Управління освіти і науки
Черкаської обласної державної адміністрації

Шполянський районний методичний кабінет

Шполянська загальноосвітня школа І-ІІІ ступенів №1
Шполянської міської ради об'єднаної територіальної громади
Черкаської області

Шкарбан Вадим Миколайович

PYTHON
Step by step

Схвалено на засіданні методичної ради
Шполянського районного методичного кабінету
протокол №4 від 29.03.2018 року

Рецензент:

Музика Світлана Анатоліївна, методист районного методичного кабінету

Шкарбан Вадим Миколайович, вчитель інформатики, спеціаліст вищої категорії, «Старший вчитель». Python. Step by step – 63 сторінки, 2018р.

У посібнику розглянуто основи однієї з найпопулярніших сучасних мов програмування – Python, що, завдяки простому і зрозумілому синтаксису, особливо пасує для новачків. Крок за кроком на простих прикладах та з допомогою цікавих завдань ви зможете пізнати тонкощі мови Python, навчитеся використовувати основні елементи цієї мови – списки, функції, модулі, малювати вбудованими інструментами Python.

Подані матеріали формують ключові компетентності та компетенції шляхом виконання навчально-пізнавальних та практично зорієнтованих завдань відповідно до змістових предметних ліній.

Посібник призначений для вчителів та викладачів інформатики, орієнтований на теми програми з інформатики для 8 класу та учнів загальноосвітніх навчальних закладів.



ЗМІСТ

Історія мов програмування. Компіляція та інтерпретація 5

Програма. Мова програмування 5

Основні етапи розвитку мов програмування 5

Розмаїття мов програмування 7

Трансляція 7

Знайомство з Python і середовищами програмування 9

Історія 9

Особливості мови програмування Python 9

Дзен Пітона 10

Крок 1. Налаштування редактора для написання програм 12

Крок 2. Створення першої програми 14

Крок 3. Коментарі 16

Крок 4. Числа і математика 18

Крок 5. Змінні і імена 20

Крок 6. Введення, виведення 22

Крок 7. Прості типи даних 24

Задачі на самостійне виконання 27

Крок 7. Вбудовані функції та підключення бібліотек 28

Задачі на самостійне виконання. 31

Крок 8. Логічний тип 33

Задачі на самостійне виконання 36

Крок 9. Умовне розгалуження 37

Задачі на самостійне виконання 43

Крок 10. Модуль random 45

Крок 11. Цикли 47

Цикл з лічильником. 47

Задачі на самостійне виконання 51

Умовні цикли 53

Задачі на самостійне виконання 56

Переривання роботи циклів 56

Крок 11. Відображення графічних об’єктів засобами мови програмування 58

Список використаних джерел: 63



Історія мов програмування. Компіляція та інтерпретація

Програма. Мова програмування

Програму можна представити як набір послідовних команд (алгоритм) для об'єкта (виконавця), який повинен їх виконати для досягнення певної мети. Так умовно можна запрограмувати і людину, якщо скласти їй інструкцію "як приготувати млинці", а вона почне чітко її виконувати. При цьому інструкція (програма) для людини буде написана на так званій природній мові (українській, англійській, тощо)

Зазвичай прийнято програмувати не людей, а обчислювальні машини, використовуючи при цьому спеціальні мови. Використання особливих мов викликано тим, що машини не в змозі розуміти наші (людські) мови. Для "інструктування" машин призначені різноманітні мови програмування, які характеризуються синтаксичною однозначністю (наприклад, в них не можна міняти місцями певні слова) і обмеженістю (строго певний набір слів і символів).

Основні етапи розвитку мов програмування

Перші програми писалися на машинній мові, тому що для ЕОМ того часу ще не існувало розвиненого програмного забезпечення, а машинна мова – це єдиний спосіб взаємодії з апаратним забезпеченням комп'ютера. Кожну команду машинної мови безпосередньо виконує той чи інший електронний пристрій. Дані і команди програмісти записували в цифровому вигляді (наприклад, в шістнадцятковій або двійковій системах числення). Зрозуміти програму на такій мові дуже складно, крім того, навіть невелика



програма складалась з безлічі рядків коду. Ситуація ускладнювалася ще й тим, що кожна обчислювальна машина розуміє лише свою машинну мову.

Людям, на відміну від машин, більш зрозумілі слова, ніж набори цифр. Прагнення людини оперувати словами а не цифрами призвело до появи *асемблерів*. Це мови, у яких замість чисельного позначення команд і областей пам'яті використовуються словесно-буквені.

При цьому з'являється проблема: машина не в змозі зрозуміти набори букв. Необхідний якийсь перекладач на її рідній машинній мові. Тому, починаючи з часів асемблерів, під кожну мову програмування створюються *транслятори* – спеціальні програми, що перетворюють програмний код з мови програмування на машинний код. Асемблери на сьогоднішній день продовжують використовуватися (в системному програмуванні – низькорівневі інтерфейси операційних систем, частини драйверів та ін.)

Після асемблерів почались розвиватись мови так званого високого рівня. Для цих мов знадобилося розробляти більш складні транслятори, тому що мови високого рівня набагато більш зручні для людини, але відповідно менш зрозумілі для обчислювальної машини. На відміну від асемблерів, які залишаються прив'язаними до своїх типів машин, мови високо рівня володіють *переносимістю*. Тобто, написавши один раз програму, програміст міг виконати її на *будь-якій* машині.

Наступним значущим кроком була поява об'єктно-орієнтованих мов програмування. За допомогою таких мов програміст як би оперує



віртуальними об'єктами. На сьогоднішній день, реалізація великих і складних проектів здійснюється в основному за допомогою об'єктно-орієнтованого програмування.

Розмаїття мов програмування

На сьогоднішній день існує величезна кількість мов програмування, які і розрізняються і схожі між собою. Причина такого явища стає зрозуміла, якщо уявити ту кількість і різноманітність завдань, які на сьогоднішній день вирішується за допомогою обчислювальної техніки. Для вирішення різних завдань потрібні різні інструменти (тобто мови програмування).

Чимало програмістів намагалися в минулому і намагаються зараз придумати свою мову програмування, що володіє тими чи іншими перевагами. Хоча переважна більшість програмістів в даний час витрачають величезну кількість часу на вивчення вже існуючого арсеналу інструментів.

Все існуюче різноманіття мов можна умовно класифікувати за різними критеріями. Наприклад, за типом розв'язуваних завдань (мови системного або прикладного призначення, мови для web-розробки та ін.)

Трансляція

Раніше було сказано, що для перекладу коду з однієї мови програмування (наприклад, мови високого рівня) на іншу (наприклад, машинну мову) потрібна спеціальна програма – транслятор.



Механізм цього перекладу вельми складний, при цьому виділяють два основних способи трансляції – *компіляція* програми або її *інтерпретація*.

При компіляції весь вихідний програмний код (той, який пише програміст) відразу переводиться в машинний. Створюється так званий окремий виконуваний файл, який ніяк не пов'язаний з вихідним кодом. Виконання виконуваного файлу забезпечується операційною системою.

При інтерпретації виконання коду відбувається послідовно (можна сказати, рядок за рядком). Операційна система взаємодіє з інтерпретатором, а не вихідним кодом.

Виконання скомпільованої програми відбувається швидше, тому що вона являє собою готовий машинний код. Однак на сучасних комп'ютерах зниження швидкості виконання при інтерпретації зазвичай не помітне.

Знайомство з Python і середовищами програмування

Історія



Мова програмування Python була створена приблизно в 1991 році голандцем Гвідо ван Россумом.

Своє ім'я - Python (Пайтон) - отримав від назви телесеріалу (“Monty Python”), а не плазуна.

Після того, як Россум розробив мову, він виклав її в Інтернет, де вже ціле співтовариство програмістів приєдналося до її поліпшення. Python активно вдосконалюється і в даний час. Часто виходять його нові версії. Офіційний сайт http://python.org.

З появою версії 3.0 розвиток Python як би пішов в двох напрямках: одночасно розвиваються і 2-га і 3-я версії Python.

Особливості мови програмування Python

Python - інтерпретована мова програмування: вихідний код частинами перетворюється в машинний в процесі виконання спеціальною програмою - інтерпретатором.

Python характеризується ясним синтаксисом. Читати код на цій мові програмування досить легко, тому що в ньому мало допоміжних елементів, а правила мови змушують програмістів робити *відступи*.



Зрозуміло, що добре оформлений текст з малою кількістю відволікаючих елементів читати і розуміти легше.

Python – це повноцінна, можна сказати універсальна, мова програмування. Він підтримує об'єктно-орієнтоване програмування (насправді вона і розроблявся як об'єктно-орієнтована мова).

Також Python поширюється вільно на підставі ліцензії GNU General Public License.

Дзен Пітона

Якщо інтерпретатору Пітона дати команду import this (імпортувати "сам об'єкт"), то виведеться так званий "Дзен Пітона", який ілюструє ідеологію і особливості даної мови. Глибоке розуміння цього дзену приходить тим, хто зможе освоїти мову Python в повній мірі і отримає досвід практичного програмування.

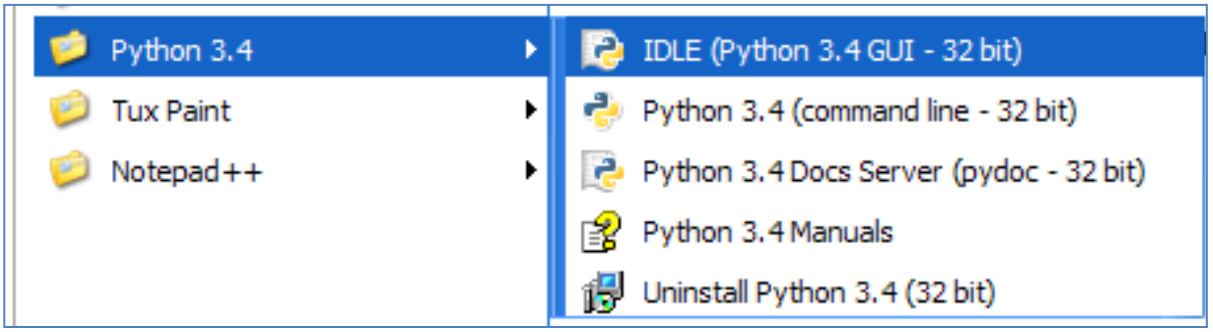
Фраза	**Переклад**
1. Beautiful is better than ugly.	Красиве краще за потворне.
2. Explicit is better than implicit.	Просте краще за складне.
4. Complex is better than complicated.	Складне краще за ускладнене.
5. Flat is better than nested.	Плоске краще ніж вкладене.
6. Sparse is better than dense.	Розріджене краще ніж щільне.
7. Readability counts.	Читабельність важлива.
8. Special cases aren't special enough to break the rules.	Виняткові випадки не настільки важливі, щоб порушувати правила.
9. Although practicality beats purity.	Однак практичність важливіша за чистоту.

Фраза	Переклад
10. Errors should never pass silently.	Помилки ніколи не повинні замовчуватися.
11. Unless explicitly silenced.	За винятком замовчування, яке задано спеціально.
12. In the face of ambiguity, refuse the temptation to guess.	У випадку неоднозначності не піддавайтеся спокусі вгадати.
13. There should be one - and preferably only one - obvious way to do it.	Повинен існувати один - і, бажано, тільки один - очевидний спосіб зробити це.
14. Although that way may not be obvious at first unless you're Dutch.	Хоча він може бути з першого погляду не очевидний, якщо ти не голландець.
15. Now is better than never.	Зараз краще, ніж ніколи.
16. Although never is often better than * Right * now.	Проте, ніколи частіше краще, ніж прямо зараз.
17. If the implementation is hard to explain, it's a bad idea.	Якщо реалізацію складно пояснити - це погана ідея.
18. If the implementation is easy to explain, it may be a good idea.	Якщо реалізацію легко пояснити — це може бути хороша ідея.
19. Namespaces are one honking great idea - Let's do more of those!	Простори назв - прекрасна ідея, давайте робити їх більше!

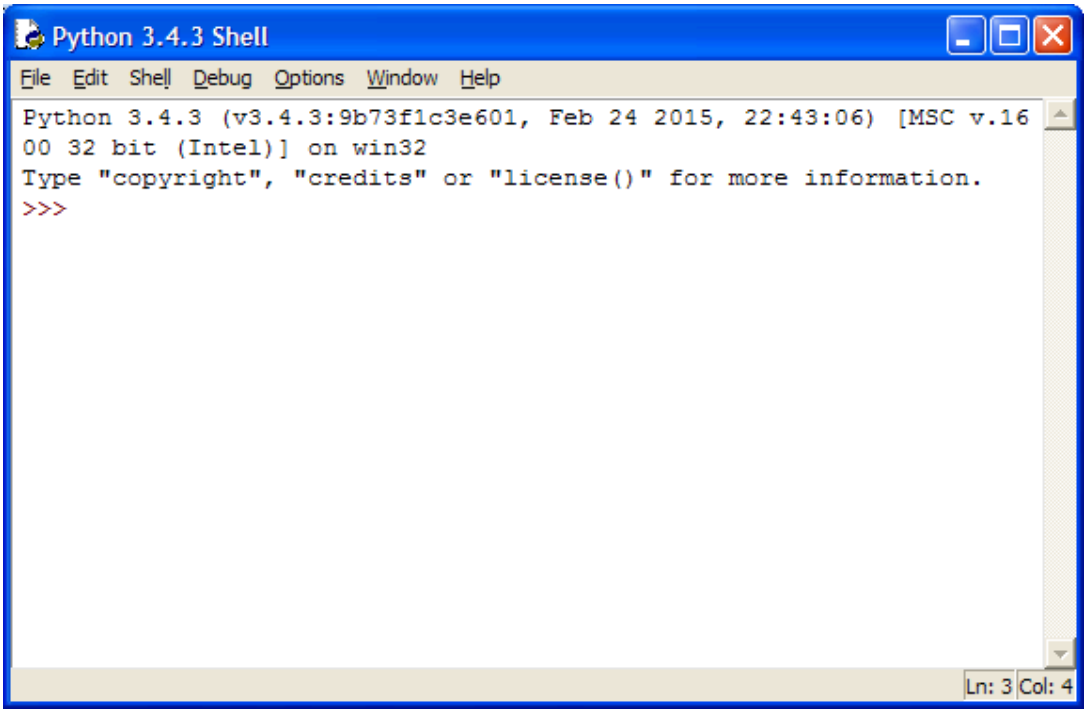
Крок 1. Налаштування редактора для написання програм

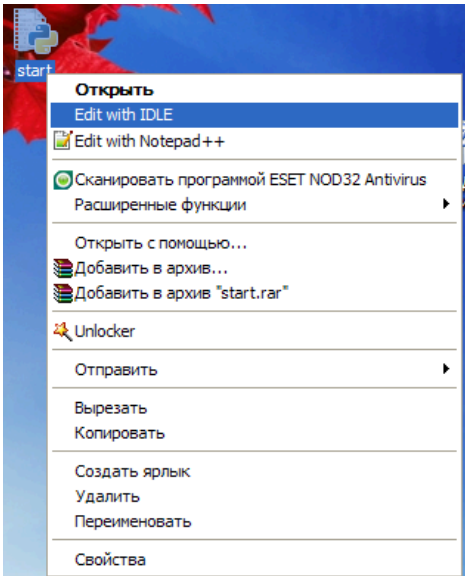
Встановлюємо Python на комп'ютер. Для цього скачуємо Python 3.4.3 (дана версія вважається найпопулярнішою і надійною) із сайта https://www.python.org/downloads/

Після встановлення мови програмування Python на комп'ютер із головного меню запускаємо програму IDLE (Python 3.4 GUI – 32 біт).

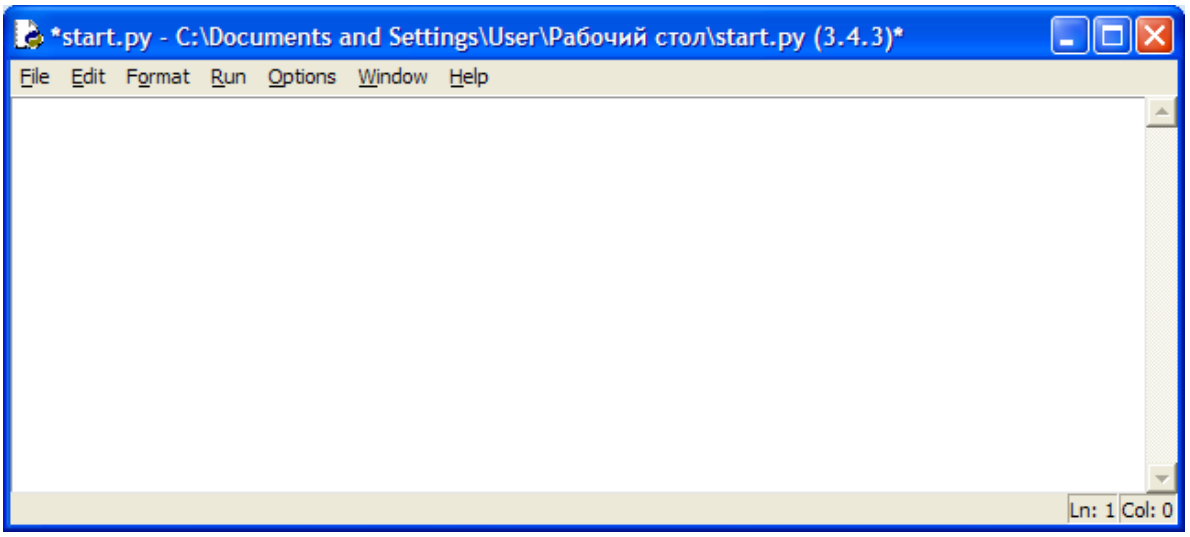


На робочому столі з'явиться вікно програми Python 3.4.3 Shell. Дане вікно є інтерактивною оболонкою використовується для виведення результатів, повідомлень про помилки, виводить підказки та ін..





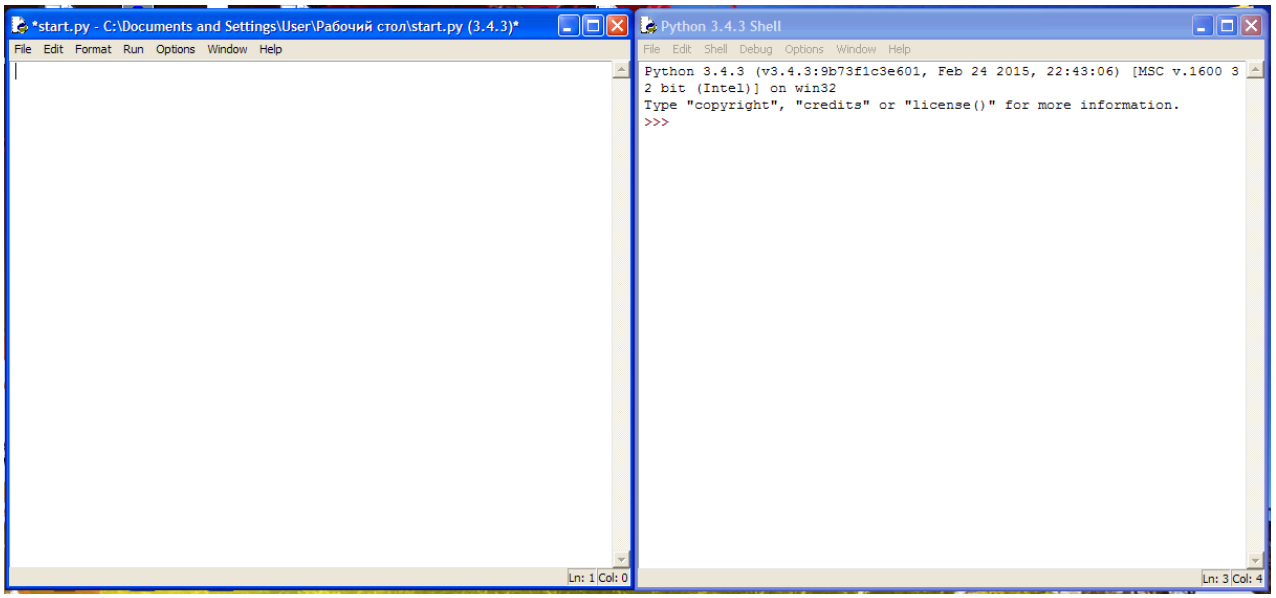
Для створення вікна редактора програм потрібно зберегти файл із назвою Start.py (File – Save – Робочий стіл - Start.py). З робочого столу відкриваємо файл Start.py, використовуючи контекстне меню, команду Edit with IDLE (в подальшому даний спосіб буде використовуватись для відкриття редактора програм створених мовою програмування Python). Наступник кроком потрібно стерти всі написи в даному редакторі і зберегти його з тим самим ім’ям.



Крок 2. Створення першої програми

На цьому кроці ми розпочнемо навчання по створенню програм за допомогою мови програмування Python. Один із перших кроків навчимось виводити результати роботи.

Розміщуємо на робочому столі 2 вікна



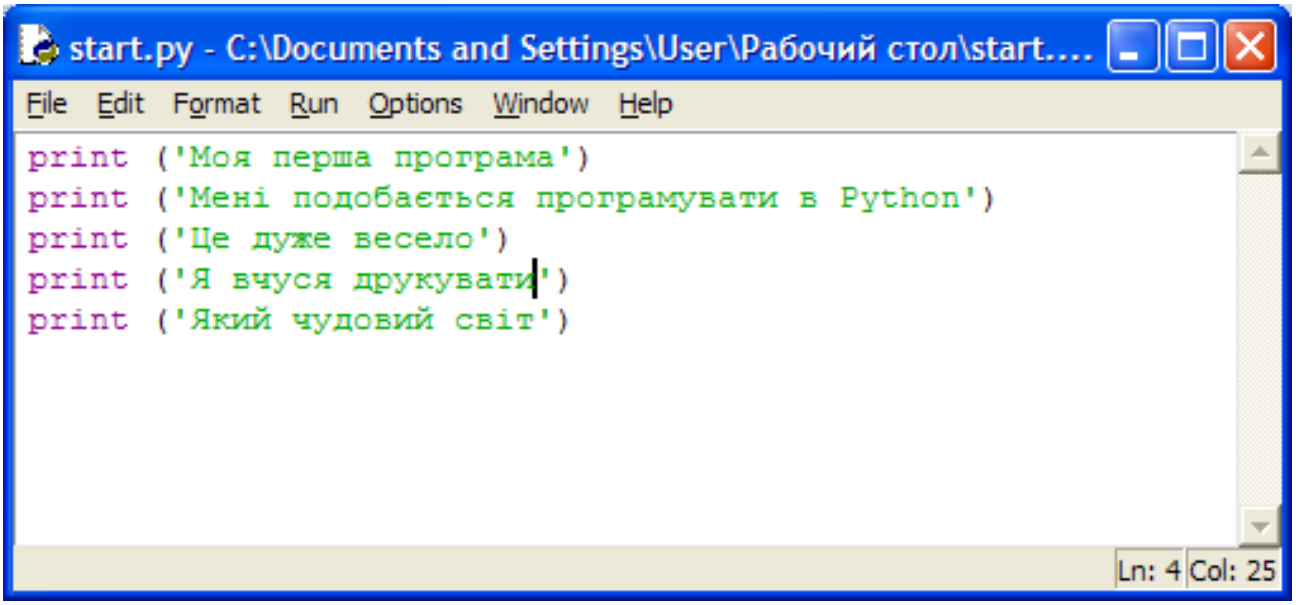
1 вікно Edit with IDLE використовується для написання тексту програми та запуску її на виконання.

2 вікно Python 3.4.3 Shell для перегляду результатів роботи програми.

Для виведення значень використовується функція **print.** В дужках вказується об’єкт виведення. Наприклад: **print ('Україна') -** надрукує текст «Україна»; **print (i)** - надрукує значення змінної «і»; **print('величина='+str(h))** – надрукує текст і число яке було в пам’яті комп’ютера.

Для запуску на перевірку та виконання програми потрібно виконати **Run – Run Module F5** або натиснути **F5**

Завдання1.



Завдання2.

```
print ('          Українською мовою')
print ('')
print ('    віщують,              речуть,')
print ('  розмовляють,         оповіщають,')
print (' теревенять,  патякають,  цідять,')
print ('галакають,  шварготять,  твердять,')
print (' балакають,  бовкають,  щебечуть,')
print ('  лепечуть, бесідують, говорять,')
print ('   гомонять, кажуть, свідчать,')
print ('     гутарять, теревенять,')
print ('          висловлюються,')
print ('             мовлять')
print ('               !!!')
```

Крок 3. Коментарі

Коментарі в коді програм дуже важливі. Вони використовуються, щоб на людській мові повідомити яку-небудь інформацію, для пояснення того чи іншого кроку, або для того щоб не втратити хід написання під час виконання великих проектів, а також для відключити частини коду вашої програми, якщо потрібно, щоб він тимчасово не працював.

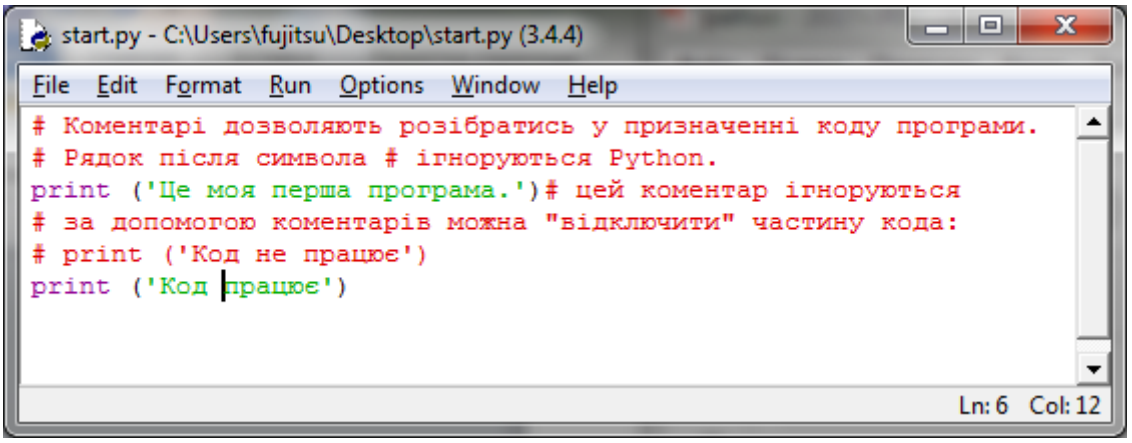
Для добавляння коментарів використовується символ **#.**

Наприклад:

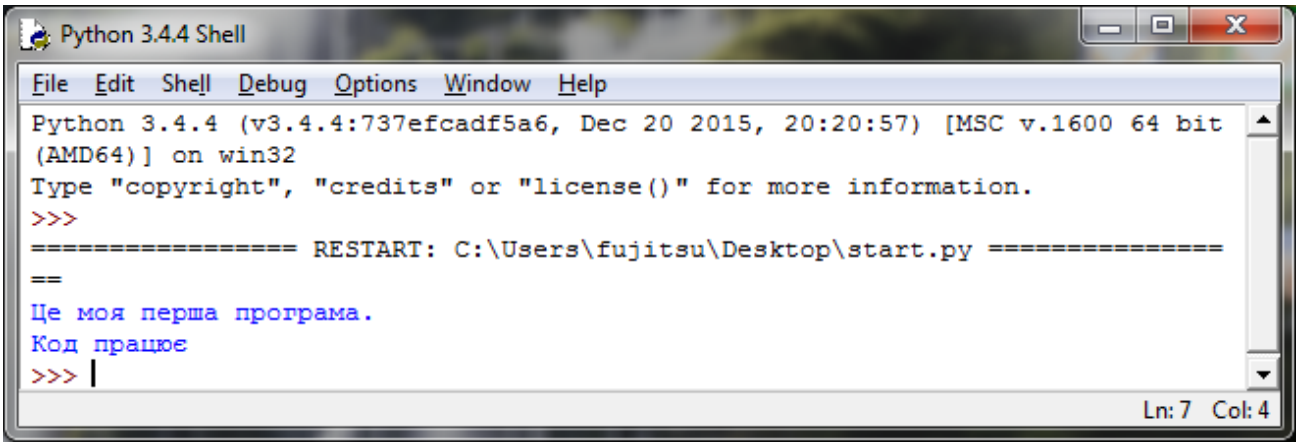
Коментарі дозволяють розібратись у призначенні коду програми.

Рядок після символа # ігноруються Python і в програмі позначається червоним кольором.

Приклад програмного коду:



Результат виконання програмного коду.





Крок 4. Числа і математика

Кожна мова програмування певним чином обробляє числа і математичні дії. Python містить кілька математичних символів. Назвемо їх:

+	плюс
–	мінус
/	слеш
*	зірочка
%	процент
<	менше
>	більше
<=	менше рівне
>=	більше рівне

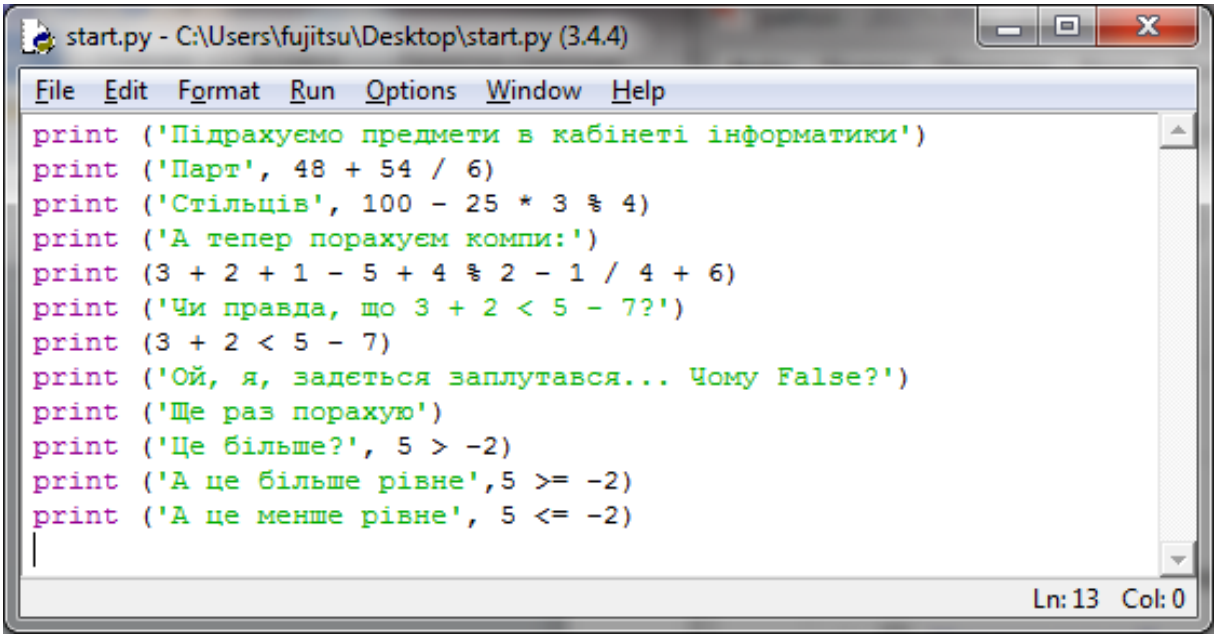
їх використовують для запису математичних операцій, а також для запису різних пояснень.

Хочу звернути вашу увагу, що розділювач цілої частини і дробової в Python використовується «.» (крапка), а не «,» (кома) яку ми використовуємо в математиці.

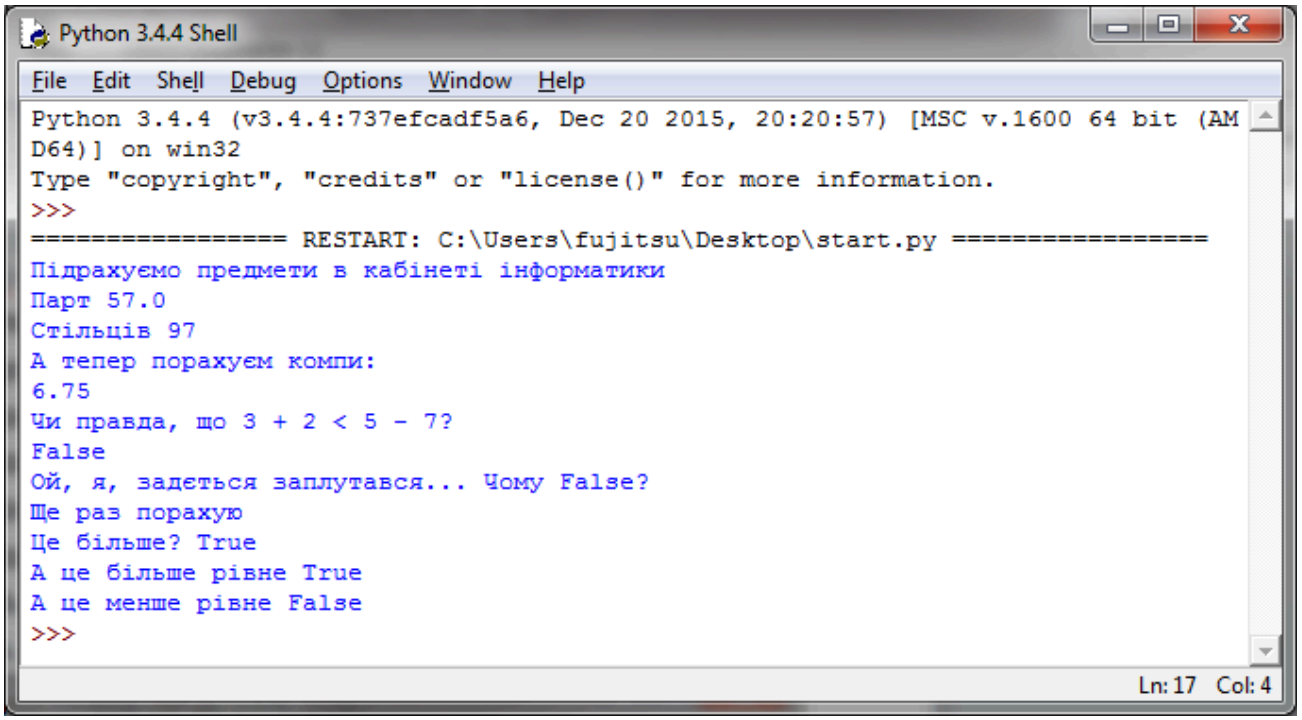
Знак % використовується не для знаходження процента від числа, а вираховує остачу від ділення націло (5%2=1, 43%5=3)

Порядок дій: дужки, експоненти (піднесення до степеня), множення, ділення, додавання, віднімання.

Наприклад:



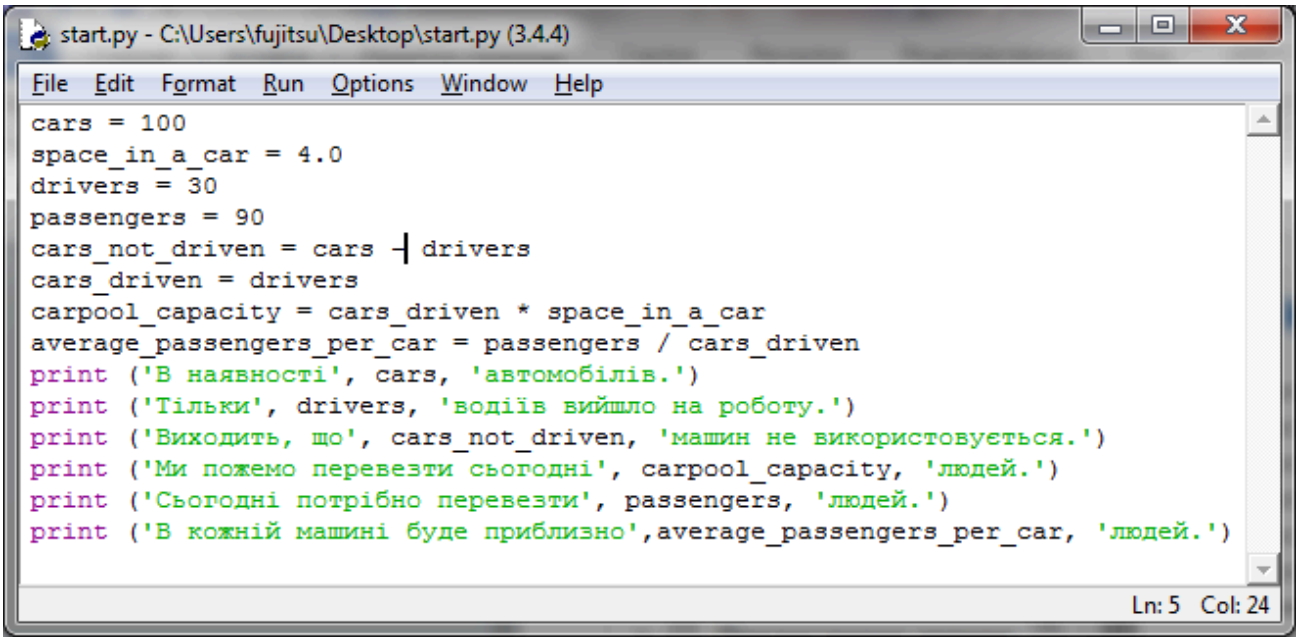
Результат виконання цього програмного коду:



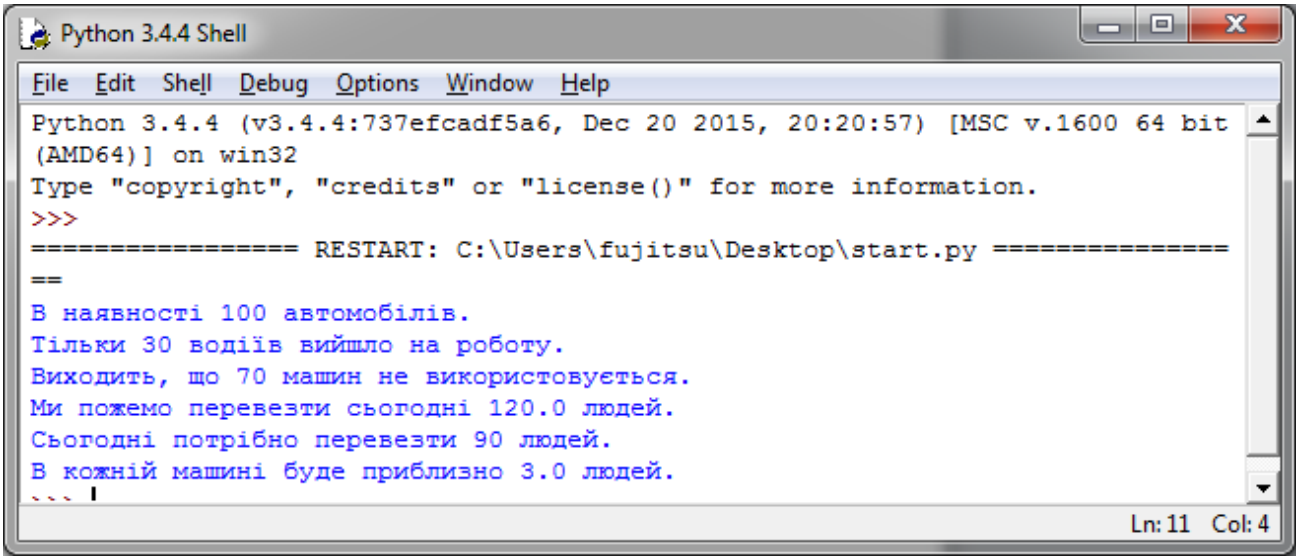
Крок 5. Змінні і імена

Ви вже можете виконувати обчислення та виводити текст за допомогою команди print. Тепер розберемося зі змінними. У програмуванні змінна є всього лише ім'ям для чого-небудь. Програмісти використовують змінні, щоб код був схожий на людську мову і його простіше було читати. Якби зручних імен змінних не існувало, програмісти б заблукали в нетрях власного коду.

Для надання імен змінним використовують англійський алфавіт, цифри та символ підкреслення яким заміняють пропуск між словами (недопустимо в назві змінної використовувати цифру на першому місці і символ пропус між символами), рекомендується при створенні змінних використовувати англійські слова для зручності читання коду програми, не рекомендується використовувати букви українського алфавіту. Наприклад:



Результат виконання цього програмного коду:

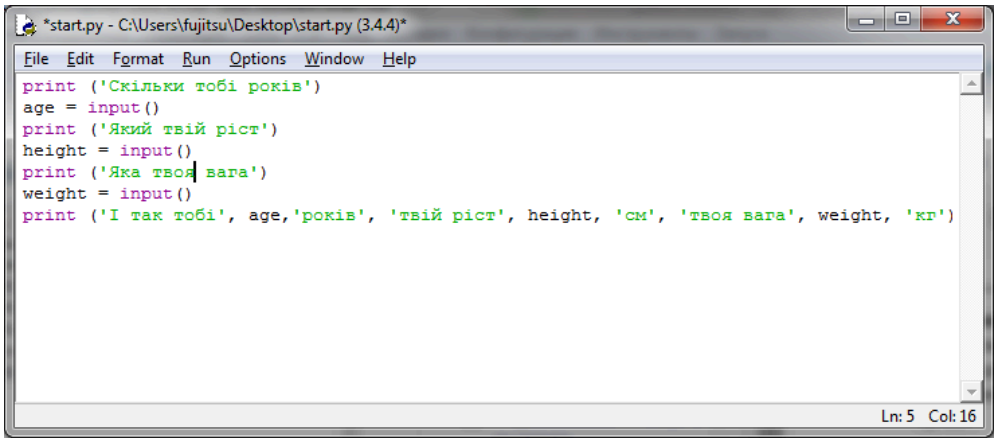


В цьому програмному коді змінні: cars, space_in_a_car, drivers, passengers змінні сталого типу, значеннь набувають напочатку програми. Змінні: cars_not_driven, cars_driven, carpool_capacity, average_passengers_per_car – змінного типу, їх значення залежиться від математичних операції (програма зама везначення тип цих величин). При заданні значення змінній використовується символ «=», що в даній мові програмування використовується як символ присвоїти (присвоїти змінній вказаній зліва значення змінної вказаної справа). Коли порівнюється дві змінні використовується сімвол «==» (подвійний символ дорівнює). Використовуючи символа «=», для полегшення читання коду, перед ним і після ставляться пропуски.

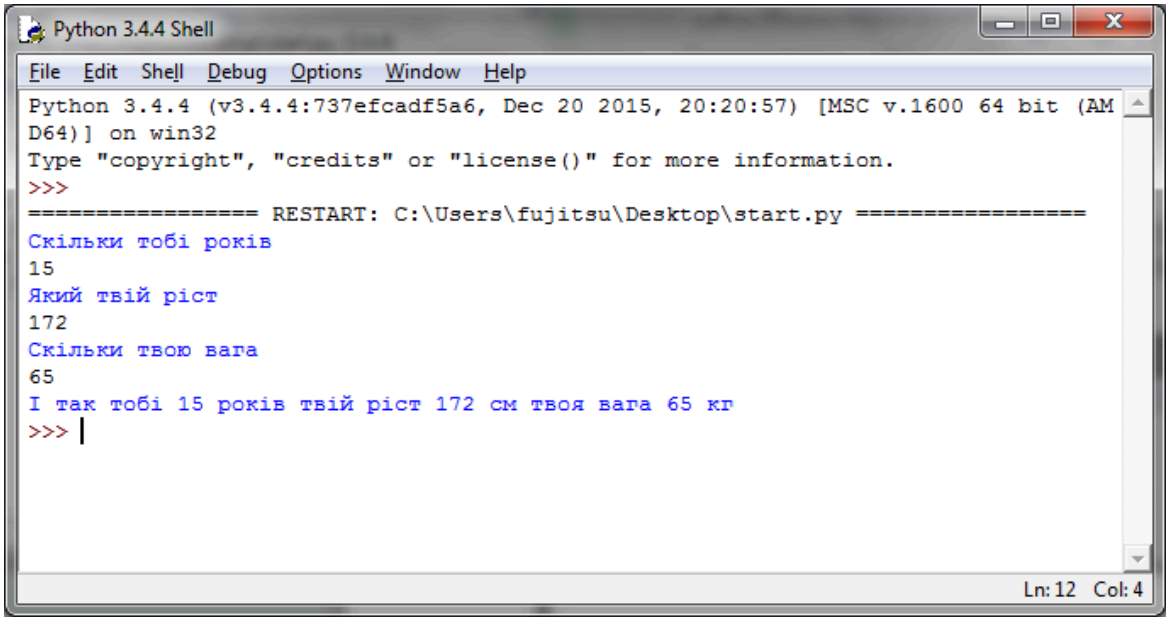
Крок 6. Введення, виведення

Виведення ми вже багато раз розглядали. Функція **print** вже знайома, настав час перейти до програм де використовується введення даних з клавіатури по запиту. Функція **input** зчитує рядок даних, отриманих з пристрою введення (клавіатури). Без перетворення це зчитаний рядок і збережений як рядок, перетворення рядка в число розглянем пізніше.

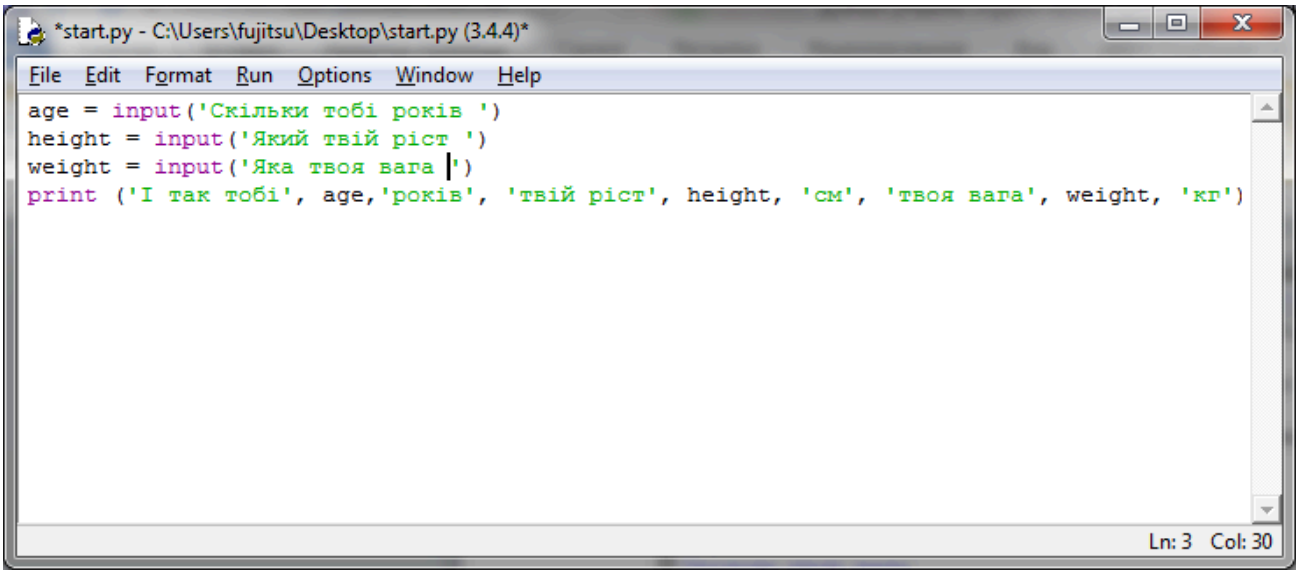
Приклад введення і виведення:



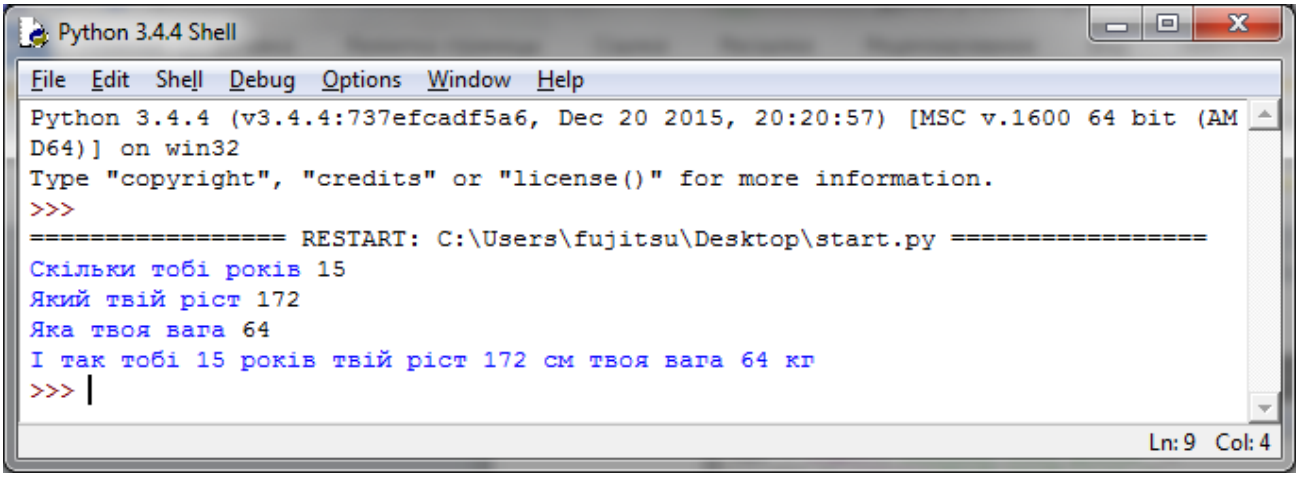
Результат виконання цього програмного коду:



2 спосіб реалізації функції **input** для введення даних на прикладі тієї ж задачі:



Результат виконання цього програмного коду:





Крок 7. Прості типи даних

Мова програмування Python використовує такі прості типи даних:

1. Цілі числа
2. Дійсні числа
3. Логічний тип даних
4. Рядки.

1. **Int** – цілі числа: 1, 2, 0, -10, 9999 і т.д. Відображаються просто як числа. Для перетворення будь-якого значення на ціле число використовується функція **int().**

Для введення цілого числа потрібно записати програмний код:

r=input('введіть r=') #вводить і запам’ятовує число як рядок

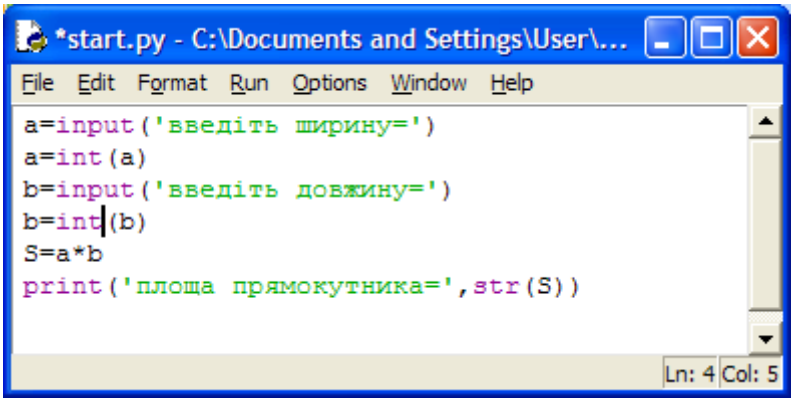
r=int(r) #перетворює рядок в ціле число

або **r=int(input('введіть r='))** #вводить рядок, перетворює рядок в ціле число і запам’ятовує його.

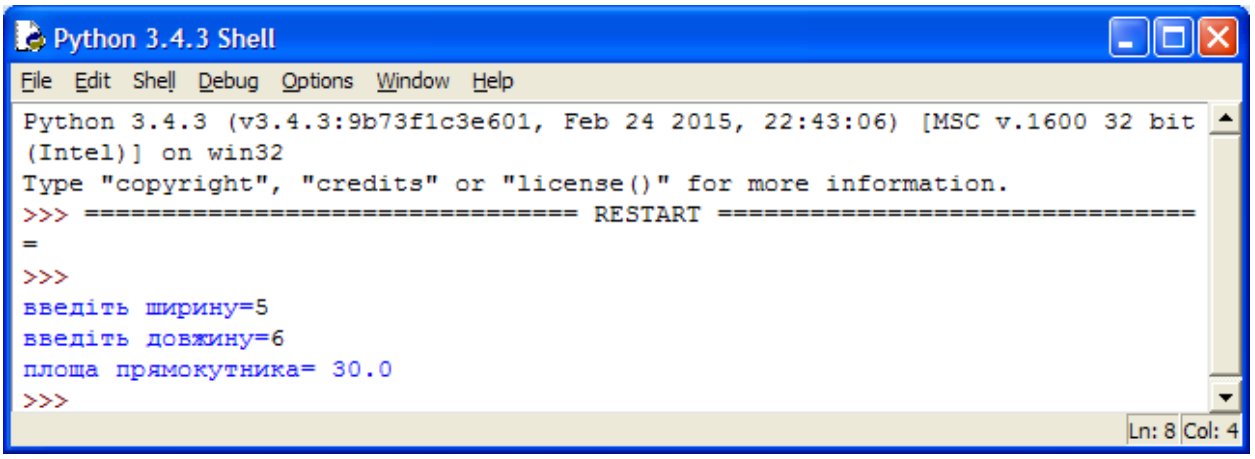
Для виведення цілого числа, як результату обрахунку на екран, потрібно число перетворити в рядок, за допомогою функції **str().** Наприклад:

print('Результат обрахунку S=', str(S)).

Задача 1. Обрахувати прощу прямокутника, коли відомо його сторони задані простими числами.



Результат виконання цього програмного коду:



2. **Float** – дійсні числа: 1.0, 2.543, 0.0, -0.99999, 123123123.4213 і т.д. Виглядають як 2 числа (ціла і дробова частина), розділені крапкою. Для перетворення будь-якого значення на дійсне число використовується функція **float().**

Для введення дійсного числа потрібно записати програмний код:

r=input('введіть r=') #вводить і запам’ятовує число як рядок

r=float(r) #перетворює рядок в дійсне число

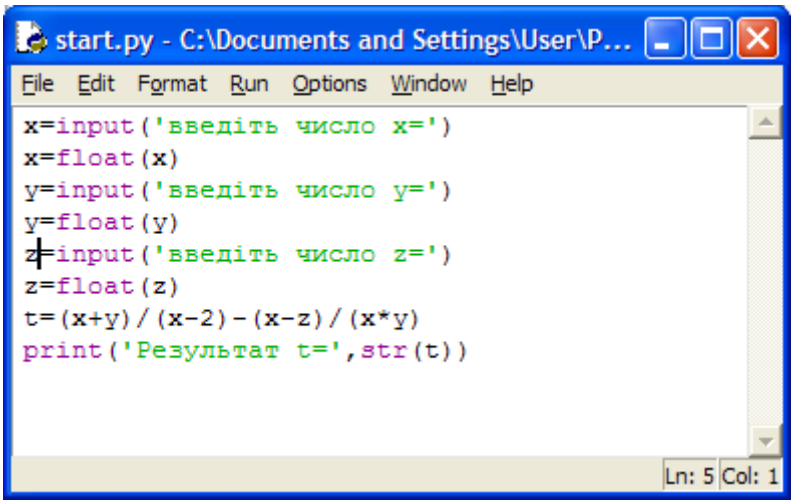
або **r=float(input('введіть r='))** #вводить рядок, перетворює рядок в дійсне число і запам’ятовує його.

Для виведення дійсного числа, як результату обрахунку на екран, потрібно число перетворити в рядок, за допомогою функції **str().** Наприклад:

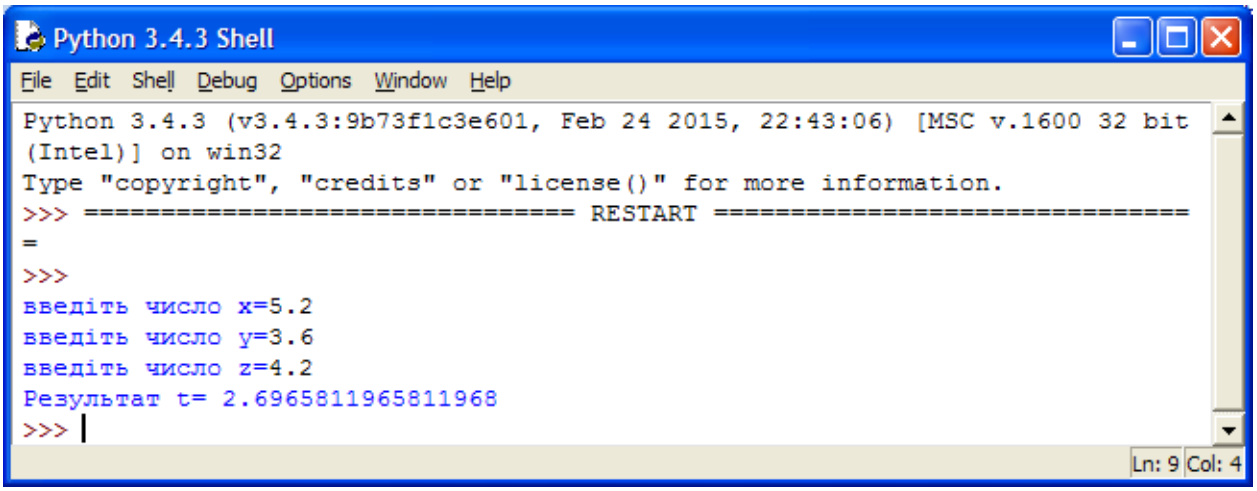
print('Результат обрахунку S=', str(S)).

Задача 2. Задані значення змінних *x, y, z.* Обчисліть значення змінної *t.*

$$t = \frac{x+y}{x-2} - \frac{x-z}{xy}$$



Результат виконання цього програмного коду:



Задачі на самостійне виконання

1. Задані два дійсних числа *a* та *b*. Одержати їхню суму, різницю та добуток.
2. Задано змінні A і B. Обміняти їх значення між собою.
3. Обміняти значення трьох змінних A,B,C так, щоб змінна A набула значення змінної B, змінна B набула значення змінної C, а змінна C набула значення змінної A.
4. Задані три дійних числа. Знайти їх середнє арифметичне.
5. Учні на свято купили цукерки та печиво. Відомо ціни цукерок і печива та маси куплених солодощів кожного виду. Знайдіть повну вартість покупки.
6. Задано радіус кола. Обрахувати довжину кола та площу круга.
7. Два потяги вирушили одночасно назустріч один одному. Визначити, через який час вони зустрінуться, якщо відома відстань між ними в момент початку руху та швидкості кожного потяга.



Крок 7. Вбудовані функції та підключення бібліотек

Часто під час створення програм на обрахунок не вистачає стандартних математичних операцій, тоді на порятунок приходять вбудовані функції або бібліотеки спеціальних функції.

Вбудовані функції використовують як частини програми без підключення бібліотек, якщо потрібно інші функції такі як: sin, cos, корінь і т.д. потрібно підключити бібліотеку.

Вбудовані функції

% – остача від ділення націло (5%2=1)

// – ціла частина від ділення націло (5//2=2)

abs(x) – модуль від числа;

pow(x,y) або x**y – піднесення числа x до числа y;

divmod(x,y) – відтворює цілу частину і остачу від ділення x на y;

bin(x) – переведення числа у двійкову систему числення;

hex(x) – переведення числа у шістнадцяткову систему числення;

max(x,y) – пошук максимуму з 2 чисел, також може приймати будь-яку кількість аргументів;

min(x,y) – пошук мінімуму з 2 чисел, також може приймати будь-яку кількість аргументів;

round(x) – округлення числа;

round(x,y) – округлення числа x із вказаною точністю – y знаків після коми.

Наприклад: abs(-5)=5, pow(2,3)=8, divmod(5,2)=(2,1) і т.д.



Бібліотеки. Підключення бібліотек.

Бібліотека математичних функцій – Math.

Для підключення бібліотек використовують команду **import** після якої вказується назва бібліотеки.

Наприклад: **import math** #підключення бібліотеки математичних функцій.

Після імпортування модуля його назва стає змінною, через яку можна отримати доступ до атрибутів модуля. Наприклад: **math.pi** # число π

Дана бібліотека містить:

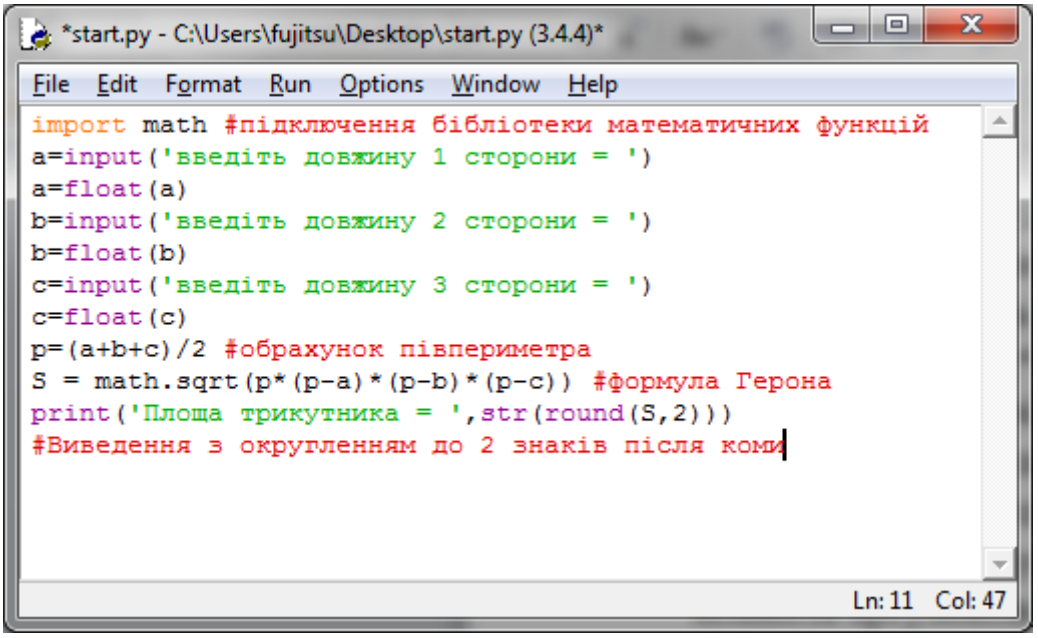
- 2 константи math.pi – число π та math.e – експонента;
- функції округлення чисел: math.ceil(x) (округлення "вверх"), math.floor(x) (округлення "вниз");
- степеневі та логарифмічні функції: math.pow(x, y), math.exp(x), math.log(x), math.log(x, y), math.log10(x), math.sqrt(x);
- тригонометричні функції: math.sin(x), math.cos(x), math.tan(x), math.asin(x), math.acos(x) та інші (включаючи гіперболічні);
- функції для переведення градусів та радіан: math.degrees(x) та math.radians(x);
- та деякі інші.

Підключити певні атрибути модуля можна за допомогою інструкції **from <назва модуля> import*.** Формат інструкції from дозволяє підключити всі (точніше, майже всі) змінні з модуля. Наприклад: **from math import*.**

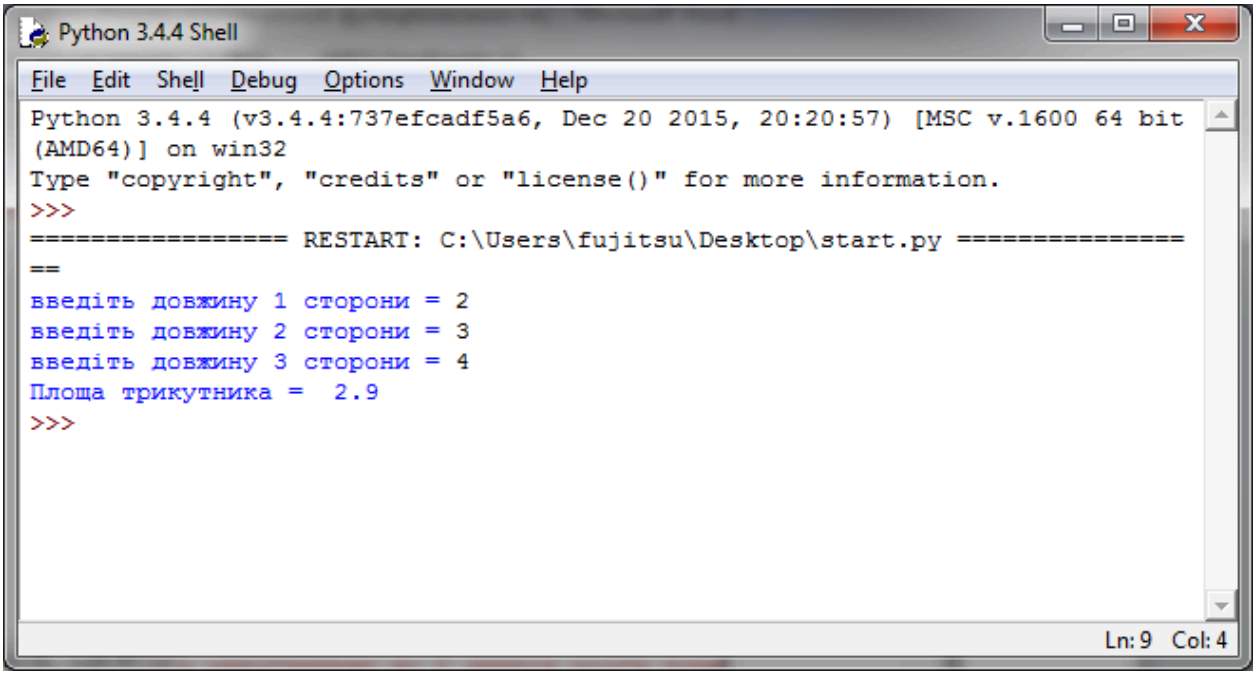
Після підключення бібліотеки таким способом перед назвою функції не потрібно вказувати назву модуля як змінної (pi, а не math.pi; sqrt, а не math.sqrt і т.д.).

Задача 1. Обчислити площу трикутника за формулою Герона.

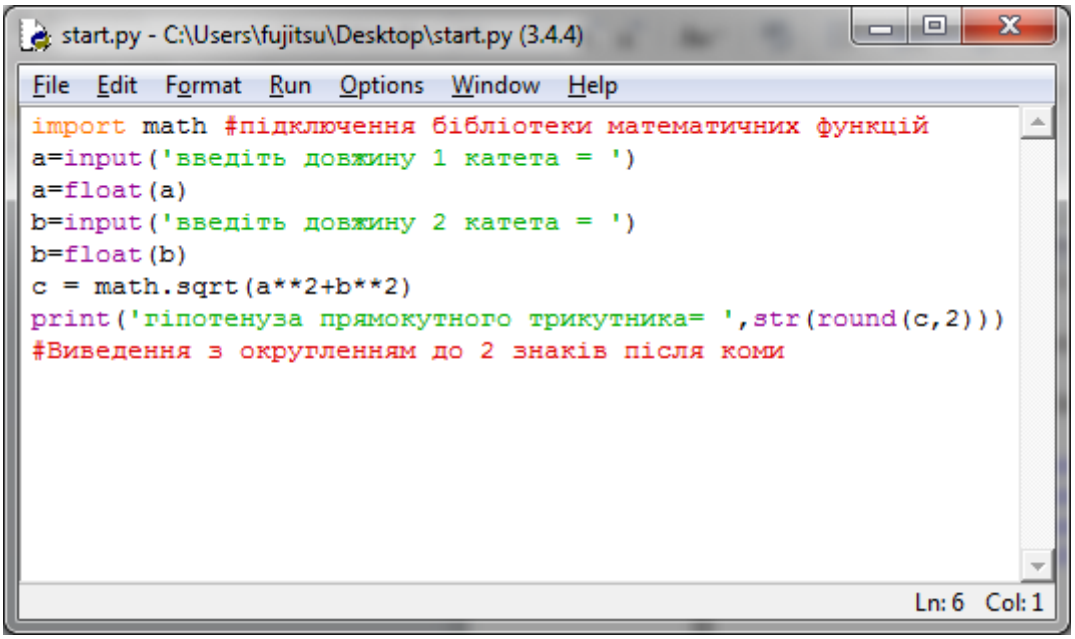
$$S = \sqrt{p(p - a)(p - b)(p - c)}$$



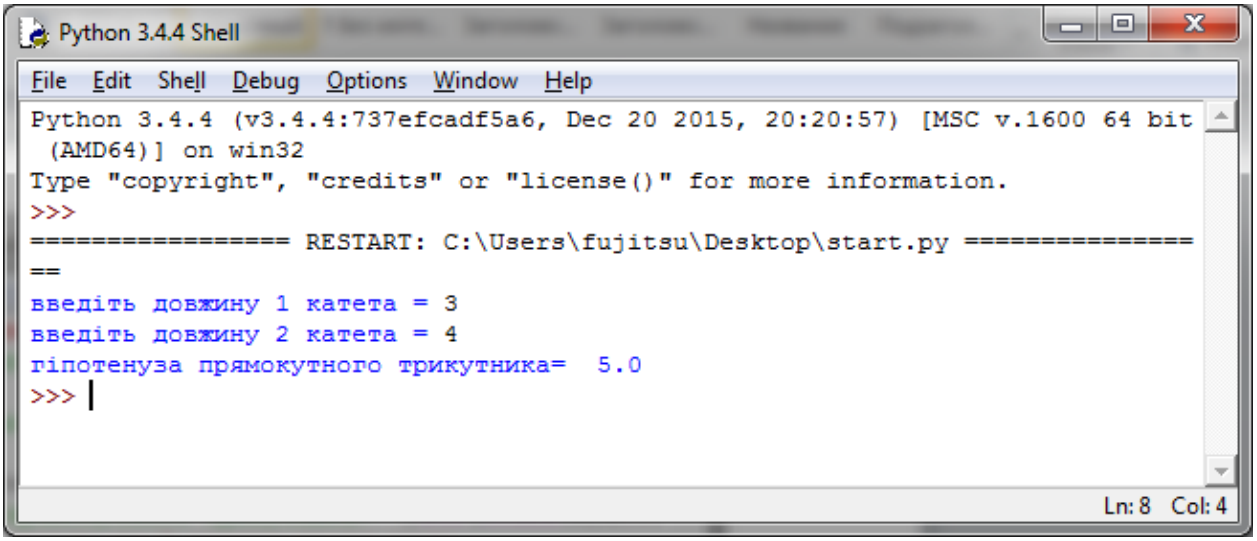
Результат виконання цього програмного коду:



Задача 2. Обрахувати гіпотенузу прямокутного трикутника за теоремою Піфагора коли відомо два катети. $c = \sqrt{a^2 + b^2}$



Результат виконання цього програмного коду:



Задачі на самостійне виконання.

1. Задані гіпотенуза і один з катетів прямокутного трикутника. Знайти другий його катет і площу трикутника.
2. Обчислити відстань між двома точками з координатами x_1,y_1 *та* x_2,y_2.
3. Трикутник заданий координатами своїх вершин. Знайти периметр та площу трикутника.
4. "Рівновеликий квадрат". Знайти сторону квадрата, що має ту ж саму площу, що й круг із заданою довжиною кола L.
5. "Секунди → Час". Починаючи з деякого моменту часу пройшло K повних секунд. Виразити пройдений час у повних годинах, хвилинах та секундах.
6. Учень школи штату Колорадо на протязі навчального року мав канікули загальною тривалістю A місяців B тижнів C днів. З них він провів на узбережжі Мексіканської затоки X місяців Y тижнів Z днів. Четвертину того часу, що залишилась, учень допомагав батькам у полі, а весь інший час готувався до переекзаменовки з математики. Виразити всі проміжки часу у повних місяцях (30 днів), тижнях та днях.

Крок 8. Логічний тип

Логічний тип даних є дуже простим і включає всього 2 можливих значення: **True** та **False** – істина та хиба. Ці значення виникають, насамперед, як результат перевірки (обчислення) будь-яких умов у програмі.

Наприклад, результатом перевірки 10 > 0 буде True, 91 < 0 – False. Коли умова містить змінні, її значення, відповідно, залежить від значень цих змінних на момент перевірки умови (в циклі або розгалуженні).

Таблиці істинності

x	not *x*
true	false
false	true

Таблиця істинності для заперечення

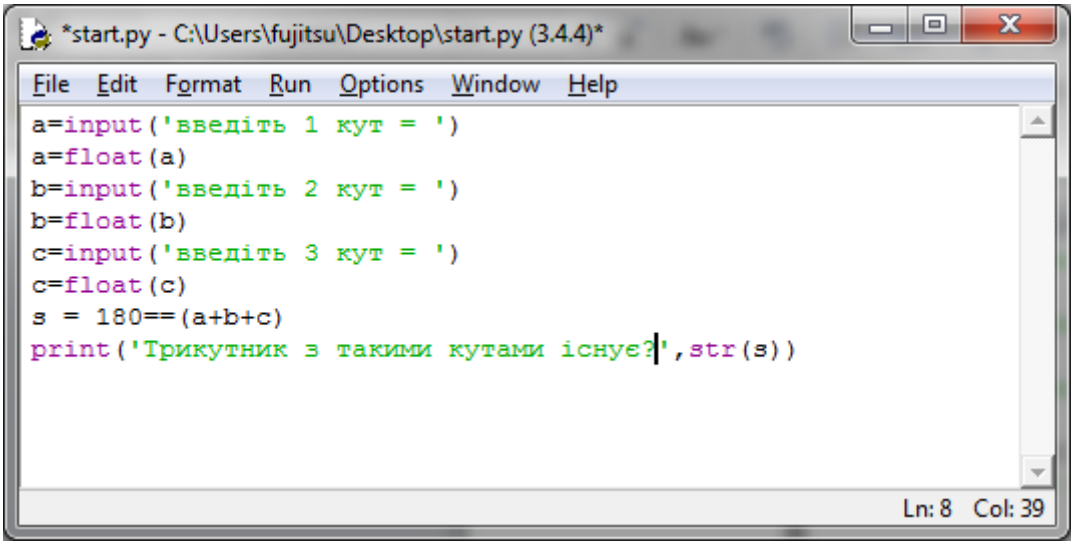
x	*y*	*x and y*
true	true	true
true	false	false
false	true	false
false	false	false

Таблиця істинності для кон'юнкції

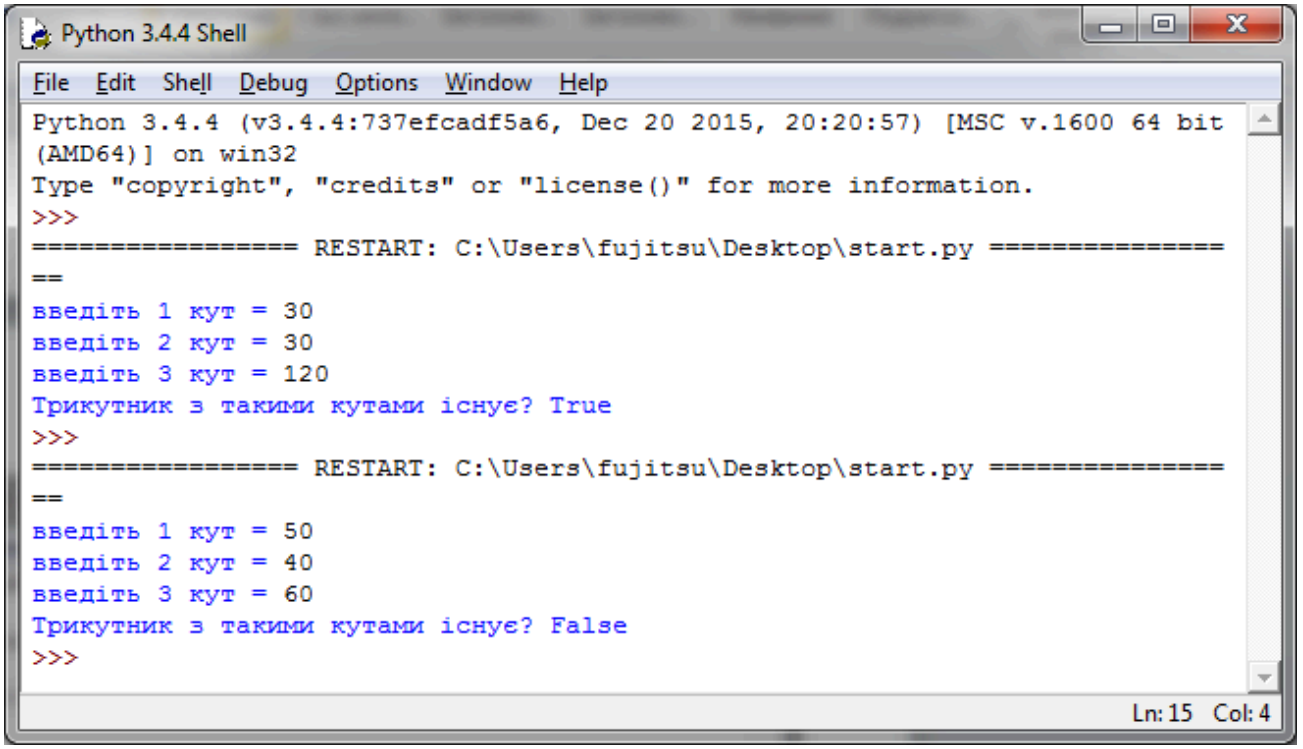
x	*y*	*x or y*
true	true	true
true	false	true
false	true	true
false	false	false

Таблиця істинності для диз'юнкції

Задача 1. Задано градусні міри трьох кутів. Визначити чи існуватиме трикутник із такими кутами.

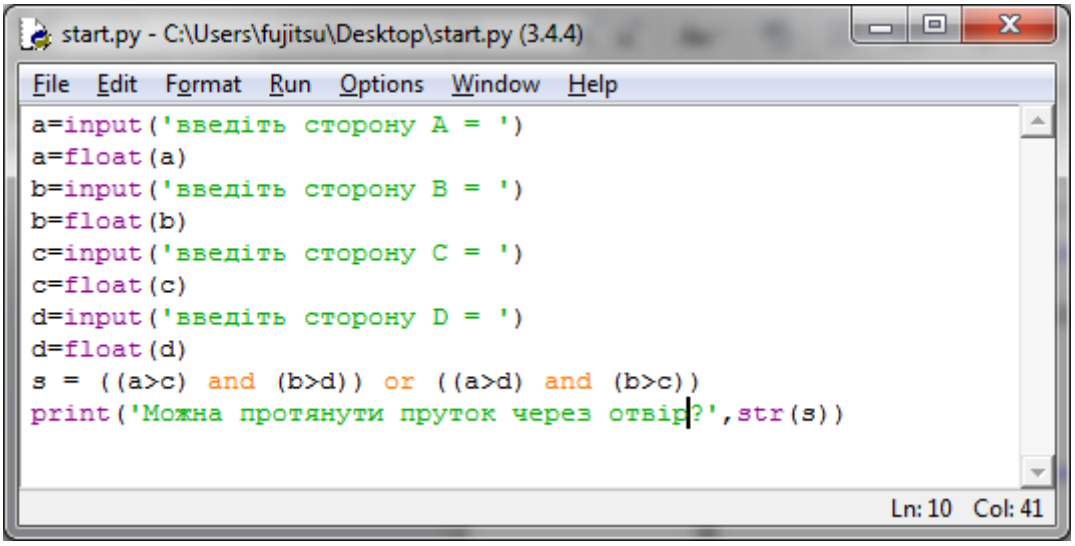


Результат виконання цього програмного коду:

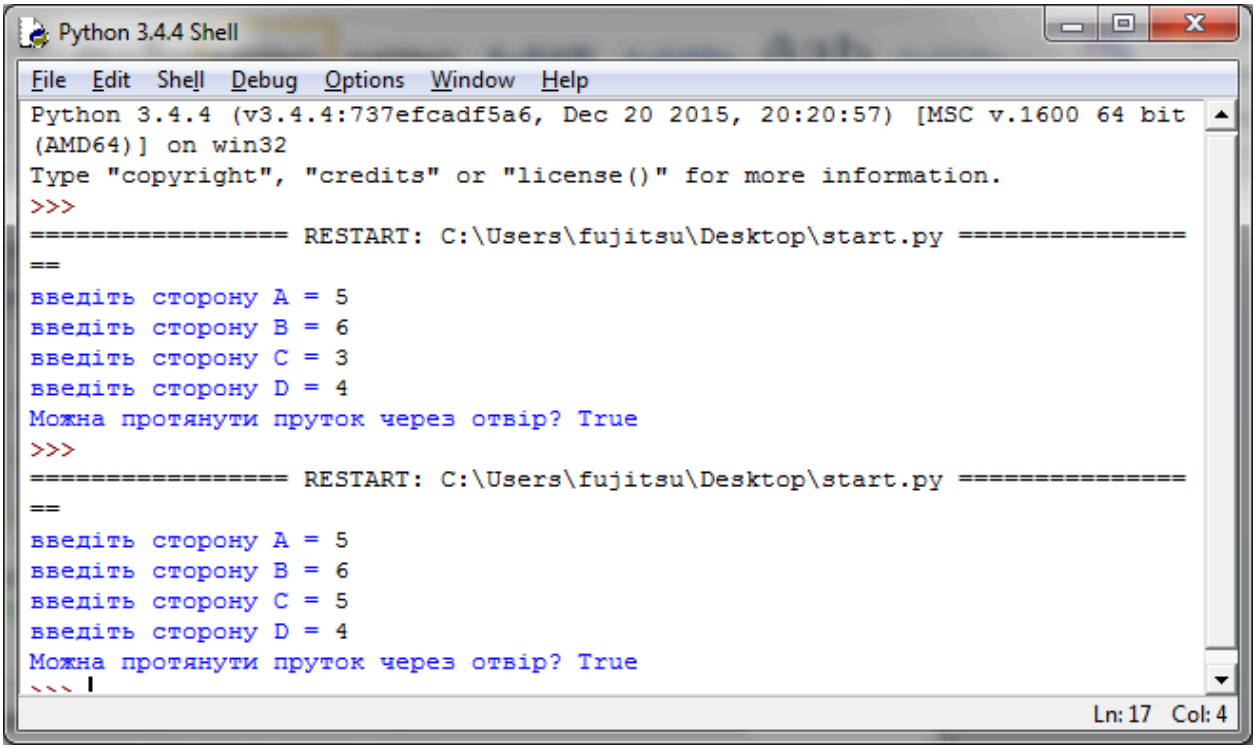


Задача 2. "Отвір". Є прямокутний отвір зі сторонами A і B. Визначити, чи можна протягнути через цей отвір дуже довгий пруток

прямокутного перерізу зі сторонами C і D при умові, що сторони перерізу паралельні сторонам отвору.



Результат виконання цього програмного коду:



Задачі на самостійне виконання

1. Задано вагу кожного з трьох камінців. Відомо, що два з них мають рівну вагу. Визначити, чи вірно, що третій камінець легший кожного з інших.
2. Квадратики для гри в "хрестики-нулики" пронумеровано зліва направо і згори донизу. Задано номери двох будь-яких квадратиків із дев'яти. Визначити, чи знаходяться ці квадратики на одній діагоналі.
3. Визначити, чи мають задані проміжки [a, b] і [c, d] спільні точки.
4. Два трикутники задано своїми сторонами. Визначити, чи дійсно перший трикутник більший за другий.



Крок 9. Умовне розгалуження

Алгоритм, що містить хоча б одну умову, в результаті перевірки якої здійснюється перехід до одного з можливих кроків, називається розгалуженим.

У більшості програм на певному кроці їх виконання потрібно вибирати той чи інший варіант подальших дій, тобто використовувати розгалуження. Умовою вибору того чи іншого варіанту є виконання деякої умови. Сама умова повинна бути логічного типу, тобто приймати лише два значення **true** або **false**. Сама умова записується одразу після оператора **if**, а після її через двокрапку записується команда, які необхідно виконати. Якщо команд декілька, то вони відокремлюються блоком, які мають однаковий відступ після оператора if. Відступ встановлюється клавішею Tab або пробілами. Якщо розгалуження закінчилося, то наступні команди слід писати під оператором if

Наприклад,

```
if a>0:
    print('Дане число додатне')
```

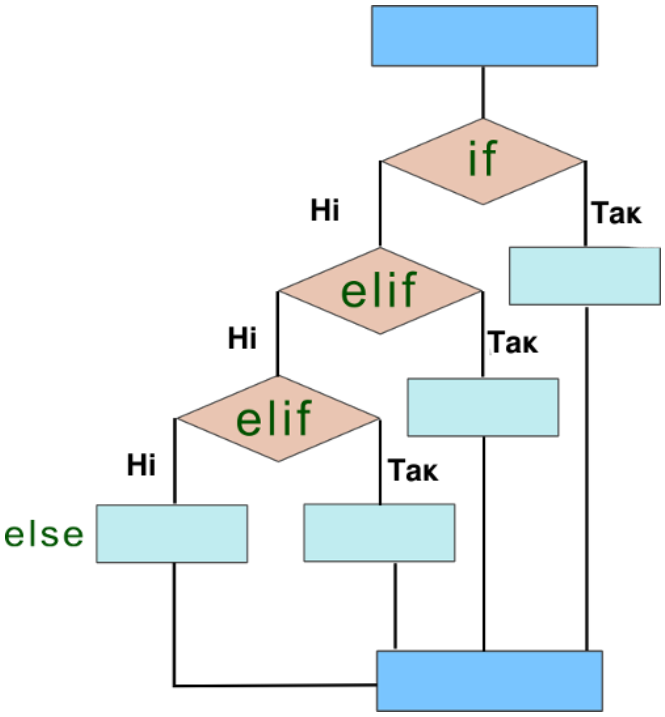
У випадку виконання інших команда у разі невиконання умови слід їх писати після оператора **else**, не забуваючи також поставити двокрапку.

Наприклад,

```
if a>0:
    print('Дане число додатне')
else:
```

print('Дане число недодатне')

Якщо необхідно використати декілька гілок розгалуження, то використовують оператор **elif**, що є скороченою формою операторів **else: if**, причому їх кількість необмежена.



Оператори розгалуження – коротке резюме

Вираз	Опис
if <умова>: <блок>	Найпростіша умовна конструкція. Якщо <умова> істинна, то <блок> виконується, в іншому випадку – пропускається.

if <умова>:
  <блок 1>
else:
  <блок 2>
	Умовна конструкція. Якщо <умова> істинна, то виконується <блок 1>, якщо хибна – <блок 2>.

if <умова 1>:
  <блок 1>
elif <умова 2
...
elif <умова N>:
  <блок N>
else:
  <блок N+1>
	Умовна конструкція з додатковою умовою elif і необов'язковим else в кінці. Буде виконаний блок після першої ж умови, яка виявиться істинною. Якщо жодна умова не приймає значення True, буде виконаний блок після заключного else.



Наприклад,

```
if a>0:
    print('Дане число додатне')
elif a<0:
    print('Дане число від'ємне')
else:
    print('Дане число дорівнює нулю')
```

Зауваження: функцію print() можна писати в тому самому рядку, де і оператори if, elif чи else. Це можна робити, коли команда, що відноситься до умови лише одна, а у випадку блоку команд краще записувати з окремого рядка.

Розглянемо, які умови можна використовувати

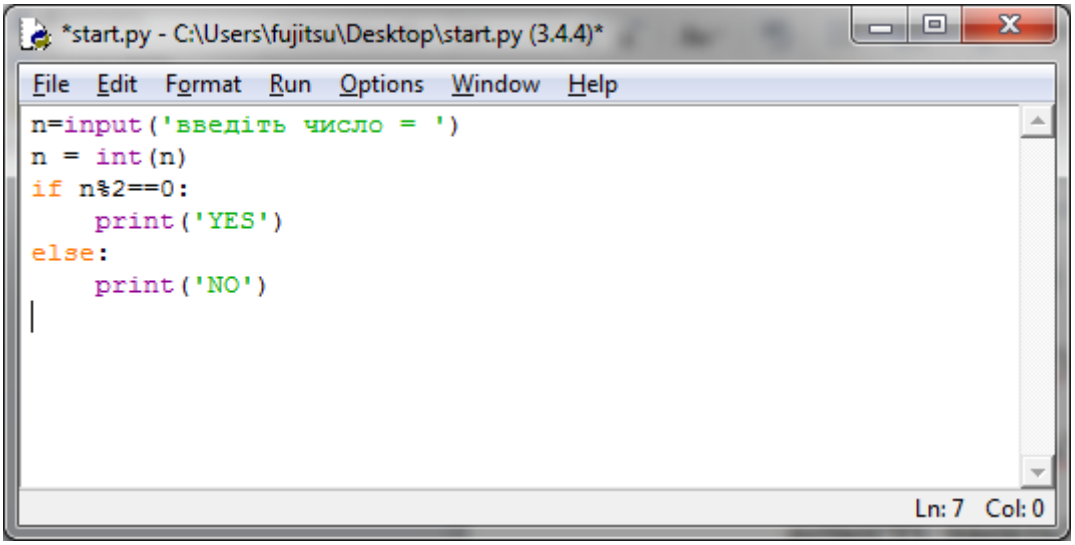
Назва	**Позначення**	**Приклад**
Рівність	==	a==25
Більше	>	a>25
Менше	<	a<25
Більше або дорівнює	>=	a>=25
Менше або дорівнює	<=	a<=25
Не дорівнює	!=	a!=25

Якщо умова складна, то прості умови пов'язуються між собою службовими словами **and**, **or** та **not()**(логічне "і", логічне "або" та логічне "ні"). На мові PYTHON у цих операцій найнижчий пріоритет, тому прості умови не беруться в дужки.

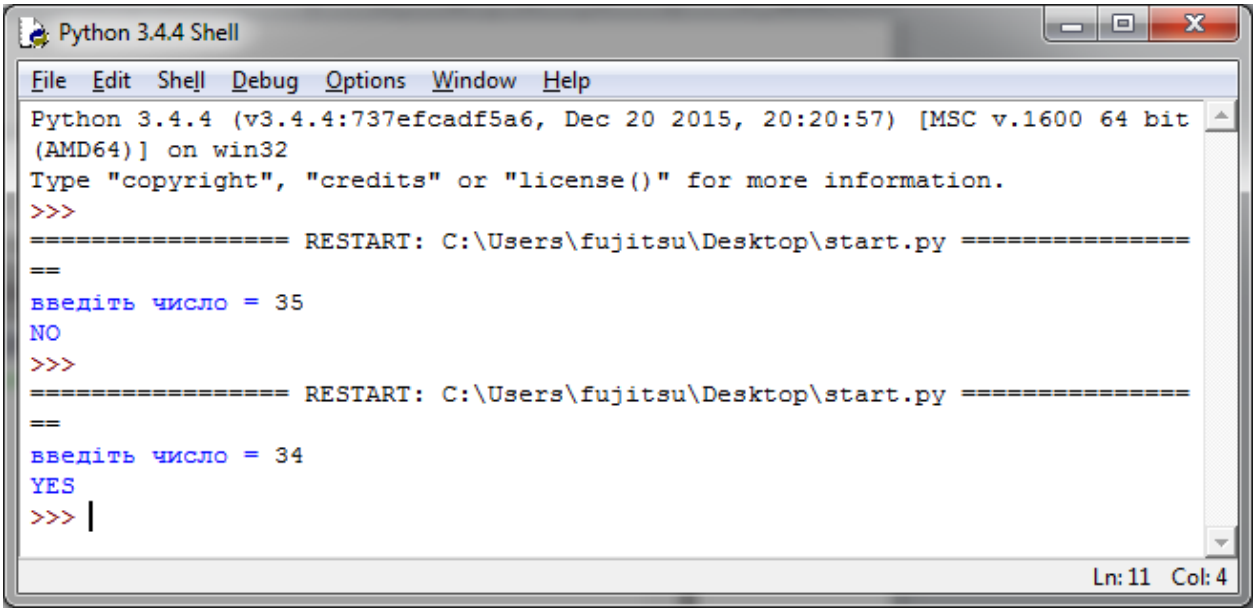
Наприклад: в якій координатній чверті знаходиться точка з координатами (x,y):

```
if x>0 and y>0:
    print('I чверть')
elif x<0 and y>0:
    print('II чверть')
elif x<0 and y<0:
    print('III чверть')
elif x>0 and y<0:
    print('IV чверть')
else:
    print('Точка лежить на координатних осях')
```

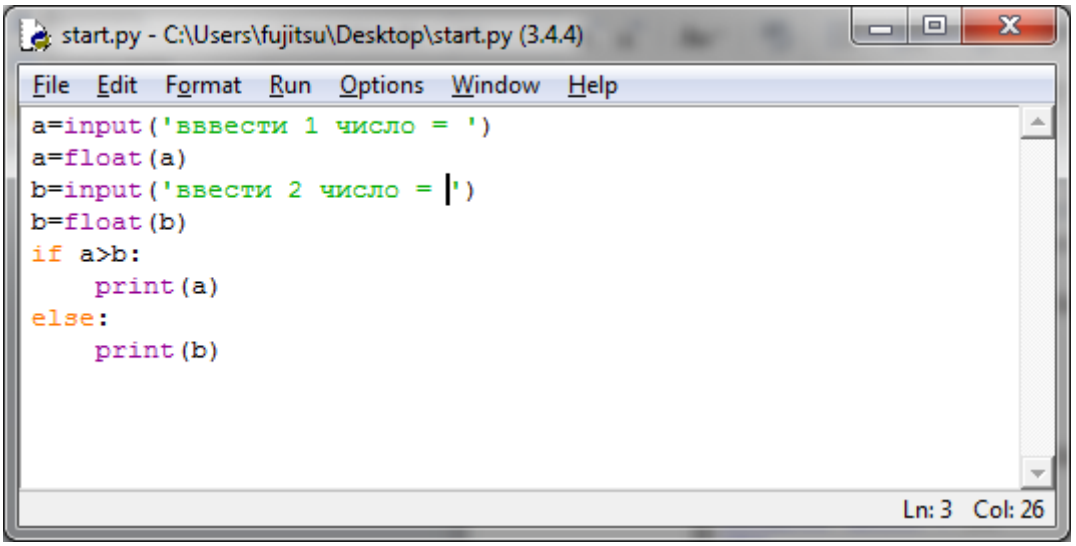
Задача 1. Дано одне натуральне число. Якщо воно парне, то вивести слово «YES», а якщо непарне, то – «NO».



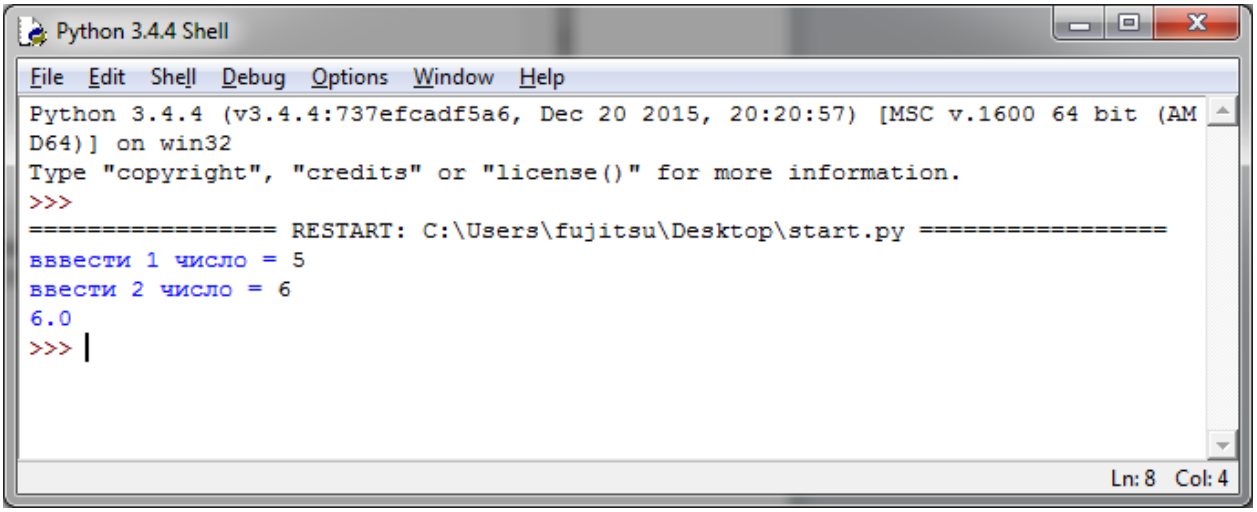
Результат виконання цього програмного коду:



Задача 2. Знайти максимальне з цих двох чисел, використовуючи розгалуження.



Результат виконання цього програмного коду:



Задача 3. Квадратне рівняння.

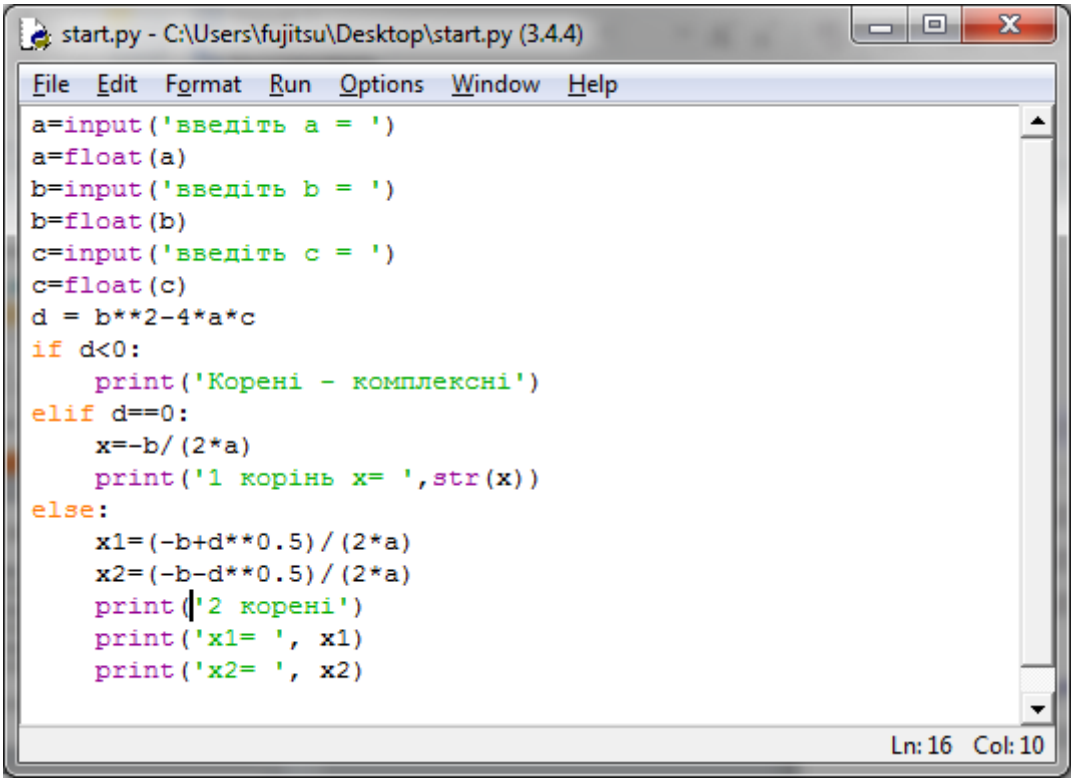
$$ax^2+bx+c=0, \quad a\neq 0$$

$$D=b^2-4ac$$

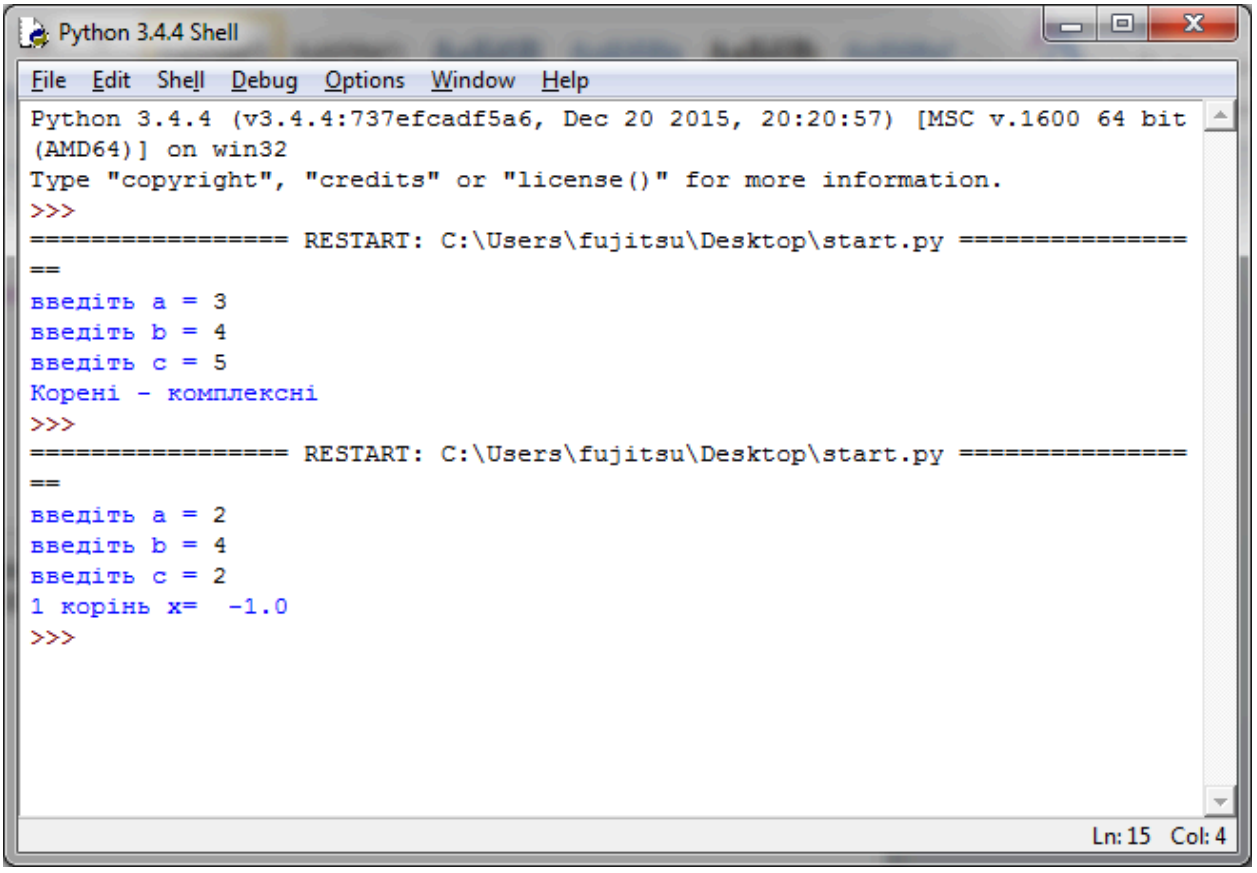
$$>0, \quad x_{1,2}=\frac{-b\pm\sqrt{D}}{2a}$$

$$<0, \quad x$$

$$=0, \quad x_1=x_2=-\frac{b}{2a}$$



Результат виконання цього програмного коду:



Задачі на самостійне виконання

1. Дані два конверта прямокутної форми з довжинами сторін (a, b) та (c, d). Визначити, чи можна один з конвертів вкласти в інший. Відповіддю повинно бути відновідно «YES» чи «NO».
2. За рейтинговою системою оцінка визначається таким чином: якщо сумарний бал учня становить не менше 92% від максимального, то виставляється бал 12, якщо не нижче 70%, то виставляється 9, якщо ж не нижче 50%, то оцінка 6, в інших випадках – оцінка 3. Визначте оцінку учня, якщо він набрав *N*, а максимальне значення сумарного балу становить *S*
3. "Порівняння". Перший пішохід пройшов S_1 км за t_1 годин, другий - S_2 км за t_2 годин. Визначити, швидкість якого з пішоходів була більшою.
4. "Тип трикутника". Задано сторони трикутника a, b, c. Визначити його тип по відношенню до сторін.
5. "Тип трикутника". Задано кути трикутника A і B (в градусах). Визначити його тип по відношенню до сторін.
6. "Тип трикутника". Задано кути трикутника A і B (в градусах). Визначити його тип по відношенню до кутів.
7. Задано площі круга та квадрата. Порівняти периметри указаних фігур.
8. На одному маленькому квадратному острові зі стороною *a* метрів мешкали *k* потерпілих корабельну аварію Робінзонів. Чи не порушені їхні права на житло, якщо на кожного Робінзона

повинно припадати $Sм^2$? Скільки Робінзонів ще вистачить місця на острові, якщо поблизу трапиться нова аварія?

Крок 10. Модуль random

«Генерація випадкових чисел занадто важлива, щоб залишати її на волю випадку»

Роберт Кавью

Python створює випадкові числа на основі формули, так що вони не насправді випадкові, а, як кажуть, псевдовипадкові. Цей спосіб зручний для більшості додатків (крім онлайнових казино).

Модуль random дозволяє генерувати випадкові числа. Перш ніж використовувати модуль, необхідно підключити його за допомогою інструкції: **import random** або **from random import*** (завантажує всю бібліотеку **random).**

Модуль випадкових чисел має декілька функцій:

1. **random.uniform (uniform)**

   random.uniform(<початок>,<кінець>) – відтворює псевдовипадкове дійсне число з діапазону (<початок>,<кінець>);

   Наприклад: **print (random.uniform(1, 6))**= 5.330185127252884

2. **random.randint (randint)**

   random.randint(<початок>,<кінець>) – відтворює псевдовипадкове ціле число з діапазону (<початок>,<кінець>);

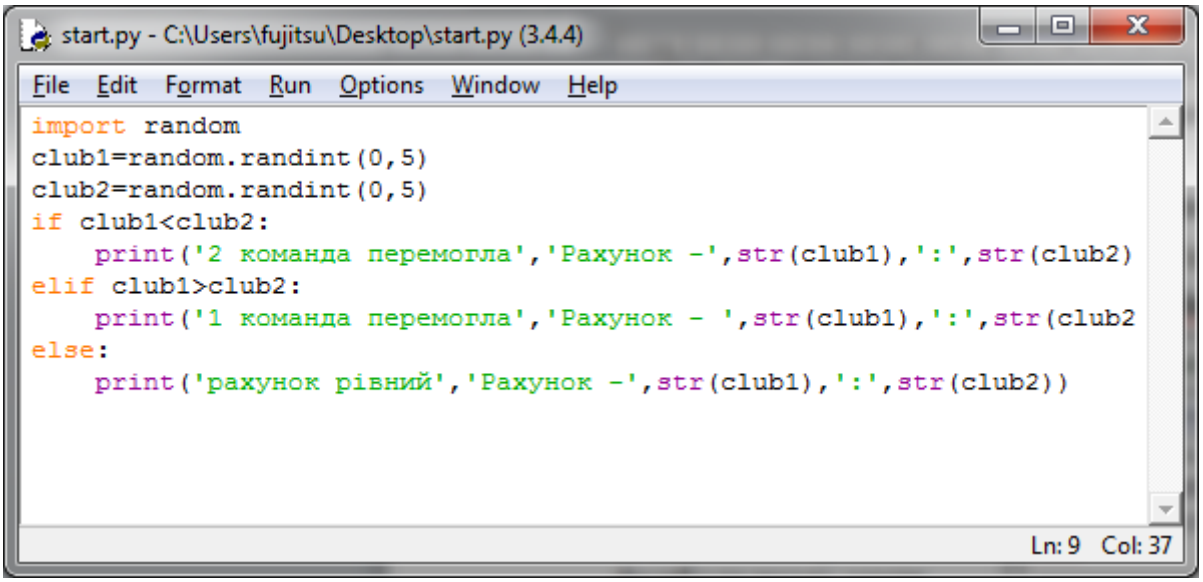
   Наприклад: **print (random.randint(1, 6))**= 4

3. **random.choice (choice)**

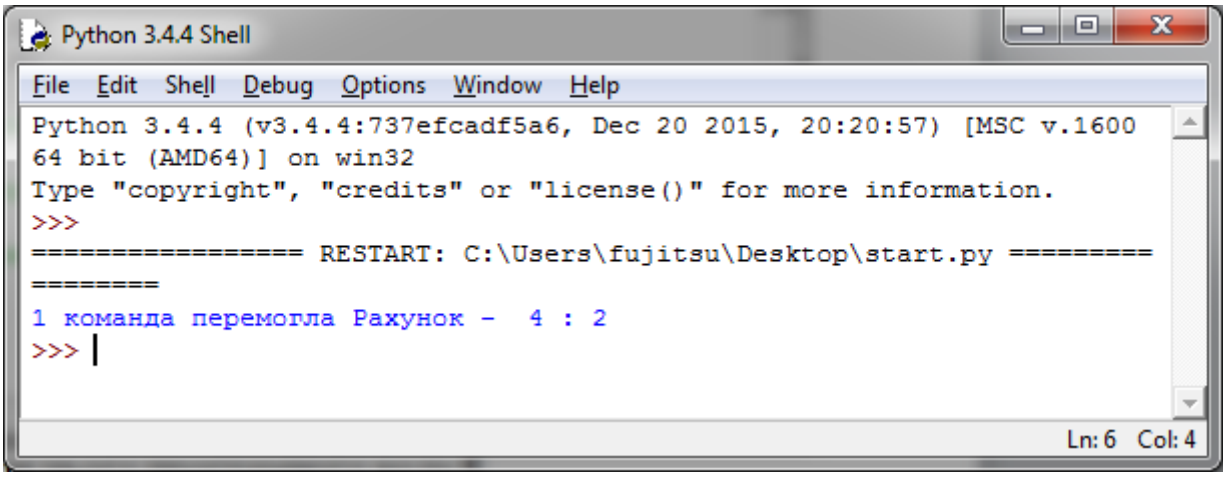
   random.choice(<послідовність>) – відтворює випадковий елемент з будь-якої послідовності (рядки, списки, кортежі);

   Наприклад: **print (random.choice('Україна')) = 'р'**

Задача1. Футбол. Визначити яка з 2 футбольних команд переможе у футбольному матч і з яким рахунком.



Результат виконання цього програмного коду:





Крок 11. Цикли

Алгоритм, в якому певна послідовність команд повторюється кілька разів з новими вхідними даними, називається циклічним.

Цикл – це процес багаторазового повторення дії або послідовності дій

На мові PYTHON є два види циклів: умовні та з лічильником.

Цикл з лічильником.

Цикли з лічильником відрізняються від умовних тим, що в них дії повторюються чітку кількість разів.

Цикл з лічильником записується так:

for i in range(n):

тіло циклу

Змінна **i** пробігає значення певного діапазону записаного у функції **range()** і для кожного значення виконується деяка послідовність команд, яка називається тілом циклу. Як видно із запису, тіло циклу знаходиться правіше оператора **for**.

Розглянемо параметри функції **range()**. У наведеному записі змінна **n** задає інтервал від 0 до **n**-1 включно (всього **n** значень). Тобто, спочатку змінна **i** набуває значення 0 і для цього значення виконуються всі команди у циклі. Наступне значення змінної **i** є 1, потім 2, потім 3 і т.д., поки не досягне значення **n**, для якого цикл вже не виконується. У випадку запуску циклу не з нуля, то вводиться ще один параметр – початкове значення.



Наприклад:

For i in range(3,n):

тіло циклу

Значення змінної **i** розпочинається з 3, потім 4, і т.д. і закінчується n-1.

Якщо ввести ще один параметр, **range(3,n,2)** отримаємо **i** яка набуде значень 3 5 7 9 …., тобто почнеться з 3 і зміниться із кроком 2.

n=17

for i in range(5,n,3):

print(i, ’ ’)

Програма виведе результат:

5 8 11 14 17

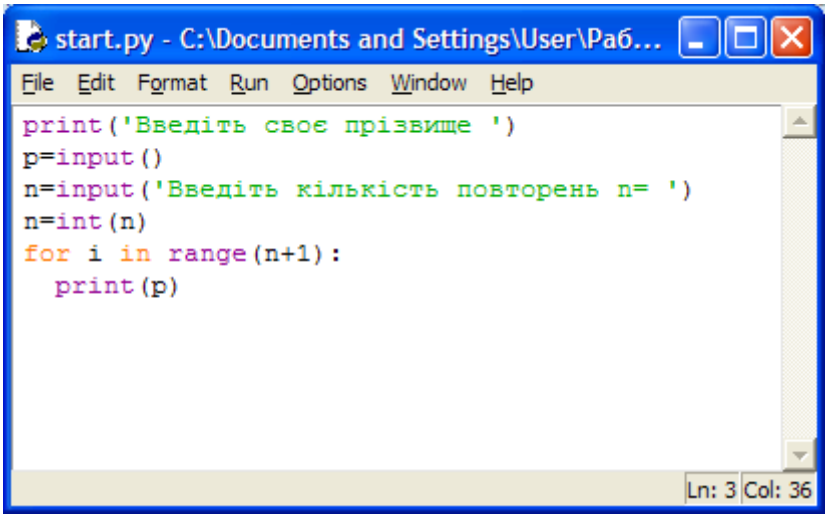
Більш формально можна сказати так:

функція **range(x,y,d)** задає інтервал від **x** до **y**-1 включно з кроком **d**, тобто **x, x+d, x+2d, x+3d,** … поки не досягне значення, рівного **y** або більше, для яких вона не виконується. Якщо значення **d**<0, то повинно бути **x>y**. Наприклад, для виведення чисел від 50 до 10 потрібно записати так:

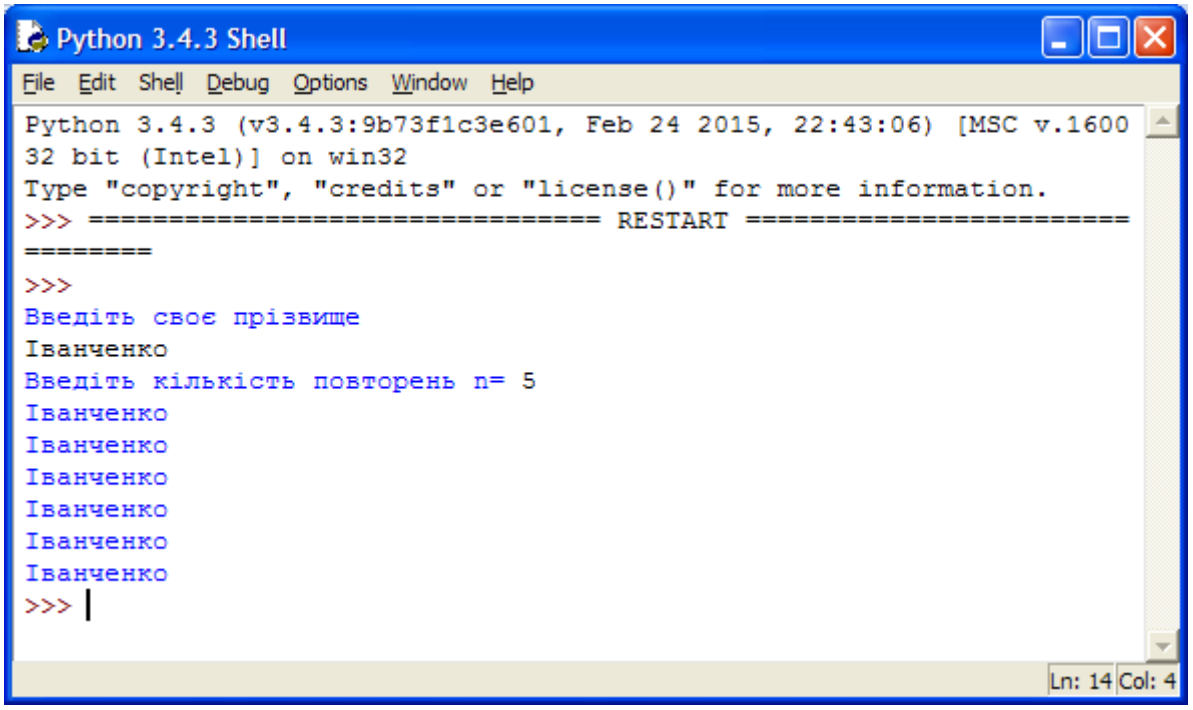
for i in range(50,10,-1):

print(i, end=’ ’)

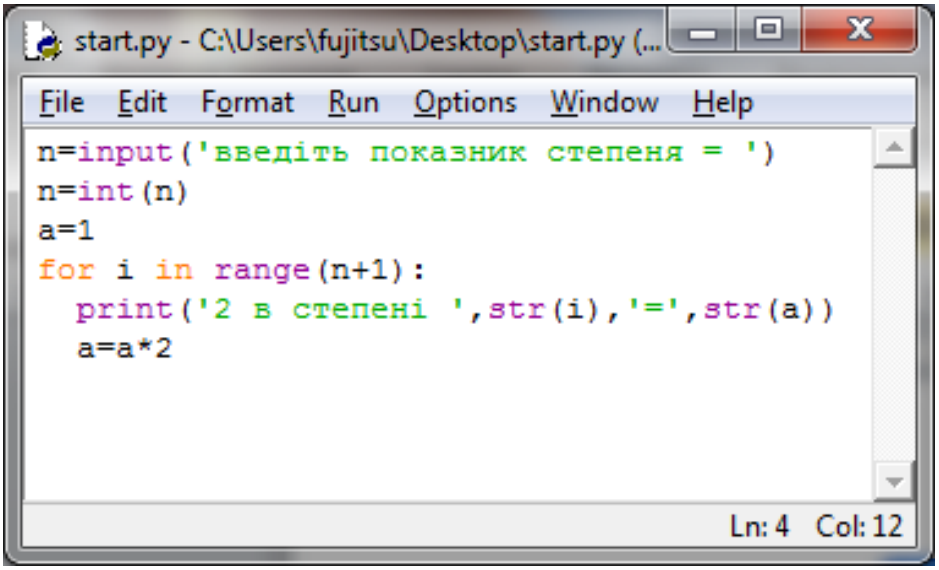
Задача1. Вивести на екран своє прізвище задану кількість разів.



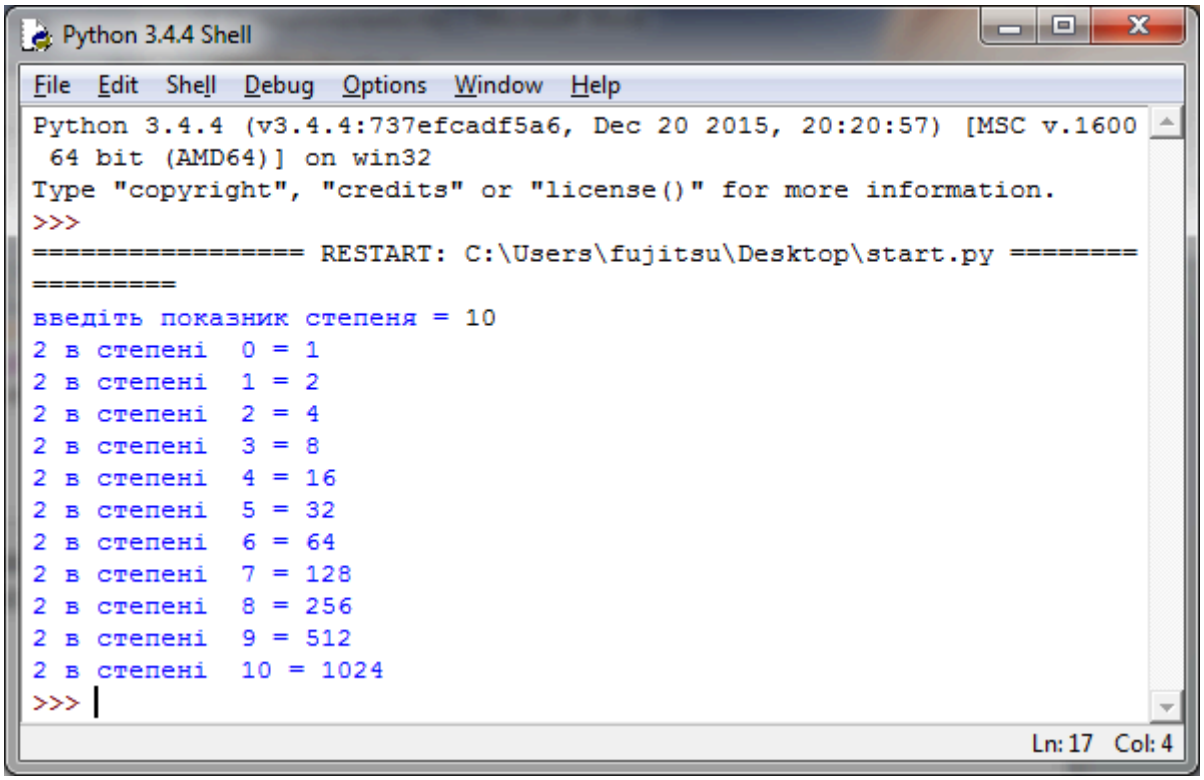
Результат виконання цього програмного коду:



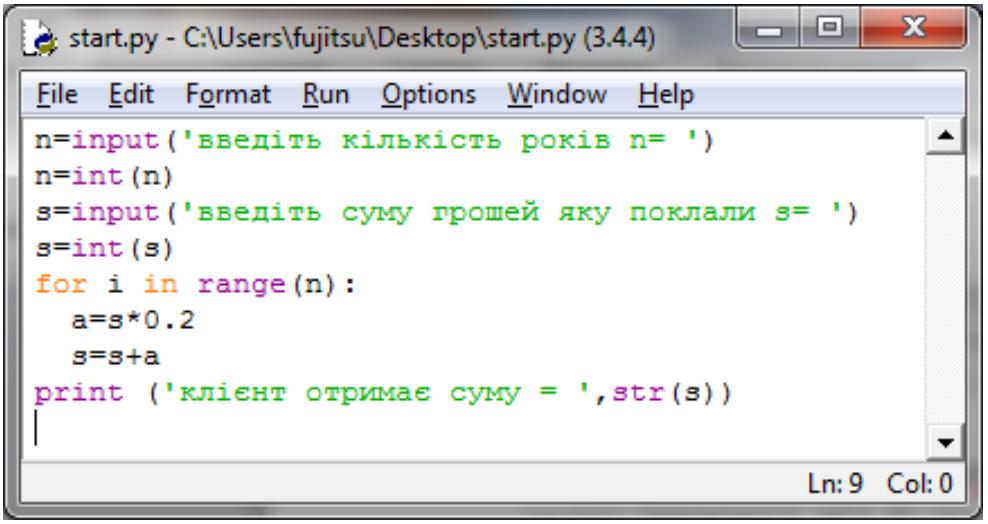
Задача2. Написати програму, яка виводить на екран таблицю степенів двійки.



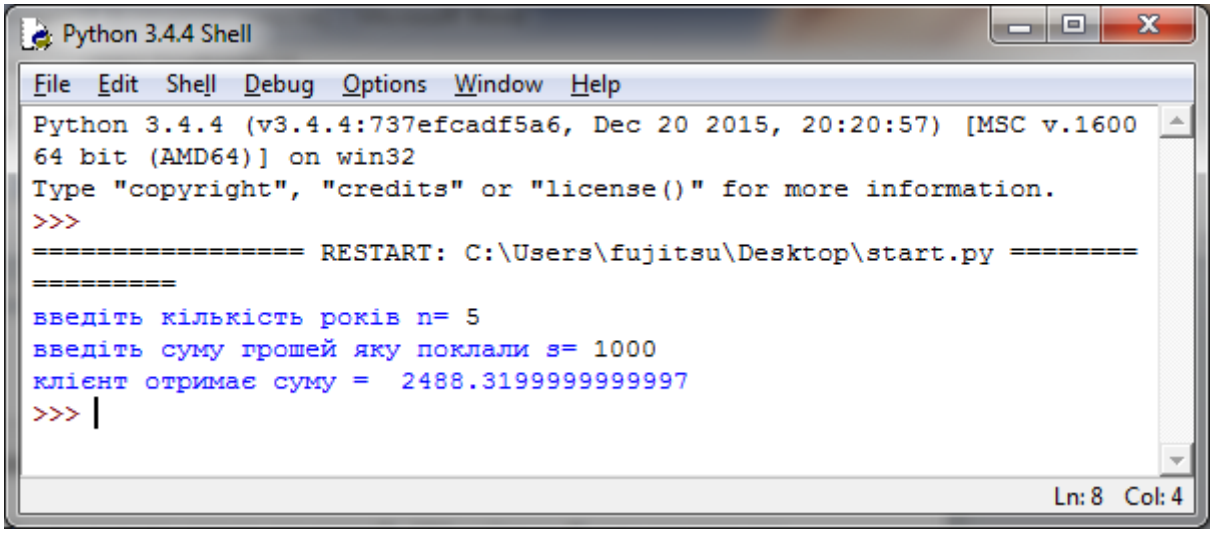
Результат виконання цього програмного коду:



Задача3. Клієнт поклав у банк певну суму грошей. Щорічно банк додає до суми 20% від суми, що зберігається на рахунку після попереднього року. Визначити суму яка буде на рахунку через *n* років.



Результат виконання цього програмного коду:



Задачі на самостійне виконання

1. Знайти суму чисел 1,2,3,4,…,n
2. Написати програму, яка виводить на екран таблицю степенів довільного числа k.
3. У понеділок Толя позичив у Миколки 2 цукерки із задоволенням їх з'їв. У вівторок він позичив удвічі більше цукерок, після чого віддав половину боргу, а решту цукерок знову із задоволенням їх

з’їв. Кожного наступного дня він позичав удвічі більше цукерок, ніж попереднього дня , віддавши з них цілу частину від половини боргу, а решту цукерок із задоволенням з’їдав. Скільки цукерок з’їсть Толя через *n* тижнів? Який у нього при цьому буде борг. Скільки цукерок устигне повернути за цей час Толя Миколці.

4. Компанія бабусь поїхала на мотоциклах на курси з комп’ютерної грамотності. Попереду на мотоциклі без глушника їхала одна бабуся, за нею – дві, потім – три і т.д. скільки бабусь їхало на заняття, якщо приголомшені пішоходи всього нарахували *n* рядів? Чи змогли бабусі зайняти всі місця у класі, якщо там стояло *k* рядів по *l* комп’ютерів у кожному?
5. Обчислити $a(a+1)(a+2)\ldots(a+n-1)$
6. Вивести таблицю температур за Цельсієм від -50 до 50 градусів та їх еквівалент за шкалою Фаренгейта, використовуючи співвідношення *farengeyt=(9*celsiy)/5+32*.
7. Вивести таблицю значень функції $y=x^4-5x^3-2x^2+0.1$ для значень *x*, які змінюються від -5 до 3 з кроком 0,2.

Умовні цикли

В умовних циклах повторення буде відбуватися доти, доки буде виконуватися певна умова.



У мові програмування Python умовний цикл застосовується за допомогою оператора **while**. У коді це буде виглядати так:

while лог. вираз:

тіло циклу

Наприклад:

```
x=1
while x<10:
   print(x**2)
   x+=1          #те саме що й x=x+1
```

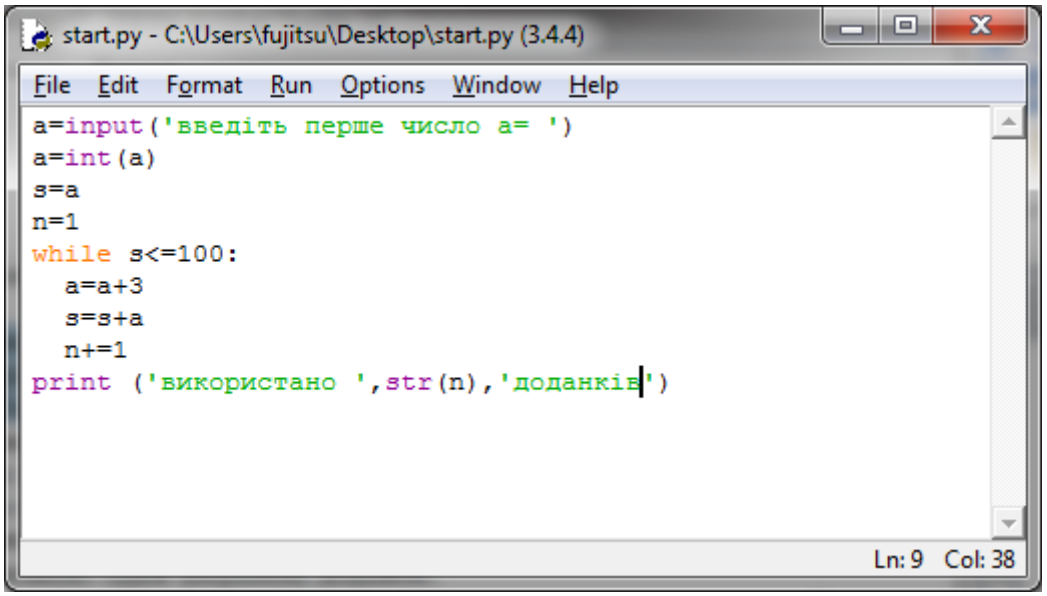
Програма виводить у окремих рядках квадрати одноцифрових чисел. Зверніть увагу, що тіло циклу знаходиться правіше оператора **while.**

Звичайно, що умова циклу може бути як проста так і складна. Наприклад, для того щоб цикл повторювався, поки не дотягнеться певний результат, наприклад, поки позиція об’єкта (i, j) не співпаде з координатою (x, y), цикл запишеться так:

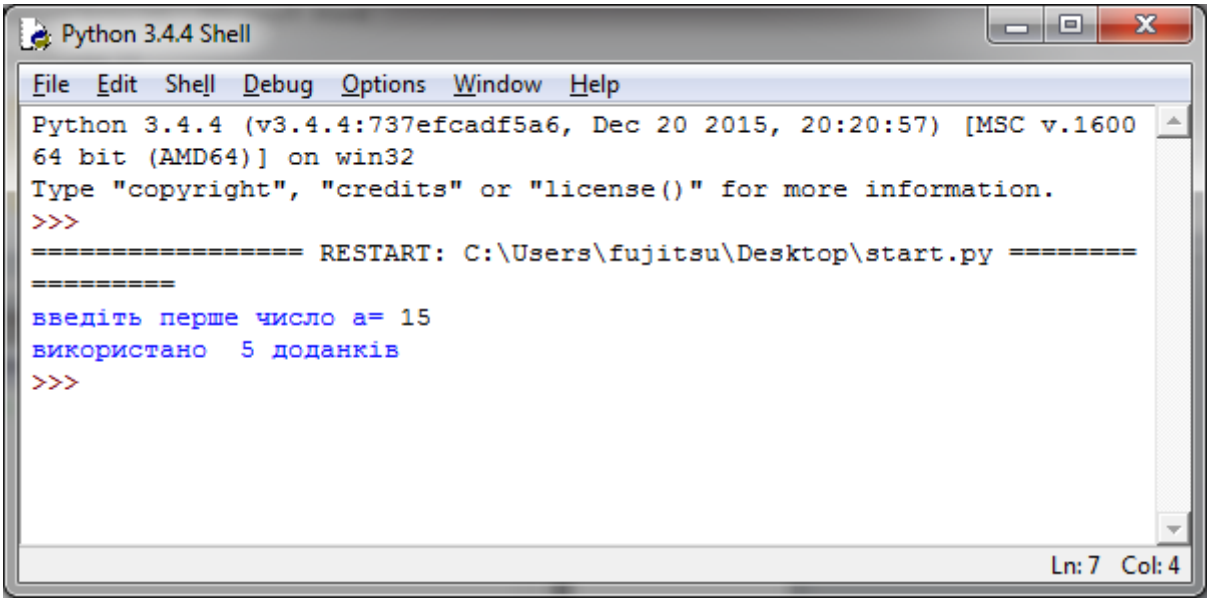
while not(x==i and y==j):

тіло циклу

Задача4. Скільки потрібно взяти доданків, перший з яких дорівнює дійсному числу, а кожний наступний на 3 більше від попереднього, щоб їхня сума перевищила 100?



Результат виконання цього програмного коду:



Задача5. Коли Василині Премудрій виповнилося 18 років Чахлик Невмирущий вирішив взяти її заміж. Василина запитала Чахлика,

скільки в нього скринь із золотом. Чахлик сказав, що в нього зараз *n* скринь і кожний рік додається ще по *m* скринь. Василина пообіцяла, що вийде заміж тоді, коли у Чахлика буде *k* повних скринь із золотом. Скільки років буде тоді нареченій?

start.py - C:\Users\fujitsu\Desktop\start.py (3.4.4)

File Edit Format Run Options Window Help

```
n=input('введіть початкове значення скринь n= ')
n=int(n)
m=input('введіть приріст скринь m= ')
m=int(m)
k=input('введіть кінцеве значення скринь k= ')
k=int(k)
r=18
while n<=k:
  n=n+m
  r+=1
print ('Василині виповнеться ',str(r),'років')
```

Ln: 11 Col: 36

Результат виконання цього програмного коду:

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600
64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\fujitsu\Desktop\start.py ========
=========
введіть початкове значення скринь n= 5
введіть приріст скринь m= 2
введіть кінцеве значення скринь k= 20
Василині виповнеться  26 років
>>>
```

Ln: 9 Col: 4

Задачі на самостійне виконання

1. Обчисліть середнє арифметичне додатних чисел серед чисел, перше з яких дорівнює 100, а кожне наступне менше від попереднього на задане число.
2. Клієнт поклав у банк певну суму грошей. Щорічно банк додає до суми наперед визначений відсоток від суми, що зберігається на рахунку після попереднього року. Через скільки років сума на рахунку перевищить *s* грн.?
3. Капосний папуга навчився висмикувати в дідуся Василя волосся, яке ще залишилося в того на голові. Почавши з однієї волосини, він щодня збільшував порцію вдвічі. Через скільки днів дідусеві не знадобиться гребінець, якщо на початку в нього на голові було аж *n* волосин?
4. Серед чисел $1,\ 1+\frac{1}{2},\ 1+\frac{1}{2}+\frac{1}{3},$ ...перше, яке більше за *a*.
5. Знайти найбільше додатне ціле число *n*, для якого виконується умова: $3n^2-730n<5$; $-4n+84\sqrt{n}+3\geq 0$.
6. Дано ціле число $m > 1$. Одержати найбільше число *k*, при якому $4^k<m$.

Переривання роботи циклів

Якщо під час виконання циклу потрібно відокремити окремий випадок, наприклад, не опрацьовувати його або вийти з циклу, то для цього використовуються оператори **continue** та **break**. Перший оператор переходить на початок циклу (до наступної ітерації), не

опрацьовуючи дані, а другий виходить із циклу. Звичайно, для цих операторів всередині тіла циклу повинна прописана умова, для яких вона виконується.

Наприклад, при виведенні коренів рівняння *(x^3+x^2-10x+8)/(x^2-6x+5)=0* в діапазоні [0;10] необхідно пропускати пошук і виведення кореня для тих випадків, коли знаменник дорівнює нулю, тому слід використати оператор **continue.**

```
for x in range(11):
    if x**2-6*x+5==0:
      continue
    if x**3+x**2-10*x+8==0:
       print(x, end=' ')
```

Програма виведе число 2, бо число 1 пропустить (знаменник дорівнює нулю), а корінь -4 не входить в область пошуку.

Коли ж потрібно зупинитися на відшукання лише одного кореня, то після його знаходження потрібно вийти із циклу за допомогою оператора **break**.

```
for x in range(-10,11):
    if x**2-6*x+5==0:
      continue
    if x**3+x**2-10*x+8==0:
      print(x, end=' ')
       break
```

Програма виведе число -4 і одразу вийде з циклу.



Крок 11. Відображення графічних об’єктів засобами мови програмування

Відображення графічних об’єктів пропоную потренуватися на написанні невеликий графічної програми на Python з використанням **tkinter** (кроссплатформенная бібліотека для розробки графічного інтерфейсу на мові Python).

Для створення малюнка потрібно створити вікно у якому буде відображуватись малюнок за допомогою програмного коду **root = Tk()**

Потім створюється полотно, для майбутнього малюнка і задається його розмір **canv = Canvas(root, width = 640, height = 480**), де width – ширина, height – висота.

Далі розташовуєм полотно всередині вікна з допомогою програмного коду **canv.pack()**

На полотні можна створювати прості двовимірні малюнки за допомогою різних функцій бібліотеки **tkinter.**

Дана бібліотека містить:

canv.create_line(x1,y1,x2,y2, width=size) – малюється лінія від точки з координатами x1,y1 до точки - x2,y2, товщиною size.

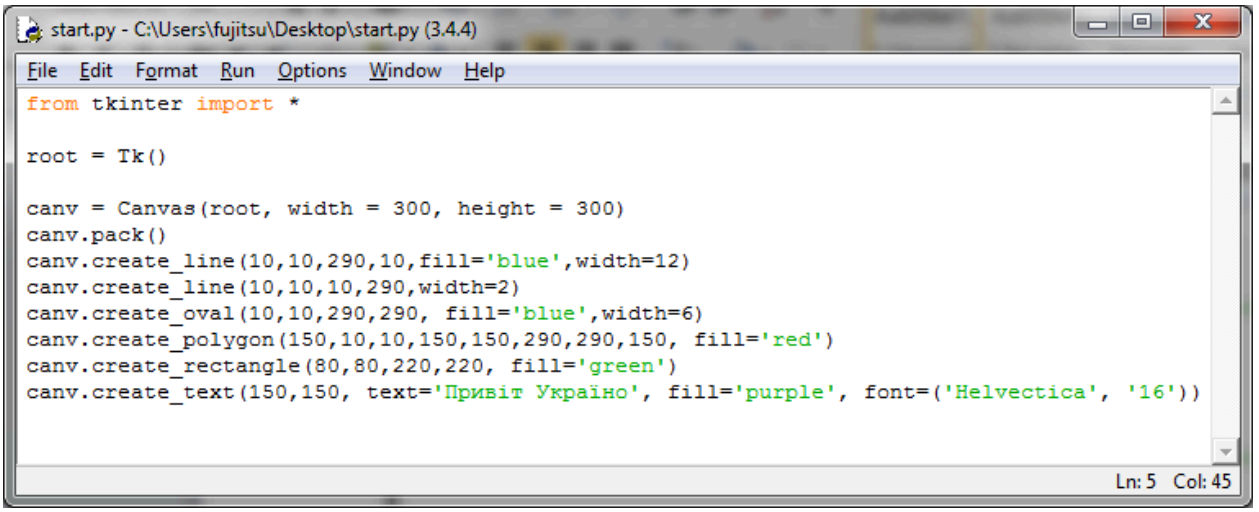
canv.create_oval(x1,y1,x2,y2, fill='колір',width=size) – малює еліпс, вписаний в прямокутник, у якого протилежні вершини розміщені в точках (x1,y1) та (x2,y2), а стороно паралельні осям координат. Прямокутник при цьому не відображається (fill – задає колір зафарбування, width – товщина ліній яка обмежує овал).

canv.create_polygon(x1,y1,x2,y2,x3,y3,….xn,yn, fill='колір',width=size) – малює багатокутник. Для того щоб намалювати багатокутник потрібно просто задати координати його вершин (x1, y1, x2, y2, x3, y3, …, xn, yn), всі вершини послідовно з'єднуються лінією і виходить фігура, яка потім зафарбовується (fill – задає колір зафарбування, width – товщина ліній яка обмежує багатокутник). За допомогою даної функції можна малювати багатокутники будь-якої форми.

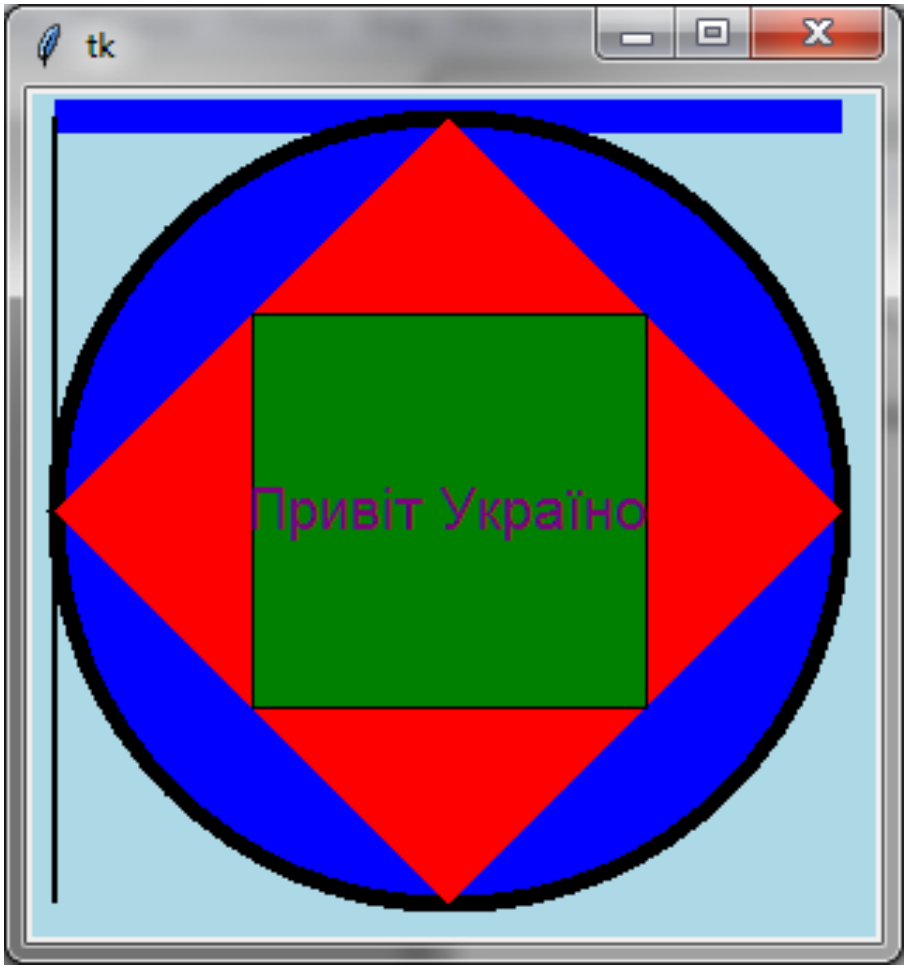
canv.create_rectangle(x1,y1,x2,y2, fill='колір', width=size) – малює прямокутник за координатами протилежних вершин (fill – задає колір зафарбування, width – товщина ліній яка обмежує прямокутник).

canv.create_text(x1,y1, text='текст', fill='колір', font=('тип шрифта', 'розмір')) – додає текст. Координати (x1,y1) даного методу задаються від центру тексту, додаткові параметри також включають колір, шрифт.

Приклад простого малюнку:



Результат виконання цього програмного коду:



Задача1. Створити малюнок за зразком використовуючи цикл з умовою.

from random import * # завантаження бібліотеки випадкових чисел

from tkinter import * # завантаження бібліотеки графічного інтерфейсу

size = 600 # розмір (буде використаний в програмі пізніше)

root = Tk() # створення вікна для малюнка



```
    canv = Canvas(root, width=size, height=size)
                                #створення полотна для малюнка
    canv.pack()                 #розміщення полотна всередині вікна
    colors = ['aqua', 'blue', 'fuchsia', 'green', 'maroon', 'orange',
                    'pink', 'purple', 'red','yellow', 'violet', 'indigo', 'chartreuse',
'lime']
                                #задання списка кольорів
    diapason = 0
    while diapason<1000:
        x0 = randint(0, size) #вибір координати про x з генератора
випадкових чисел
        y0 = randint(0, size) #вибір координати про y з генератора
випадкових чисел
        d = randint(0, size/5)#вибір приросту з генератора випадкових
чисел
        color = choice(colors)      # вибір кольора з випидкового
елемента послідовності
        canv.create_oval(x0, y0, x0+d, y0+d, fill=color )
                                    # малювання еліпса
        root.update()
        diapason+=1             #збільшення діапазону на 1
```

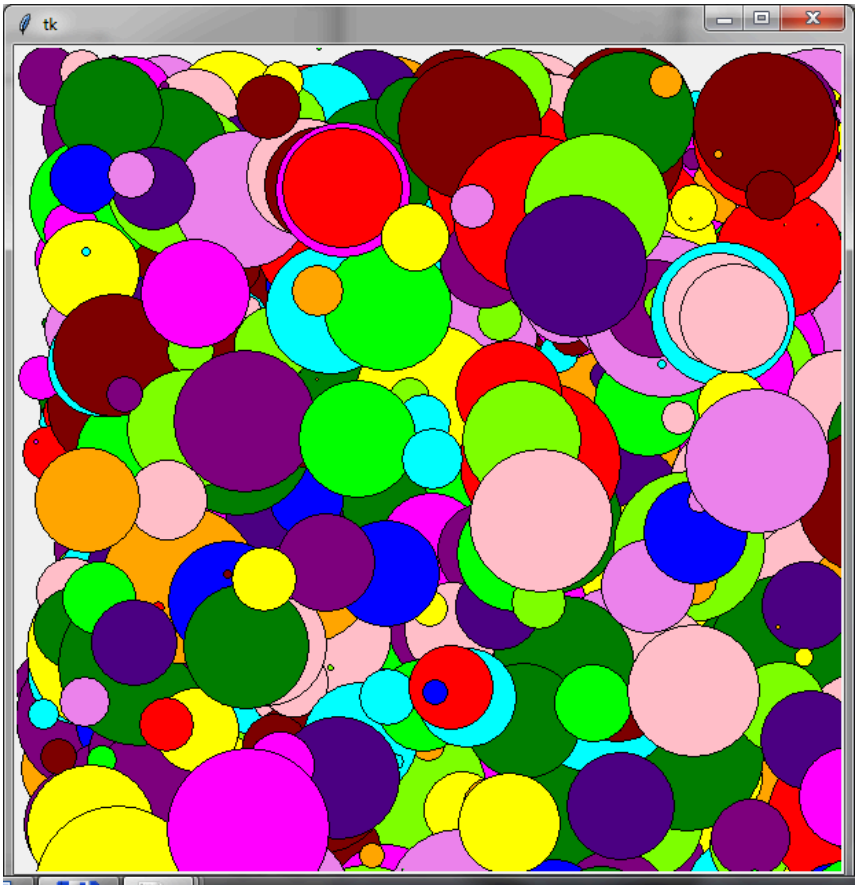
krugi2.py - D:\krugi2.py (3.4.4)

File Edit Format Run Options Window Help

```
from random import * # завантаження бібліотеки випадкових чисел
from tkinter import *# завантаження бібліотеки графічного інтерфейсу
size = 600 # розмір (буде використаний в програмі пізніше)
root = Tk()# створення вікна для малюнка
canv = Canvas(root, width=size, height=size) #створення полотна для малюнка
canv.pack() #розміщення полотна всередині вікна
colors = ['aqua', 'blue', 'fuchsia', 'green', 'maroon', 'orange',
                  'pink', 'purple', 'red','yellow', 'violet', 'indigo', 'chartreuse', 'lime']
#задання списка кольорів
diapason = 0
while diapason<1000:
    x0 = randint(0, size)#вибір координати про x з генератора випадкових чисел
    y0 = randint(0, size)#вибір координати про y з генератора випадкових чисел
    d = randint(0, size/5)#вибір приросту з генератора випадкових чисел
    color = choice(colors)# вибір кольора з випидкового елемента послідовності
    canv.create_oval(x0, y0, x0+d, y0+d, fill=color )# малювання еліпса
    root.update()
    diapason+=1#збільшення діапазону на 1
```

Ln: 19 Col: 0

Результат виконання цього програмного коду:



Бібліографія

1. Інформатика: підруч. Для 8-го кл. загальноосвіт. навч. закл. / Й.Я.Ривкінд [та ін..]. Київ: Генеза, 2016. – 288с.
2. Караванова Т.П. Інформатика: основи алгоритмізації та програмув.: 777 задач з рек. Та прикл.: Навч. Посіб. Для 8-9 кл. із поглибл. Вивч. інф-ки /За заг. ред. М.З.Згуровського –К.: Генеза, 2009. – 286с.
3. Шоу, Зед. Легкий способ выучить Python / Зед Шоу; [пер. с англ. М.А. Райтмана]. Москва: Издательство «Э», 2017. – 352с.

Інтернет-ресурси:

1. https://sites.google.com/site/pythonukr/vstup
2. http://python3school.blogspot.com/
3. http://proginschool.inf.ua/
4. http://ps.readthedocs.io/ru/latest/random.html
5. http://poligraph.blogspot.com/2010/01/tkinter-canvas-python.html

