%2==0 : print(1)`
	`else : print(0)`

8866 Подільність

На вході програми маємо два цілих ненульових числа **n** і **m**, записаних в одному рядку через пропуск. Програма повинна вивести **1**, якщо число **n** ділиться на **m** націло, і **0** у протилежному випадку.

7 4	0
8 -2	1

Вводимо значення **n** та **m**, а далі просто переписуємо умову задачі в стилі **Python**.	`n,m=map(int,input().split())`
	`if n%m==0 : print(1)`
	`else : print(0)`

8867 Менше з двох

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Потрібно вивести менше з них.

7 4	4

8868 Більше з двох

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Потрібно вивести більше з них.

7 4	7

8869 Впорядкування двох

На вході програми маємо два цілих числа **a** і **b**, записаних в одному рядку через пропуск. Задані числа потрібно вивести в порядку зростання тобто спочатку менше, а потім більше з них.

7 4	4 7

Перший варіант програми використовує функції **min** і **max**.	`a,b=map(int,input().split())` `print(min(a,b),max(a,b))`

У другій версії виконуємо обмін значень у змінних, якщо вони не зростають. Зверніть увагу наскільки красиво виглядає обмін в стилі **Python**.	`a,b=map(int,input().split())`
	`if a>b : a,b=b,a`
	`print(a,b)`

8870 Менше з трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Потрібно вивести менше з них.

7 2 4	2

8871 Більше з трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Потрібно вивести більше з них.

7 2 4	7

8872 Впорядкування трьох

На вході програми маємо три цілих числа **a**, **b** і **c**, записаних в одному рядку через пропуск. Задані числа потрібно вивести в порядку зростання.

7 2 4	2 4 7

Послідовно виконуємо обміни значень у змінних, якщо вони не зростають. Таких аналогічних спроб буде три, краще перевірити на конкретних прикладах.	`a,b,c=map(int,input().split())`
	`if a>b : a,b=b,a`
	`if b>c : b,c=c,b`
	`if a>b : a,b=b,a`
	`print(a,b,c)`

Задачі з розгалуженням 2

8873 Одноцифрове число

Задано ціле число **n**. Вивести **Ok**, якщо число **n** одноцифрове та **No** у протилежному випадку.

7	Ok
-13	No

8874 Двоцифрове число

Задано ціле число **n**. Вивести **Ok**, якщо число **n** двоцифрове та **No** у протилежному випадку.

17	Ok
-123	No

Вводимо ціле значення **n**, застосовуємо функцію **abs** для від'ємних, а далі просто переписуємо умову задачі в стилі **Python**.	`n=int(input())`
	`n=abs(n)`
	`if 9<n<100 : print("Ok")`
	`else : print("No")`

8875 Трицифрове число

Задано ціле число **n**. Вивести **Ok**, якщо число **n** трицифрове та **No** у протилежному випадку.

123	Ok
-27	No

8876 Ціле число

Задано дійсне число **n**. Вивести **Ok**, якщо число **n** ціле та **No** у протилежному випадку.

7.000	Ok
-21.121	No

З консолі вводимо дійсне значення **n**, застосовуємо функцію **round** , щоб округлити його до цілого. Далі перевіряємо, чи співпадають ці значення та виводимо відповідне повідомлення, як того вимагає умова задачі.	`n=float(input())`
	`if round(n)==n :`
	`  print("Ok")`
	`else : print("No")`

8877 Повний квадрат

Задано натуральне число **n**. Якщо число **n** є повним квадратом деякого натурального **m**, то виведіть число **m**. У протилежному випадку вивести відповідь **No**.

25	5
27	No

Вводимо ціле значення **n** та знаходимо в **m** корінь квадратний з **n**. Використавши функцію **round** перевіряємо, чи значення змінної **m** ціле, публікуємо результат. Не забуваємо імпортувати функцію **sqrt** з модуля **math**.

```
from math import sqrt
n=int(input())
m=sqrt(n)
if round(m)==m :
    print(int(m))
else : print("No")
```

8878 Степінь числа 10

Програма має прочитати з консолі натуральне число **N**. Якщо задане число **N** дорівнює $\mathbf{10}^{\mathbf{M}}$,тобто є деяким степенем числа **10**, то у відповідь вивести число **M**. У протилежному випадку вивести відповідь **No**.

100	2
27	No

8879 Трикутник

На вході програми маємо три натуральних числа **a**, **b** і **c**, записані в одному рядку через пропуск. У відповідь потрібно вивести суму заданих чисел, якщо існує трикутник з довжинами сторін **a**, **b**, **c** і повідомлення **No** у протилежному випадку.

3 4 2	9
3 2 7	No

За правилом існування трикутника сума будь-яких двох його сторін має бути більшою за третю сторону. У програмі вводимо значення сторін та записуємо це правило в стилі **Python**.

```
a,b,c=map(int,input().split())
if (a<b+c)and(b<a+c)and(c<a+b):
    print(a+b+c)
else : print("No")
```

8880 Рівносторонній трикутник

На вході програми маємо три натуральних числа **a**, **b** і **c**, записані в одному рядку через пропуск. У відповідь потрібно вивести квадрат будь-якого з заданих чисел, якщо існує рівносторонній трикутник з довжинами сторін **a**, **b**, **c** і повідомлення **No** у протилежному випадку.

7 7 7	49
4 6 9	No

Вводимо з консолі значення сторін **a**, **b**, **c**. Очевидно, потрібно перевірити їх на однаковість. Так це виглядає в стилі **Python**.	`a,b,c=map(int,input().split())`
	`if (a==b==c) : print(a*a)`
	`else : print("No")`

8881 Рівнобедрений трикутник

На вході програми маємо три натуральних числа **a**, **b** і **c**, записані в одному рядку через пропуск. У відповідь потрібно вивести суму заданих чисел, якщо існує рівнобедрений трикутник з довжинами сторін **a**, **b**, **c** і повідомлення **No** у протилежному випадку.

7 7 3	17
4 6 9	No

8882 Квадрат

На вході програми маємо чотири натуральних числа **a**, **b**, **c** і **d**, записані в одному рядку через пропуск. У відповідь потрібно вивести квадрат будь-якого з заданих чисел, якщо існує квадрат з довжинами сторін **a**, **b**, **c**, **d** і повідомлення **No** у протилежному випадку.

7 7 7 7	49
9 6 9 6	No

8883 Прямокутник

На вході програми маємо чотири натуральних числа **a**, **b**, **c** і **d**, записані в одному рядку через пропуск. У відповідь потрібно вивести суму заданих чисел, якщо існує прямокутник з довжинами сторін **a**, **b**, **c**, **d** і повідомлення **No** у протилежному випадку.

7 4 4 7	22
9 9 9 6	No

Вводимо з консолі значення сторін **a**, **b**, **c**, **d**. Звісно, прямокутник можна скласти, якщо маємо дві пари однакових значень. Але які з них? Тому перевіряємо всі можливі випадки, їх всього три. В програмі кожен варіант прямокутника описує окрема логічна змінна.	`a,b,c,d=map(int, input().split())`
	`u=(a==b and c==d)`
	`v=(a==c and b==d)`
	`w=(a==d and c==b)`
	`if u or v or w :`
	`  print(a+b+c+d)`
	`else : print("No")`

8884 Трикутник 2

На вході програми маємо три натуральних числа **a**, **b** і **c**, записані в одному рядку через пропуск. У відповідь потрібно вивести одне з повідомлень:
– **equilateral**, якщо існує рівносторонній трикутник з довжинами сторін **a**, **b**, **c**;
– **isosceles**, якщо існує рівнобедрений трикутник з довжинами сторін **a**, **b**, **c**;
– **versatile**, якщо існує різносторонній трикутник з довжинами сторін **a**, **b**, **c**;
– **invalid**, якщо трикутника з довжинами сторін **a**, **b**, **c** не існує.

7 7 7	equilateral
9 9 3	isosceles
2 3 4	versatile
2 3 5	invalid

Задачі з розгалуженням 3

8885 Попереднє непарне число

Дано ціле число **n**. Вивести попереднє непарне до числа **n**.

7	5
8	7

Вводимо ціле значення **n**. Якщо воно непарне, то попереднє непарне буде на **2** менше і менше на **1** в протилежному випадку. Це й ілюструє код в стилі **Python**.	`n=int(input())`
	`if n%2!=0 : print(n-2)`
	`else : print(n-1)`

8886 Попереднє парне число

Дано ціле число **n**. Вивести попереднє парне до числа **n**.

7	6
6	4

8887 Наступне непарне число

Дано ціле число **N**. У потрібно вивести наступне непарне за числом **N**.

7	9
6	7

8888 Наступне парне число

Дано ціле число **N**. У відповідь вивести наступне парне за числом **N**.

7	8
4	6

Вводимо ціле значення **n**. Якщо воно парне, то наступне парне буде на **2** більше і більше на **1** в протилежному випадку. Це й ілюструє код в стилі **Python**.	`n=int(input())`
	`if n%2==0 : n+=2`
	`else : n+=1`
	`print(n)`

8889 Кількість непарних цифр

Задано п’ятизначне натуральне число **N**. Потрібно знайти кількість непарних цифр в числі **N**.

12345	3

Вводимо ціле значення **n**. В змінні **a**, **b**, **c**, **d**, **e** послідовно знаходимо цифри заданого числа, використавши відомі команди. Виводимо суму їх залишків від ділення на **2.** Логічно, що для непарних цифр такі значення будуть рівні **1**, а для парних - **0**. Отже їх сума дасть кількість непарних цифр.

```
n=int(input())
a=n//10000%10
b=n//1000%10
c=n//100%10
d=n//10%10
e=n%10
print(a%2+b%2+c%2+
d%2+e%2)
```

8890 Збільшити парні цифри

Задано п’ятизначне натуральне число **N**. Потрібно всі парні цифри в числі **N** збільшити на **1**.

12345	13355

8891 Рівно одна умова з двох

Для заданого цілого числа **N** вивести відповідь **YES**, якщо виконується рівно одна з наступних умов і **NO** у протилежному випадку.

– число **N** парне;
– число **N** від’ємне і кратне 3.

22	YES
7	NO

Вводимо ціле значення **n**. Кожну умову записуємо в окрему змінну, перетворивши логічний тип в цілий, причому, якщо умова виконується, то значення відповідної змінної дорівнює **1** і **0** в протилежному випадку. Щоб виконувалась рівно одна умова сума змінних **x**, **y** має дорівнювати одиниці.

```
n=int(input())
x=int(n%2==0)
y=int(n<0 and n%3==0)
if x+y==1 :
    print("YES")
else : print("NO")
```

8892 Хоча б одна умова з двох

Для заданого цілого числа **N** вивести відповідь **YES**, якщо виконується хоча б одна з наступних умов і **NO** у протилежному випадку.

- число **N** непарне;
- число **N** додатне і тризначне.

7	YES
8	NO

8893 Кожна умова з двох

Для заданого цілого числа **N** вивести відповідь **YES**, якщо виконується кожна з наступних умов і **NO** у протилежному випадку.

- число **N** кратне трьом;
- число **N** парне і двозначне.
-

12	YES
27	NO

8894 Жодна умова з двох

Для заданого цілого числа **N** вивести відповідь **YES**, якщо не виконується жодна з наступних умов і **NO** у протилежному випадку.

- число **N** парне і додатне;
- число **N** непарне і від'ємне.
-

7	YES
24	NO

Вводимо ціле значення **n**. Кожну умову записуємо в окрему змінну аналогічно попереднім задачам, перетворивши логічний тип в ціле число. Щоб не виконувалась жодна умова сума змінних **x**, **y** має дорівнювати нуль.	`n=int(input())`
	`x=int(n%2==0 and n>0)`
	`y=int(n%2!=0 and n<0)`
	`if x+y==0 :`
	`  print("YES")`
	`else : print("NO")`

8895 Додатні і від'ємні числа

Задано три цілих числа **a**, **b**, **c**. У відповідь потрібно вивести **YES**, якщо серед них є хоча б одне додатне та хоча б одне від'ємне число і **NO** у протилежно-му випадку.

1 -2 3	YES
1 2 3	NO

З консолі вводимо значення цілих чисел **a**, **b**, **c**. У програмі краще застосувати твердження протилежне до умови задачі: якщо всі три числа одночасно додатні або від'ємні, то виводимо негативну відповідь і позитивне повідомлення в протилежному випадку. Код в стилі **Python** просто ілюструє сказане в поясненні.	`a,b,c=map(int,input(`
	`).split())`
	`x=(a>0 and b>0 and c>0)`
	`y=(a<0 and b<0 and c<0)`
	`if x+y==1 :`
	`  print("NO")`
	`else : print("YES")`

8896 Різні цифри

Програма повинна ввести з консолі ціле трицифрове число **N** та вивести у відповідь **YES**, якщо всі цифри числа **N** різні і **NO** у протилежному випадку.

123	YES
477	NO

Цикл while з передумовою

Кожна поважаюча себе мова програмування містить команду повторення або циклу, причому не одну, а дві або більше. В Python їх дві **while** ("поки") і **for** ("для") і вони надають широкий діапазон можливостей для запису алгоритмів і обробки даних.

Цикл **while** традиційно містить блок з умовою, який керує повторенням і тіло циклу, де записана послідовність команд. Тіло циклу виконується поки умова істинна (**True**). Умова перевіряється перед виконанням тіла циклу і у випадку, якщо вона хибна (**False**) – тіло циклу не виконується, а керування передається наступній команді після циклу.

```
while умова :          або    while умова : тіло циклу
    тіло циклу
```

Зазвичай, команди циклу записують через табуляцію у наступних стрічках після команди **while** або в одній стрічці після двокрапки (декілька команд розділяють крапкою з комою "**;**"). Як правило, цикл **while** використовується, коли неможливо визначити точно кількість ітерацій циклу.

Завдання, що включають цикл, на відміну від перших двох розділів, непросто сприймаються учнями і викликають у них ряд труднощів. Зрозуміло, що для засвоєння циклу **while** необхідне неабияке динамічне мислення. Адже потрібно не тільки зрозуміти і переписати код програми, а й навчитись самостійно сконструювати команди циклу, поступаючи по аналогії до готових кодів програм. Отже, запропонований набір нескладних задач з циклом **while**, безумовно, допоможе учням розібратися в цій роботі.

Тут пропонується два нескладні набори по 12 завдань, де можливо спрацювати по аналогії, і на кінець 10 завдань складнішого рівня. Звісно, більше пояснень і кодів програм слід очікувати якраз в завершальному наборі.

Задачі з циклом while

8897 Наступне число 2

Програма має ввести з консолі ціле число **n** та знайти число наступне за **n**, яке кратне **10**.

7	10

Вводимо ціле **n** і збільшуємо його значення на **1.** Адже, початкове значення **n** теж може ділиться на **10,** а це буде неправильна відповідь – важливо правильно вказати початкове значення змінної, яка керує циклом. Далі перевіряємо **n** на кратність **10** і якщо результат негативний, то переходимо до наступного числа. Команда **n+=1** в **Python** збільшує значення змінної **n** на **1**. Відповідь буде знайдено не більш як за 9 ітерацій циклу, коли **n** стане кратним **10**.

```
n=int(input())+1
while n%10!=0 :
    n+=1
print(n)
```

8898 Наступне число 3

Програма має ввести з консолі натуральне число **n** та знайти число наступне за **n**, що є повним квадратом.

7	9

8899 Наступне число 4

Програма має ввести з консолі натуральне число **n** та знайти число наступне за **n**, що є деяким степенем двійки.

7	8

В цій задачі моделюємо послідовні степені двійки, аж поки деяке значення 2^k не перевищить заданого значення **n**, тоді цикл **while** закінчується і програма виводить шукану степінь двійки - 2^k. Зверніть увагу, що в цьому прикладі ітераціями циклу керує змінна **k**.

```
n=int(input())
k=1
while n>=2**k : k+=1
print(2**k)
```

8900 Найменше з більших

На вході програми маємо ціле число **n**. Серед цілих чисел більших **n** та кратних **7** знайти найменше.

3	7

8901 Найменше з більших 2

На вході програми маємо натуральне число **n**. Серед натуральних чисел, що мають більше цифр ніж число **n** знайти найменше.

7	10

8902 Найменше з більших 3

На вході програми маємо натуральне число **n**. Серед натуральних чисел більших ніж **n**, що не діляться на **2**, **3** і **5** знайти найменше.

4	7

Зауважимо, що визначення ”найменше з більших” ні що інше, як наступне число, тому умову можна прочитати так: знайти наступне число, яке не кратне жодному з чисел **2**, **3** і **5**. Отож, поступаємо аналогічно попереднім завданням. Змінюється тільки умова закінчення, а точніше умова продовження циклу. Ось такий код в стилі **Python** .

```
n=int(input())+1
while (n%2==0 or n%3==0
or n%5==0) :
    n+=1
print(n)
```

8903 Попереднє число 2

Програма має ввести з консолі ціле число **n** та знайти число попереднє до **n**, що кратне **5**.

7	5

У наступних задачах будемо вести пошук серед менших чисел, ніж задане. Отже, значення **n** починаємо з попереднього числа і продовжуємо ітерації зменшуючи **n** на **1**, команда **n-=1**.

```
n=int(input())-1
while n%5!=0 :
    n-=1
print(n)
```

8904 Попереднє число 3

Програма має ввести з консолі натуральне число **n** та знайти число попереднє до **n**, що є деяким степенем двійки.

7	4

8905 Попереднє число 4

Програма має ввести з консолі натуральне число **n** та знайти число попереднє до **n**, що не ділиться на **2**, **3** і **5**.

11	7

8906 Найбільше з менших

На вході програми маємо ціле число **n**. Серед цілих чисел менших **n** та кратних **11** знайти найбільше.

17	11

8907 Найбільше з менших 2

На вході програми маємо натуральне число **n**. Серед чисел менших **n**, які рівні повним кубам знайти найбільше.

11	8

Очевидно, що "найбільше з менших" - це просто попереднє число. Думаю, Ви вже здогадались? Тому, моделюємо повні куби чисел 1 2 3… аж поки деяке значення k^3 не перевищить заданого значення **n**, тоді ітерації циклу закінчуються, а програма виводить куб попереднього числа $(k-1)^3$. Ось такий код в стилі **Python** .

```
n=int(input())
k=0
while k**3<n :
    k+=1
print((k-1)**3)
```

8908 Найбільше з менших 3

На вході програми маємо натуральне число **n** (**n>9**). Серед натуральних чисел, що мають менше цифр ніж число **n** знайти найбільше.

11	9

Зрозуміло, що у пошуку будуть приймати участь числа 9 99 999…Моделювати їх можна за формулою 10^k-1. Код на **Python** напишіть самостійно.

Задачі з циклом while 2

8909 Довжина послідовності

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Знайти довжину даної послідовності, не враховуючи останнього нуля.

7 -1 4 -6 0	4

У наступних задачах працюємо з послідовностями чисел в плані їх введення, виведення, підрахунку і тд. Отже, читаємо з консолі числа в змінну **a**, паралельно рахуємо в змінній **k** ітерації циклу або кількість прочитаних чисел. Початковий ввід змінної **a** виконуємо перед циклом, щоб передбачити випадок появи першого **0**.

```
a,k=int(input()),0
while a!=0 :
    k+=1
    a=int(input())
print(k)
```

8910 Сума послідовності

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Потрібно знайти суму даної послідовності, не враховуючи останнього нуля.

7 -1 4 -6 0	4

На початок була кількість, а тепер сума. Коди програм відрізняються лише одною командою **k+=a** – збільшити значення **k** на **a.** Потрібно взяти на замітку.

```
a,k=int(input()),0
while a!=0 :
    k+=a
    a=int(input())
print(k)
```

8911 Кількість від'ємних

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Потрібно знайти кількість від'ємних чисел в даній послідовності.

7 -1 4 -6 0	2

8912 Сума додатних

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Потрібно знайти суму додатних чисел в даній послідовності, не враховуючи останнього нуля.

7 -1 4 -6 0	11

Перегляд чисел виконуємо аналогічно, як в попередніх завданнях. Перший раз читаємо з консолі значення **a** і починаємо змінну **s** з **0**. Тіло циклу всі додатні значення **a** закидає в **s** та продовжує вводити числа, поки не зустрінеться **0**. На кінець виводимо значення **s**.	`a,s= int(input()),0`
	`while a!=0 :`
	`  if a>0 : s+=a`
	`  a=int(input())`
	`print(s)`

8913 Кількість непарних

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Потрібно знайти кількість непарних чисел в даній послідовності.

7 -1 4 -6 0	2

В код програми пишемо аналогічний перегляд, як в попередніх завданнях, тільки тепер потрібно ловити непарні значення **a**, у яких остача від ділення на **2** не дорівнює **0** . Ось такий код в стилі **Python** .	`a,k= int(input()),0`
	`while a!=0 :`
	`  if a%2!=0 : k+=1`
	`  a=int(input())`
	`print(k)`

8914 Сума парних

На вході програми маємо послідовність цілих чисел, що закінчується числом **0**. Потрібно знайти суму парних чисел в даній послідовності, не враховуючи останнього нуля.

7 -1 4 -6 0	-2

8915 Всі непарні

Дано натуральне число **n**. Вивести в порядку зростання всі непарні натуральні числа менші **n**.

8	1 3 5 7

По умові задачі потрібно послідовно вивести всі непарні натуральні числа менші заданого **n.** Отже вводимо значення **n** та починаємо значення **k** з одиниці – найменше непарне число. Цикл виконується поки значення **k** менше **n** і якщо це так, то тіло циклу виводить поточне **k** і збільшує його значення на **2**, щоб отримати наступне непарне число. Ось такий код в стилі **Python** .	`n=int(input())`
	`k=1`
	`while k<n :`
	`  print(k,end=' ')`
	`  k+=2`

8916 Перші парні

Програма має ввести з консолі натуральне число **n** та вивести в порядку зростання **n** перших парних натуральних чисел.

3	2 4 6

8917 Степені двійки

Для заданого натурального числа **n** вивести всі степені двійки менші за **n**.

7	2 4

8918 Повні квадрати

Дано натуральне число **n**. Вивести в порядку зростання **n** перших квадратів натуральних чисел.

3	1 4 9

Опишемо змінні: **n** - замовлена кількість квадратів чисел, **k** – кількість ітерацій циклу, **k²** – квадрат поточного числа для виведення у відповідь. Опція **end**=’ ‘ в операторі **print** виводить числа через проміжок.	`n,k=int(input()),0`
	`while k<n :`
	`  k+=1`
	`  print(k*k,end=' ')`

8919 Повні куби

Програма повинна прочитати з консолі натуральне число **n** та вивести в порядку зростання всі повні куби, що менші **n**.

9	1 8

8920 Некратні 2 3 5

Програма повинна прочитати з консолі натуральне число **n** та вивести в порядку зростання **n** перших натуральних чисел, що не діляться на **2**, **3** і **5**.

2	1 7

Задачі з циклом while 3

Більшість завдань цього розділу виконують обробку цифр цілих чисел з невідомою кількістю цифр. Перегляд цифр в кожному випадку починається з розряду одиниць і продовжується, поки в заданого числа не закінчяться цифри. Позначимо задане число змінною **n**, а поточні його цифри з'являються в змінній **x**, то наступний цикл моделює описаний процес:

```
while n>0 :
    x=n%10
    n//=10
```

8921 Цифри числа

Вивести всі цифри заданого натурального числа **n**, починаючи від менших розрядів.

123	3 2 1

І ще раз, повторимо описане вище. Вводимо з консолі натуральне число **n**. Ітерації циклу будуть продовжуватись поки в числі **n** ще лишаються цифри, тобто **n>0**. Перша команда циклу виводить його цифру розряду одиниць, а друга команда видаляє її з числа **n**.	`n=int(input())`
	`while n>0 :`
	`  print(n%10)`
	`  n//=10`

8922 Кількість цифр

Знайти кількість цифр заданого цілого числа **n**.

37	2

8923 Число навпаки 3

Записати задане натуральне число **n** в зворотному порядку.

27	72

Не будемо знову зупинятися на процесі виділення цифр числа **n**, а уважно розглянемо команду **k=k*10+n%10** -вона виконує циклічний зсув цифр числа **k** вліво і додає наступну цифру числа **n**. До речі, скласти по цифрам число, що дорівнює заданому не так просто. Швидше виконати процес обертання двічі.	`n,k=int(input()),0`
	`while n>0 :`
	`  k=k*10+n%10`
	`  n//=10`
	`print(k)`

8924 Сума парних цифр числа

Знайти суму парних цифр натурального числа **n**.

234	6

8925 Добуток непарних цифр числа

Знайти добуток непарних цифр натурального числа **n**.

327	21

Модель процесу аналогічна попереднім завданням. Ось команда, яка обчислює добуток від'ємних **if x%2!=0 : k*=x** – якщо поточна цифра **x** числа **n** непарна, то помножимо на неї **k.** І такий код в стилі **Python**.	`n,k=int(input()),1`
	`while n>0 :`
	`  x=n%10`
	`  if x%2!=0 : k*=x`
	`  n//=10`
	`print(k)`

8926 Заміна парності

Задано натуральне число **n**. Потрібно збільшити на **1** усі його парні цифри та зменшити на **1** усі його непарні цифри.

30	21

8927 Найменший дільник

Для заданого натурального числа **n** (**1** < **n** < **2147000000**) виведіть його найменший дільник, відмінний від **1**.

21	3

Спробуємо написати код, який першим приходить на думку: значення **k** - кандидата в дільники починаємо з **2** і поки **n** некратне **k**, збільшуємо значення **k**. Отже, цей код працює не на всіх тестах, тому що значення **n** може бути достатньо великим і єдиним своїм дільником, тому ліміт часу.	`n,k=int(input()),2`
	`while n%k!=0 :`
	`  k+=1`
	`print(k)`

Згадуємо з математики, що якщо у числа **n** немає дільників на проміжку [**2**,$\sqrt{n}$], то число **n** просте і єдиним його дільником, відмінним від **1** буде саме число **n**. В програмі достатньо команду **k+=1** замінити розгалуженням з нижньої правої клітини таблиці. Маємо **100%**.	`if k*k>n : k=n` `else : k+=1`

8928 Найбільший дільник

Для заданого натурального числа **n** (**1** < **n** < **2147000000**) виведіть його найбільший дільник, відмінний від **n**.

21	7

8929 Просте число

На вході програми маємо натуральне число **n** (**n**>**1**). Потрібно перевірити, чи задане число — просте, тобто ділиться тільки на **1** і **n**.

7	1
15	0

8930 Прості множники

На вході програми маємо натуральне число **n** (**n**>**1**). Потрібно розкласти його на прості множники.

60	2 2 3 5

Опишемо зрозумілий алгоритм розкладу натурального числа на прості множники. Очевидно, що пошук триватиме поки ще є множники або число **n** більше **1**. Почнемо з дільника **d**=**2**. Якщо число **n** кратне **d**, то виконуємо ділення та виводимо значення **d** на екран. В протилежному випадку переходимо до наступного кандидата в дільники, збільшивши **d** на **1**. І знову ліміт часу.

А як виправити?

```
n=int(input())
d=2
while n>1 :
    if n%d==0 :
        n//=d
        print(d)
    else : d+=1
```

Цикл FOR

Цикл **for** або цикл з параметром в мові Python містить ім'я змінної і множину значень, які буде пробігати змінна. Областю значень може бути список (масив), рядок або діапазон.

```
for v in список :          або    for u in рядок :
    тіло циклу                        тіло циклу
```

В першому випадку змінна **v** послідовно набуває значень кожного елементу списку, а в другому **u** пробігає всі символи рядка, причому, тіло циклу виконується щоразу для нового значення змінної. Кількість ітерацій циклу можна визначити, застосувавши стандартну функцію **len**() до відповідного об'єкта.

Найчастіше вказаний діапазон для циклу **for** задає функція **range**, тому доцільно детальніше розібратись з її параметрами.

Діапазон **range**(**n**) забезпечує повторення тіла циклу рівно **n** раз, причому змінна циклу буде набувати значень **0 1**…(**n-1**). Якщо значення **n** дорівнює нулю або від'ємне, то цикл не виконається жодного разу.

Виклик **range**(**a**,**b**) означає, що індексна змінна прийматиме значення від **a** до **b-1**, тобто перший параметр функції **range**, задає початкове значення змінної, а другий – граничне значення, якого вона не набуде. Якщо ж **a**≥**b**, то тіло циклу не виконається жодного разу.

Нарешті, щоб організувати цикл **for**, в якому індексна змінна буде змінюватись з кроком відмінним від **1**, необхідно використовувати функцію **range** з трьома параметрами. Перший параметр задає початкове значення індексної змінної, другий – значення, до якого буде змінюватися індексна змінна (не включаючи його), а третій – крок зміни індексу.

Запис в циклу	Значення індексної змінної **n**
`for n in range(10)`	`0 1 2 3 4 5 6 7 8 9`
`for n in range(1,11)`	`1 2 3 4 5 6 7 8 9 10`
`for n in range(11,1)`	
`for n in range(1,11,2)`	`1 3 5 7 9`
`for n in range(0,11,2)`	`0 2 4 6 8 10`
`for n in range(10,0,-1)`	`10 9 8 7 6 5 4 3 2 1`
`for n in range(11,0,-2)`	`11 9 7 5 3 1`

Команди і функції, що пов'язані з циклом	
Негайно починає наступну ітерацію циклу (**for** або **while**)	`continue`
Оператор **break** достроково перериває цикл	`break`
Блок інструкцій в секції **else** записується в кінці циклу і виконується тільки в тому випадку, якщо вихід з циклу стався без допомоги команди **break**.	`else`
Запис діапазону в циклі **for**	`range()`
Повертає число елементів в зазначеному об'єкті	`len()`

Логічно запитати: команда розгалуження справляється однією конструкцією, а для чого тоді два різних цикли і який з них використовується частіше. Цикл **for** завжди має фіксовану кількість ітерацій. Вже в початковому рядку вказується діапазон зміни індексної змінної або список, елементи якого будуть переглядатись. Тобто все більш-менш консервативно. А цикл **while** схожий на розгалуження з багаторазовою перевіркою умови, тому в залежності від завдання і даних, можуть з'являтись доволі неочікувані результати. Методично вони вивчаються саме в такій послідовності, як їх тут розміщено. Звісно, цикл **for** зручніший для розуміння, тому ним користуються в програмах на порядок частіше, а **while** використовують лише тоді, коли **for** не справляється з поставленою задачею і іншого виходу немає.

Придумана велика кількість завдань де використовується цикл **for**, але, нажаль, мало, які дають можливість правильно зрозуміти і опанувати від азів до подальшого застосування, адже **for** це як плюс в математиці.

Нагадую, що посібник називається "Абетка програмування", тому завдання підібрані тут, мають на меті навчити від самого початкового рівня, а після будуть нові завдання.

Задачі з циклом for

8931 Усі ОК

Задано натуральне число **n**. Порахуйте і виведіть повідомлення **OK** як показано у прикладі.

1 3	1 OK 2 OK 3 OK

Дві версії коду програми, поки майже однакові.	

`n=int(input())`
`for i in range(n) :`
`  print(i+1,'OK')`
	`n=int(input())`
	`for i in range(1,n+1) :`
	`  print(i,'OK')`

8932 Марафон 2

Учасникам забігу надали номери від **a** до **b** та занесли цю інформацію до комп'ютера. Виведіть номери спортсменів.

3 7	3 4 5 6 7

8933 Таймер

Задано натуральне число **n**. Змоделюйте зворотній відрахунок таймера від **n** до **0** як наведено у прикладі.

2	2 sek 1 sek 0 sek

8934 Марафон 3

Учасникам забігу надали номери від **a** до **b** (**a** ≤ **b**), цю інформацію занесли до комп'ютера у зворотному порядку. Вивести номери спортсменів одним рядком через проміжок в спаданні.

3 7	7 6 5 4 3

Починаємо від більшого, зупиняємся на попередньому перед меншим і крок у нас **-1** - сумна реальність. Згадали, що опція **end**=’ ‘ в операторі **print** виводить числа через проміжок?	`a,b=map(int,input().split())`
	`for i in range(b,a-1,-1) :`
	`  print(i,end=' ')`

8935 Непарні на проміжку

На вході програми маємо два цілих числа **a** і **b** (**a < b**), записані в одному рядку через пропуск. Потрібно вивести всі цілі непарні числа, що належать інтервалу [**a**,**b**] - в одному рядку, через проміжок і в зростаючому порядку.

2 7	3 5 7

Індексна змінна **i** в циклі набуває числових значень з інтервалу [**a**,**b**], кожне з них аналізуємо на непарність та виводимо, якщо піймали. Кількість ітерацій циклу **b-a+1**.	`a,b=map(int,input().split())`
	`for i in range(a,b+1) :`
	`  if i%2!=0 :`
	`  print(i,end=' ')`

Щоб пройти інтервал [**a**,**b**] тільки по непарних числах значення **a** гарантовано має бути непарним, а індексна змінна **i** повинна рухатись з кроком **2**. Яка буде кількість ітерацій в цій версії програми?	`a,b=map(int,input().split())`
	`if a%2==0 : a+=1`
	`for i in range(a,b+1,2) :`
	`  print(i,end=' ')`

8936 Парні на проміжку

На вході програми маємо два цілих числа **a** і **b** (**a < b**), записані в одному рядку через пропуск. Потрібно вивести всі цілі парні числа, що належать інтервалу [**a**,**b**] - в одному рядку, через проміжок і в спадному порядку.

2 7	6 4 2

8937 #Смужка

Дано натуральне **n**. Вивести смужку з **n** символів '**#**' як показано у зразку.

7	#######

8938 #Прямокутник

Задано натуральне число **n**. Вивести прямокутник розміром **n** * **3** з символів **#** як показано у прикладі.

2	### ###

Пишемо в змінну **a** три решітки, в циклі **n** раз виконуємо команду вивести **a** в новому рядку.	`n=int(input())`
	`a="#"*3`
	`for i in range(n) : print(a)`

Але ж можна обійтись без циклу. В змінну **a** три решітки плюс **Enter**, повторити **a** **n** раз в команді **print**.	`n=int(input())`
	`a="###"+"\n"`
	`print(a*n)`

8939 #Прямокутник 2

Задано натуральне число **n**. Вивести прямокутник розміром **4** * **n** з символів **#**, як показано у прикладі.

2	## ## ## ##

8940 #Прямокутник 3

Задано два натуральних числа **n** та **m**. Вивести прямокутник розміром **n** * **m** із символів **#**, як показано у прикладі.

2 3	### ###

Аналогічно	`n,m=map(int,input().split())`
	`a="#"*m`
	`for i in range(n) : print(a)`

	`n,m=map(int,input().split())`
	`a="#"*m+"\n"`
	`print(a*n)`

8941 Матриця

Задано два натуральних числа **n** і **m**. Вивести матрицю, що складається з **n** рядків та **m** стовпчиків, заповнену натуральними числами від **1** до **n** * **m**, як показано у прикладі.

2 3	1 2 3 4 5 6

Класичний код - два вкладених цикли - зовнішній керує рядками, внутрішній виводить в рядку змінну **k,** яка за кожну ітерацію збільшується на **1** та набуває значень **1 … n*m**.	`n,m=map(int,input().split())`
	`k=0`
	`for i in range(1,n+1):`
	`  for j in range(1,m+1):`
	`  k+=1`
	`  print(k,end=' ')`
	`  print()`

Але є такий варіант. Один цикл індексна змінна **k** змінюється **1**..**n*m** і виводиться на через проміжок. Якщо досягли кінця рядка, тобто значення **k** стало кратне **m**, то звичайний **print(k)** виконує **Enter**.	`n,m=map(int,input().split())`
	`for k in range(1,n*m+1):`
	`  if k%m==0 : print(k)`
	`  else : print(k,end=' ')`

Задачі з циклом for 2

8942 *Рамка

Для заданого натурального числа **n** вивести горизонтальну прямокутну рамку розміром **3** * **n** із зірочок, заповнену проміжком (як у прикладі).

5	***** * * *****

8943 *Рамка 2

Для заданого натурального числа **n** вивести горизонтальну прямокутну рамку розміром **n** * **3** із зірочок, заповнену проміжком як у прикладі.

5	*** * * * * * * ***

Без коментарів	`n=int(input())`
	`print("***")`
	`for i in range(n-2) : print("* *")`
	`if n>1 : print("***")`

8944 *Рамка 3

Для заданого натурального числа **n** вивести квадратну рамку розміром **n** * **n** із зірочок, заповнену проміжком як показано у прикладі.

5	***** * * * * * * *****

8945 *Рамка 4

Для заданих натуральних чисел **n** та **m** вивести прямокутну рамку розміром **n** * **m** із зірочок, заповнену проміжком як показано у прикладі.

4 7	******* * * * * *******

Універсальний код, який підходить до будь-якої рамки. Два вкладених циклизовнішній керує рядками - внутрішній виводить в рядку зірочку або проміжок. Індексні змінні **i**, **j** набувають значень **1**..**n** та **1**..**m** відповідно. Аналізуючи поточні значення індексів, виводимо зірочку, якщо ми на границі прямокутника з символів або проміжок у протилежному випадку. В кінці кожного рядка не забуваємо виводити **Enter**.

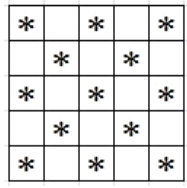
```
n,m=map(int,input().split())
for i in range(1,n+1) :
    for j in range(1,m+1) :
        if i==1 or i==n or j==1 or j==m :
            print("*",end='')
        else : print(" ",end='')
    print()
```

Ще декілька задач, під загальною назвою “шаблон” на вивід прямокутних фігур з зірочок і проміжків. До речі, код попередньої задачі тут теж актуальний, змінюється тільки умова в команді **if** і за розміри відповідає одна змінна **n**, бо матриця квадратна . Тому розв’язки будуть без пояснень. Потрібно оптимально придумати умову, щоб вийшов правильний “шаблон”.

8946 Шаблон

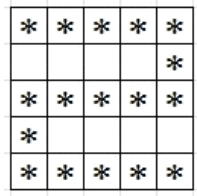
За заданим натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.

```
n=int(input())
for i in range(1,n+1) :
    for j in range(1,n+1) :
        if (i+j)%2==0:
            print("*",end='')
        else : print(" ",end='')
    print()
```



8947 Шаблон 2

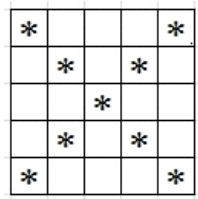
За заданим натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.



8948 Шаблон 3

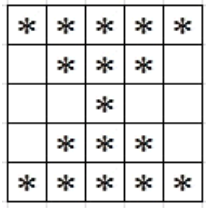
За заданим натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.

```
n=int(input())
for i in range(1,n+1) :
    for j in range(1,n+1) :
        if i==j or i+j==n+1:
            print("*",end='')
        else : print(" ",end='')
    print()
```



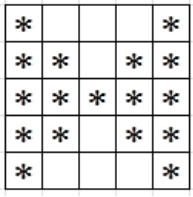
8949 Шаблон 4

За заданим непарним натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.



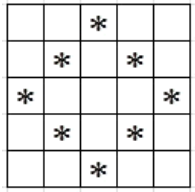
8950 Шаблон 5

За заданим непарним натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.



8951 Шаблон 6

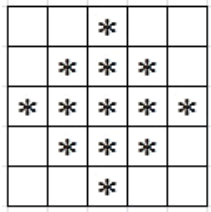
За заданим непарним натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.



```
n=int(input()); k=n//2
for i in range(1,n+1) :
    for j in range(1,n+1) :
        if (i+k==j) or (i==j+k) or (
        i+j==k+2) or (i+j==n+k+1):
            print("*",end='')
        else : print(" ",end='')
    print()
```

8952 Шаблон 7

За заданим непарним натуральним числом **n** вивести зображення розміром **n** * **n**, утворене символами зірочка та проміжок як показано у прикладі.



МАСИВИ АБО СПИСКИ

Масив – це дещо схоже на багатоквартирний будинок, аркуш табличного процесора Excel або ж оперативну пам'ять комп'ютера, коли по заданому адресу завжди знаходимо деякий об'єкт, подібний до всіх інших. Причому всі елементи масиву, зазвичай (але необов'язково), мають однаковий тип і позначаються одним ім'ям, різні тільки індекси, які починаються з **0** і йдуть підряд. Однакові значення можуть входити в масив багаторазово.

Перегляд елементів масиву найчастіше здійснюється командою циклу з параметром **for** двома способами:

- по елементах **for q in a**, де **q** пробігає послідовно всі елементи масиву **a**;
- по індексах **for i in range**(**n**), де **a**[**i**] (**i**=**0**..**n-1**) пробігає послідовно всі **n** елементів масиву **a**;

Введення масиву, елементи якого записані в стрічку через проміжок можна виконати такими інструкціями Python:

- **a** = **list**(**input**()**.split**()) – введення текстового масиву **a**;
- **a** = [**v for v in input**()**.split**()] – аналогічно попередньому;
- **a** = **list**(**map**(**int, input**()**.split**()))– введення масиву цілих чисел **a**;
- **a** = [**int**(**i**) **for i in input**()**.split**()] – аналогічно попередньому.

Виведення всього масиву виконується командою **print**(**a**), для поелементного виведення в стрічку через проміжок або в стовпчик команда **print**() викликається в циклі **for** для кожного елементу масиву.

Для кращого розуміння пропонується програма, яка дозволяє ввести з консолі масив цілих чисел. А далі виводить масив, сортує елементи у зростанні і виводить їх в стовпчик. І ще, розвертає елементи масиву в протилежному напрямку і виводить елементи масиву в стрічку через проміжок. Кількість елементів масиву в такій версії програми вводити не потрібно.

```
a=list(map(int,input().split()))
print(a)
a.sort()
for q in a : print(q)
a.reverse()
for q in a : print(q,end=' ')
```

Методи і функції пов'язані з масивами	
Створення списку об'єктів (масиву).	`x=list()`
Визначення довжини списку (масиву).	`len(x)`
Додає елемент **z** в кінець списку **x**.	`x.append(z)`
Видаляє перший елемент у списку **x**, який має значення **z**.	`x.remove(z)`
Вставляє на **i**-ий елемент значення **z**.	`x.insert(i,z)`
Видаляє **i**-ий елемент і повертає його. Якщо індекс не вказано, видаляється останній.	`x.pop(i)`
Повертає положення першого елемента, значення якого **z**.	`x.index(z)`
Повертає кількість елементів, що дорівнюють **z**.	`x.count(z)`
Сортує список в неспадному порядку.	`x.sort()`
Розвертає елементи списку.	`x.reverse()`
Очищає список, видаляє всі його елементи.	`x.clean()`
Перетворення списку у множину.	`set(x)`

Задач, що використовують масиви в своїх розв'язках на порядок більше ніж завдань, з яких можна засвоїти початкові поняття: введення, виведення елементів, підрахунок суми, добутку, пошук , зміна порядку, сортування елементів масиву. Адже, громіздка програма для розв'язання складної задачі повинна складатись з максимально лаконічно і точно описаних кубиків коду, тоді можна говорити про оптимальність і ефективність алгоритму. Не вдається створити гарний будинок з бракованих матеріалів, він швидко зруйнується від холоду, дощу і вітру.

На сайті знаходимо багато задач для покращення навиків роботи з одновимірними, а також двовимірними масивами. Можна запропонувати до уваги список задач з розділу «Просто масиви» з книги «Практикум програмування Python / C++ на e-olymp.com». Корисні завдання, скоріше розраховані на використання готових процедур і функцій **Python** для обробки списків. Навіть,

якщо розглянути тільки табличку вище, можна переконатись, що **Python** надає масу засобів для ефектної роботи в такому напрямку.

7829	Сума елементів масиву
7830	Найбільший елемент масиву
7832	Кількість максимальних
7831	Сума без максимального
7841	Непарні елементи
7842	Парні індекси
7843	Парні індекси
7844	Сусіди одного знаку
7845	Більші своїх сусідів
7846	Найбільший елемент
7847	Кількість різних елементів
7848	Переставити сусідні
7849	Обміняти max і min
7850	Обміняти max і min
7833	Більші за середнє арифметичне
7834	Два найбільших
5337	Різні-різні
2440	N-те найбільше значення
5721	Пошук елемента
3935	Реверс
2099	Два масиви

Не має границь фантазія і творчість у справжніх програмістів, як і немає такого програмного коду, якого не можливо було б покращити і оптимізувати. А якщо не знайшлося потрібного інструменту у списку стандартних процедурі функцій, отже потрібно вміти дещо написати вручну. Саме про це наступний набір задач в «Абетці програмування», тут і зараз можна познайомитись з доповненням і продовженням розділу «Просто масиви».

Задачі на масивах

8953 Вивести масив

Задано масив з **n** цілих чисел. Вивести його елементи в стовпчик, не змінюючи початковий порядок.

4 5 0 -7 2	5 0 -7 2

Читаємо значення **n** та цілі числа з одного рядка через проміжок, як елементи масиву **a**. Виводимо значення елементів циклом **for**, переглядаючи їх по елементах.	`n=int(input())`
	`a=[int(i) for i in input(`
	`).split()]`
	`for j in a : print(j)`

8954 Вивести масив 2

Програма має прочитати з консолі масив з **n** цілих чисел та вивести елементи масиву в одному рядку через проміжок, змінивши початковий порядок на протилежний.

7 0 4 7 -4 0 3 -2	-2 3 0 -4 7 4 0

Читаємо значення **n** та утворюємо масив **a** з **n** нулів. Вводимо значення елементів масиву – кожне з окремого рядка, переглядаючи їх по індексах. Виводимо масив в один рядок в оберненому порядку.	`n=int(input())`
	`a=[0]*n`
	`for i in range(n) :`
	`  a[i]=int(input())`
	`for i in range(n-1,-1,-1):`
	`  print(a[i], end=' ')`

8955 Вивести масив 3

Задано масив з **n** цілих чисел. Вивести тільки додатні його елементи, не змінюючи їх початковий порядок або повідомлення **NO** , якщо їх немає .

7 -2 5 4 -3 7 -3 0	3 5 4 7
5 -2 -1 0 -1 -5	NO

8956 Вивести масив 4

Програма має прочитати з консолі масив з **n** цілих чисел та вивести тільки від'ємні елементи цього масиву одним рядком через проміжок, змінивши початковий порядок на протилежний або повідомлення **NO** , якщо їх немає.

7 -2 5 4 -3 7 -1 0	3 -1 -3 -2
5 2 1 0 1 5	NO

Стандартний ввід масиву **a**. Шукаємо кількість від'ємних в змінну **k**. Виводимо **NO**, якщо не знайшли або виводимо **k** і потім всі від'ємні елементи, переглядаючи їх по індексах в оберненому порядку.	`n,k=int(input()),0`
	`a=[int(j) for j in input(`
	`).split()]`
	`for j in a :`
	`  if j<0 : k+=1`
	`if k==0 :`
	`  print('NO')`
	`else :`
	`  print(k)`
	`  for i in range(n-1,-1,-1) :`
	`  if a[i]<0 :`
	`  print(a[i], end=' ')`

8957 Вивести масив 5

Програма має прочитати з консолі масив з **n** цілих чисел та вивести тільки парні елементи цього масиву одним рядком через проміжок, змінивши початковий порядок на протилежний або повідомлення **NO** , якщо їх немає.

7 -2 5 4 -3 7 -1 0	3 0 4 -2

8958 Вивести масив 6

Програма має прочитати з консолі масив з **n** цілих чисел та вивести тільки елементи з непарними індексами одним рядком через проміжок, не змінюючи початковий порядок або повідомлення **NO** , якщо їх немає.. Нумерація починається з **0**.

7 -2 5 4 -3 7 -1 0	3 5 -3 -1

Стандартний ввід масиву **a**. Елементів з непарними індексами буде **k=n//2**. Виводимо **NO**, якщо **k=0** або виводимо **k**, а після всі шукані елементи, переглядаючи їх по індексах від **1** до **n** з кроком **2**.	`n,k=int(input()),0`
	`a=[int(j) for j in input(`
	`).split()]`
	`k=n//2`
	`if k==0 :`
	`  print('NO')`
	`else :`
	`  print(k)`
	`  for i in range(1,n,2) :`
	`  print(a[i], end=' ')`

8959 Різниця між найбільшим і найменшим

Задано **n** цілих чисел. Вивести різницю між найбільшим і найменшим числом.

7 0 -7 -13 14 -2 13 13	27

8960 Крім найменших і найбільших

З консолі вводиться масив з **N** цілих чисел. Програма повинна знайти суму елементів масиву, не враховуючи всіх його найменших і найбільших елементів.

7 6 2 7 1 7 1 2	10

Стандартний ввід масиву **a**. Далі скористаємося стандартними функціями **Python** для масивів. Особливий випадок, коли **min=max**.	`n=int(input())`
	`a=[int(j) for j in input().split()]`
	`l,m=min(a),max(a)`
	`p=a.count(l)`
	`q=a.count(m)`
	`if l==m :`
	`  print(sum(a)-l*p)`
	`else :`
	`  print(sum(a)-l*p-m*q)`

8961 Перший найменший

Задано масив з **n** цілих чисел. Знайдіть найменший елемент масиву, що зустрічається найпершим та поміняйте його з першим елементом у масиві, не змінюючи порядок інших елементів.

7 6 -3 5 -5 -4 7 -5	-5 -3 5 6 -4 7 -5

8962 Крайній найбільший

Задано масив з **n** цілих чисел. Знайдіть найбільший елемент масиву, що зустрічається останнім та поміняйте його з крайнім елементом у масиві, не змінюючи порядок інших елементів.

7 6 -3 7 -4 7 4 -5	6 -3 7 -4 -5 4 7

Після стандартного вводу масиву **a** знаходимо номер крайнього найбільшого елементу масиву. Виконуємо обмін значень у вказаних елементів масиву **a**. Виявляється вивід масиву можна однією командою **print(*a)** – вивести вміст пам'яті, на яку вказує ім'я **a**.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`m=max(a)`
	`for i in range(n) :`
	`  if a[i]==m : k=i`
	`a[k]=a[n-1]; a[n-1]=m`
	`print(*a)`

8963 Найменші вліво

Задано масив з **n** цілих чисел. Перемістити всі найменші елементи на початок масиву, не змінюючи порядок інших.

7 6 -3 -7 4 -7 -4 5	-7 -7 6 -3 4 -4 5

8964 Найбільші вправо

Задано масив з **n** цілих чисел. Перемістити всі найбільші елементи в кінець масиву, не змінюючи порядок інших.

7 6 -2 7 1 7 -1 2	6 -2 1 -1 2 7 7

Задачі на масивах 2

8965 Початкові значення елементів

До кожного елементу масиву з **n** цілих чисел додали його найменший елемент. Отримані значення подали на вхід програми, не змінюючи їх порядок. Вам потрібно відновити початкові значення елементів масиву.

7 1 7 -1 -4 2 1 2	3 9 1 -2 4 3 4

Найменший збільшився на найменший, але лишився найменшим, тому знайдемо найменший, розділивши його навпіл – це буде найменший елемент початкового масиву **a**. Лишається кожен елемент зменшити на цей найменший і вивести початковий масив.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`l=min(a)//2`
	`for i in range(n): a[i]-=l`
	`print(*a)`

8966 Початковий порядок

Масив з **N** цілих чисел подається на вхід програми в зворотному порядку, тобто від останнього до першого елемента. Вам потрібно відновити початковий порядок елементів в масиві.

7 6 0 -7 2 -3 5 7	7 5 -3 2 -7 0 6

Стандартний ввід масиву **a**. Далі скористаємося стандартним методом **Python**, який розвертає масив у зворотному порядку. На кінець виводимо масив.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`a.reverse()`
	`print(*a)`

8967 Початкові значення елементів 2

До кожного елементу масиву з **N** цілих чисел додали його найбільший і відняли найменший елемент. Отримані значення подали на вхід програми, не змінюючи їх порядок. Вам потрібно відновити початкові значення елементів масиву.

7 7 5 14 9 7 5 8	-2 -4 5 0 -2 -4 -1

8968 Початкові значення елементів 3

До кожного додатного елементу масиву з **N** цілих чисел додали його найбільший елемент, а до від'ємного - найменший. Отримані значення подали на вхід програми, не змінюючи їх порядок. Вам потрібно відновити початкові значення елементів масиву.

7 14 -10 10 8 11 -9 -6	7 -5 3 1 4 -4 -1

8969 Початкові значення елементів 4

До кожного елемента масиву з **N** цілих чисел додали суму всіх елементів з меншими індексами. Оновлений масив подається на вхід програми. Вам потрібно відновити початкові значення елементів масиву.

7 1 5 2 0 2 5 9	1 4 -3 -2 2 3 4

Лишився “незайманим” тільки перший елемент масиву, а щоб знайти другий елемент від нього потрібно відняти перший, тобто для відновлення кожний потрібно зменшити на попередній. Перегляд елементів слід виконати з кінця масиву, тобто спочатку знайти останній і тд. На кінець виводимо масив.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`for i in range(n-1,0,-1):`
	`  a[i]-=a[i-1]`
	`print(*a)`

8970 Відновлення масиву

Масив з **N** цілих чисел перегрупували, розмістивши спочатку елементи, що були на парних місцях, а лише потім елементи з непарними індексами, не змінюючи їх наступності у масиві. Рахуємо, що нумерація починається з **0**. Оновлений масив подається на вхід програми. Вам потрібно відновити початковий порядок елементів.

7 6 0 -7 -3 -3 5 6	6 -3 0 5 -7 6 -3

8971 Без повторень

З заданого масиву цілих чисел потрібно видалити всі дублювання елементів. Тобто з декількох однакових елементів в масиві залишається тільки елемент з найменшим індексом.

7 0 1 -2 1 0 0 3	0 1 -2 3

Ввід масиву **a** та порожній масив **b**. По елементах передивляємось масив **a** та дописуємо елемент в масив **b**, якщо його там ще немає. На кінець виводимо масив **b**.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`b=[]`
	`for j in a :`
	`  if not (j in b) :`
	`  b.append(j)`
	`print(*b)`

8972 Модні елементи

В заданому масиві цілих чисел потрібно вивести тільки такі елементи, що повторюються декілька разів. Тобто числа, що зустрічаються один раз не виводяться взагалі, а з декількох однакових елементів в масиві вибираємо один з найменшим індексом. Якщо таких елементів немає потрібно вивести повідомлення **NO**.

7 0 1 -2 1 0 0 3	0 1
5 -1 -2 0 2 1	NO

Ввід масиву **a** та порожня множина **b**. По елементах передивляємось масив **a.** Якщо поточний елемент масиву **a** повторюється (кількість появи більша **1**) і його ще немає в множині **b**, то додаємо його туди і виводимо у відповідь. Якщо ж множина **b** лишилась порожньою, то виводимо повідомлення **NO**.	`n=int(input())`
	`a=[int(j) for j in input(`
	`).split()]`
	`b=set()`
	`for j in a :`
	`  if a.count(j`
	`  )>1 and not(j in b):`
	`  b.add(j)`
	`  print(j, end=' ')`
	`if len(b)==0 : print('NO')`

8973 Без повторень 2

З заданого масиву цілих чисел потрібно видалити всі дублювання елементів. Тобто з декількох однакових елементів в масиві залишається тільки елемент з найбільшим індексом.

7 0 1 -2 1 0 0 3	-2 1 0 3

8974 Модні елементи 2

В заданому масиві цілих чисел потрібно вивести тільки такі елементи, що повторюються декілька разів. Тобто числа, що зустрічаються один раз не виводяться взагалі, а з декількох однакових елементів в масиві вибираємо один з найбільшим індексом. Якщо таких елементів немає потрібно вивести повідомлення **NO**.

7 0 1 -2 1 0 0 3	1 0

8975 Модні елементи 3

В заданому масиві цілих чисел потрібно знайти елементи, що повторюються максимальну кількість разів. Спочатку потрібно опублікувати максимальну частоту дублювання в масиві, а вже потім елементи, які мають таку кількість повторень. Порядок виведення визначається першою появою модного елемента на вході програми.

9 0 -1 -2 1 1 0 1 0 3	3 0 1

8976 Модні елементи 4

В заданому масиві цілих чисел потрібно знайти елементи, що повторюються максимальну кількість разів. Спочатку потрібно опублікувати максимальну частоту дублювання в масиві, а вже потім елементи, які мають таку кількість повторень. Порядок виведення визначається модним елементом з найбільшим індексом на вході програми.

9 0 -1 1 0 1 1 -2 0 3	3 1 0

Аналізуємо частоту появи кожного елементу масиву **a** та знаходимо максимальну в змінну **h**. Розвертаємо масив **a** в оберненому напрямку. В порожній масив **b** додаємо без повторів елементи, які мають в масиві **a** частоту **h**. Виводимо масив **b** в оберненому порядку.	`n,h=int(input()),0`
	`a=[int(j) for j in input(`
	`).split()]`
	`for j in a :`
	`  if a.count(j)>h :`
	`  h=a.count(j)`
	`print(h)`
	`b=[]`
	`a.reverse()`
	`for j in a :`
	`  if a.count(j)==h and not(j`
	`  in b):`
	`  b.append(j)`
	`print(*b[::-1])`

Обробка рядків

В цьому розділі перераховані далеко не всі, а лише найпоширеніші функції і методи рядкових величин, що застосовуються у конкретних задачах. Якщо необхідно більше інформації, то в пригоді стане розділ довідки String Methods і клавіша F1 – її ще ніхто не відміняв. Також відповідний список з поясненнями можна знайти в Інтернеті на відомих сайтах або за допомогою Google.

Ось деякі позначення в таблиці:

- **i** – стартовий індекс в зрізі (початкове значення 0);
- **j** – кінцевий індекс (або перший, який не входить в зріз);
- **h** – крок зміни індексу;
- **s** – рядкова величина ;
- **u** – рядкова величина шаблон;
- **v** – рядкова величина для заміни;
- **c** – деякий символ;
- **n** – деяке ціле число.

Довідник Python

Методи і функції пов'язані з рядками	
Конкатенація (додавання рядків)	`s1 + s2`
n повторень рядка **s**	`s * n`
i-й символ рядка **s**, відлік починається з 0	`s[i]`
зріз рядка **s**	`s[i:j:h]`
довжина рядка **s**	`len(s)`
пошук підрядка в рядку, повертає номер першого входження або **-1**	`s.find(u,i,j)`
пошук підрядка в рядку, повертає номер останього входження або **-1**	`s.rfind(u,i,j)`
заміна шаблонів **u** на **v** в рядку **s**	`s.replace(u, v)`
розбиття рядка по розділовому символу **c**	`s.split(c)`
чи складається рядок **s** тільки з цифр	`s.isdigit()`
чи складається рядок **s** тільки з букв	`s.isalpha()`

збірка рядка із списка з роздільником **c**	**c.join(список)**
видалення пропусків або **Enter** (символа **c**) на початку і в кінці рядка	**s.strip([c])**
код ASCII символу **c**	**ord(c)**
символ, код ASCII якого **n**	**chr(n)**
кількість непересічних входжень підрядка	**s.count(u,i,j)**

Тренувальні вправи для обробки текстових рядків підшукуються на сайті непросто. Тому крім завдань, які розглянуті в цьому збірнику хочу запропонувати до розгляду дванадцять задач з оновленої "Абетки" з номерами 9912..9923 – з них варто почати. А також декілька завдань з книги «Практикум програмування Python / C++ на e-olymp.com».

4718 Привіт, Гаррі!
622 Одиниці
3254 01110001, ось запитання
7340 Поле-чудес
1427 Калькулятор
4724 Робимо зрізи
963 Перестановка слів
494 Голосні
4722 Квадрат числа
4726 Перше та останнє входження
5049 Видали пропуски
2164 Шифр Юлія
2704 Римська система числення

Задачі про рядки

8977 Робимо зрізи 2

Задано рядок **s** довжиною більше десяти символів. Враховуємо, що перший символ рядка має індекс **0**. Знайти і вивести отаке.

– Слово, що утворюють третій, сьомий і одинадцятий символ рядка **s**.

– Слово, що утворюють перший і два останніх символу рядка **s**.

– Слово, що утворюють сім перших символів рядка **s**.

– Рядок **s** без чотирьох перших символів.

– Слово, що утворюють усі символи з непарними індексами, починаючи з другого символу рядка **s**.

– Довжину слова з попереднього пункту.

– Рядок **s** у зворотному порядку.

abrakadabra	rda ara abrakad kadabra baaar 5 arbadakarba

Коментарі зайві	`a=input().strip()`
	`print(a[2]+a[6]+a[10])`
	`print(a[0]+a[-2:])`
	`print(a[:7])`
	`print(a[4:])`
	`print(a[1::2])`
	`print(len(a[1::2]))`
	`print(a[::-1])`

8978 Робимо зрізи 3

Задано рядок **s** довжиною більше десяти символів. Враховуємо, що перший символ рядка має індекс **0**. Знайти і вивести отаке.

– Слово, що утворюють другий, четвертий і десятий символ рядка **s**.

– Слово, що утворюють перший, другий і останній символ рядка **s**.

– Слово, що утворюють п'ять останніх символів рядка **s**.

– Рядок **s** без чотирьох останніх символів.

– Слово, що утворюють усі символи з парними індексами, починаючи з першого символу рядка **s**.

– Довжину слова з попереднього пункту.

– Усі символи рядка **s** через один у зворотному порядку, починаючи з останнього.

abrakadabra	bar aba dabra abrakad arkdba 6 abdkra

8979 Лише одна буква

Вивести тільки маленькі латинські літери **a**, що зустрічаються в заданому рядку.

Abrakadabra	aaaaa
qwertyuiopsdfghjkl	-1

Ввід в змінну **w** текстового рядка – тільки видимі символи. В змінну **n** кількість латинських букв "**a**" в змінній **w.** Якщо **n!=0**, то виводимо **n** символів "**a**", в протилежному випадку **-1**. Отак, в команді **print** можна сховати розгалуження.	`w=input().strip()`
	`n=w.count("a")`
	`print(n*"a"+(n==0)*'-1')`

8980 Кількість речень

Визначити кількість речень у заданому тексті. Вважайте, що речення закінчується одним із символів "**.**"(крапка), "**!**"(знак оклику) або "**?**"(знак запитання), а наборів знаків типу "**...**", "**!!!**" і "**?!**" в тексті немає.

Hello, world!	1

8981 Індекси проміжків

Знайдіть індекси першого та останнього проміжку в заданому рядку, враховуючи, що перший символ має індекс **0**.

I am programming on Python.	1 19
abrakadabra	-1

8982 Індекси літери

Виведіть усі індекси маленької латинської літери **a** в заданому рядку, враховуючи, що перший символ має індекс **0**.

abrakadabra	0 3 5 7 10
Hello, world!	-1

Стандартний ввід в змінну **w** текстового рядка – тільки видимі символи. Якщо немає букв "**a**" в змінній **w**, то виводимо **-1**. В протилежному випадку переглянемо **w** і виводимо індекси маленької латинської літери "**a**".

```
w=input().strip()
if w.count("a")==0:
    print(-1)
else:
    for i in range(len(w)):
        if w[i]=="a":
            print(i, end=' ')
```

8983 Кількість цифр

Задано рядок, що містить вираз з цифр, знаків арифметичних операцій і дужок. Знайдіть кількість використаних у виразі цифр.

9*8+76-54/3+2**10	10

8984 Кількість арифметичних операцій

Задано рядок, що містить правильний вираз з цифр, дужок і знаків арифметичних операцій, до прикладу, таких як в **Python**: + (додавання), **-** (віднімання), * (множення), ** (піднесення до степені), / (ділення), // (ділення націло), % (залишок у діленні націло). Знайти кількість використаних у виразі арифметичних операцій.

9*8+76-54/3+2**10	6

Ввід в змінну **w** текстового рядка – тільки видимі символи. Замінимо в змінній **w** знак степінь на множення, а ділення з остачею на просто ділення. Кількість арифметичних операцій в виразі не змінилась, але рахувати їх стало простіше.

```
w=input().strip()
w=w.replace('**','*')
w=w.replace('//','/')
a=w.count('+')
b=w.count('-')
c=w.count('*')
d=w.count('/')
e=w.count('%')
print(a+b+c+d+e)
```

8985 Видалення букви

Видалити всі маленькі латинські літери **a** у заданому рядку.

abrakadabra	brkdbr

Стандартний ввід в змінну **w** текстового рядка – тільки видимі символи. Змінна **u** для формування відповіді поки що порожня. Переглянемо **w**, всі символи, крім маленької латинської літери "**a**" додаємо в змінну **u** і вивід.	`w=input().strip()`
	`u=''`
	`for i in range(len(w)):`
	`  if w[i]!="a":`
	`  u+=w[i]`
	`print(u)`

8986 Видалення фрагменту

Дано рядок, що складається з латинських літер та проміжків. Видаліть в ньому всі символи з індексами від **n** до **m** включно. Нумерація символів з **0**.

Abrakadabra 3 6	abrabra
This is my lovely beautiful house 7 16	This is beautiful house

8987 Заміна символів

У рядку, що складається з латинських літер і проміжків, замінити всі символи **a** на **b** і навпаки.

abrakadabra	barbkbdbarb

8988 Заміна символів 2

У рядку, що складається з латинських літер і проміжків, замінити кожну послідовність символів "**ab**" на "**ups**".

abrakadabra	upsrakadupsra

Стандартний ввід в змінну **w** текстового рядка. Змінна **u** для формування відповіді поки що порожня. Перегляд заданого рядка **w** курує цикл **while**, подумайте чому. Якщо зустрічаємо "**ab**", то в змінну **u** додаємо "**ups**".	`w=input().strip()`
	`u,i="",0`
	`while i<len(w) :`
	`  if w[i:i+2]=="ab":`
	`  u+="ups"; i+=1`
	`  else : u+=w[i]`
	`  i+=1`
	`print(u)`

Задачі про рядки 2

8989 Подвоєння символа

Задано рядок, що складається з англійських букв і проміжків. Потрібно в ньому подвоїти всі символи **a**.

abrakadabra	aabraakaadaabraa

8990 Подвоєння голосних

Рядок складається з маленьких латинських літер і проміжків. Подвійте в ньому всі голосні літери, тобто літери **a**, **e**, **i**, **o**, **u** та **y**.

welcome to python	weelcoomee too pyythoon

Стандартний ввід в змінну **w** текстового рядка. Відповідь буде сформована у **u**, яка поки що порожня. В масив символів **h** записуємо всі голосні. Переписуємо всі літери **w** у **u**, подвоюючи їх, якщо це символ з **h**.

```
w,u=input(),""
h=["a","e","i","o","u","y"]
for i in range(len(w)) :
    if w[i] in h :
        u+=w[i]*2
    else : u+=w[i]
print(u)
```

8991 Подвоєння символів

Рядок складається з маленьких латинських літер, розділових знаків та проміжків. Подвійте в ньому всі латинські літери.

welcome to python!	wweellccoommee ttoo ppyytthhoonn!

8992 Без повторень 3

Задано рядок, що складається з англійських букв, розділових знаків і проміжків. Потрібно видалити будь-які повторення символів, тобто однакові символи, що йдуть підряд замінити одним.

WWWellcccooomee ttoo PPyythooonn!!!	Welcome to Python!

Переглядаючи заданий рядок **w**, поточний його символ **v** порівнюємо попередньо виведеним символом **u** – виводимо **v**, якщо не співпадають та змінюємо попередній символ **u**.	`w,u = input().strip(),""`
	`for v in w:`
	`  if u != v:`
	`  print(v, end = "")`
	`  u = v`

8993 Скласти слово

Чи можна з літер слова **a** скласти слово **b**, причому кожну літеру можна використати тільки один раз?

abrakadabra dakar	Ok
abrakadabra kabak	No

8994 Скласти слово 2

Чи можна з літер слова **a** скласти слово **b**, причому кожну з літер можна використовувати декілька разів?

abrakadabra kabak	Ok
abrakadabra kakadu	No

Всі літери рядка **b** шукаємо в текстовому рядку **a**, якщо не знаходимо навіть одного символу, то відповідь буде **No**.	`a=list(input())`
	`b=list(input())`
	`h="Ok"`
	`for q in b :`
	`  if not (q in a) : h="No"`
	`print(h)`

8995 Модні символи

На вході програми маємо рядок, що складається тільки з англійських букв. Потрібно вивести літери, що повторюються декілька разів. Отже букви, що зустрічаються один раз пропускаємо, а з декількох однакових символів виводимо тільки один з найменшим індексом. Якщо літери в рядку різні, то вивести повідомлення **NO**.

abrakadabra	Abr
abcdefghijklpqrstuvwxyz	NO

8996 Без повторень 4

На вхід програми подається рядок символів, що складається тільки з англійських букв. Потрібно вивести цей рядок без повторень його літер, тобто кожен символ виводиться тільки один раз - коли він зустрічається вперше на вході програми.

abrakadabra	Abrkd

Переписуємо літери рядка **w** в новоутворений рядок **u** без повторень, тобто символ додаємо тільки тоді, якщо такого в **u** ще немає.	`w,u=input().strip(),""`
	`for a in w :`
	`  if not(a in u) : u+=a`
	`print(u)`

8997 Модні символи 2

У заданому рядку символів, що складається тільки з англійських букв, потрібно знайти літери, що повторюються максимальну кількість разів. Спочатку потрібно опублікувати максимальну частоту дублювань у рядку, а вже потім символи, які мають таку кількість повторень. Порядок виведення визначається першою появою модної літери на вході програми.

HelloPython	2 l o

8998 Наймолодший символ

У рядку, що складається з латинських літер, знайти та вивести наймолодшу літеру (символ, що має найменший ASCII код), а також кількість її повторень у рядку.

abrakadabra	a 5

Без коментарів	`a=input().strip()`
	`u=min(a)`
	`print(u,a.count(u))`

8999 Найстарший символ

Є рядок, що складається з латинських літер. Знайдіть та виведіть найстаршу літеру (символ, що має найбільший ASCII код), а також кількість її повторень у рядку.

SOS	S 2

9000 Впорядкування

Задано рядок, що складається з маленьких латинських літер. Виведіть усі його літери, впорядковані за алфавітом (в порядку зростання ASCII кодів його символів).

abrakadabra	aaaaabbdkrr

Без коментарів	`a=list(input())`
	`a.sort()`
	`print("".join(a))`

Навчально-методичне видання

Сергій МАТВІЙЧУК

АБЕТКА ПРОГРАМУВАННЯ
на Python

збірник задач вміщує пояснення алгоритмів та
коди програм до 200 окремих задач
з сайту **e-olymp.com**