

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Програмування

Лабораторна робота №2

«Типи даних, змінні та оператори мови програмування Python»

Виконав:
студент групи ІО-32
Крадожон М. Р.
Номер у списку групи: 16
Перевірів:
Пономаренко А. М.

Лабораторна робота №2

Тема: «Типи даних, змінні та оператори мови програмування Python».

Мета: вивчити типи даних, які використовуються в мові програмування Python. Змінні та правила їх іменування, операції над змінними. Оператори та їх застосування.

Загальне завдання:

1. Вивчити матеріал лекцій 3, 4, 5 та 6.
2. Виконати індивідуальне завдання лабораторної роботи, вибране відповідно до варіанту.

Теоретичні основи:

Основні типи даних:

`bool` – логічний тип даних.
`NoneType` – об'єкт зі значенням `None`.
`int` – цілі числа.
`float` – дійсні числа.
`complex` – комплексні числа.
`str` – `Unicode`-рядки.
`bytes` – незмінювана послідовність байтів.
`bytearray` – змінювана послідовність байтів.
`list` – списки.
`tuple` – кортежі.
`Range` – діапазони.
`dict` – словники.
`set` – множини.

Змінні:

~~12x~~ – не може починатися із цифри
~~привіт~~ – не використовуємо кирилицю
~~break~~ – не може збігатися за ключовими словами
~~!a,%b~~ – не може починатися зі службових знаків
`x=23` – присвоєння значення
`x = int(input("x = "))` – ввід значення
`x, y, z = 1, 2, 3` – позиційне присвоювання
`del x` – видалення змінної

Математичні оператори:

`*` – множення.
`/` – ділення.

// – Ділення з округленням униз.

% – остача від ділення.

** – піднесення до степеня.

Унарний мінус (-) і унарний плюс (+) .

+ – додавання.

- – віднімання.

Двійкові математичні оператори:

~ – двійкова інверсія.

& – двійкове І.

| – двійкове АБО.

^ – двійкове виключення.

<< – зсув вліво.

>> – зсув вправо.

Оператори для послідовностей:

+ - конкатенація.

* - повторення.

in - перевірка на входження.

not in - перевірка на невходження.

Оператори присвоювання:

= - присвоює змінній значення.

+= - збільшує значення.

-= зменшує значення.

*= - множить значення.

/= - ділить значення.

//= ділення з округленням вниз й присвоювання.

%= - ділення по модулю й присвоювання.

**= - піднесення до степеня і присвоювання.

Оператори порівняння:

== - дорівнює.

!= - не дорівнює.

< - менше.

> - більше.

<= - менше або дорівнює.

>= - більше або дорівнює.

is - перевіряє, чи посилаються дві змінні на той самий об'єкт

or - логічне АБО .

not – логічне заперечення.

and – логічне І

Оператори розгалуження й цикли:

Оператор розгалуження `if ... else:`

```
if <Логічний вираз>:
    <Блок, виконуваний, якщо умова дійсна>
    [elif <Логічний вираз>:
        <Блок, виконуваний, якщо умова дійсна>
    ]
[else:
    <Блок, виконуваний, якщо всі умови неправильні>
]
```

Оператор циклу `for:`

```
for <Поточний елемент> in <Послідовність>:
    <Інструкції усередині циклу>
[else:
    <Блок, виконуваний, якщо не використовувався оператор break>
]
```

Оператор циклу `while:`

```
for <Початкове значення> while <Послідовність>:
    <Інструкції>
    <Збільшення>
[else:
    <Блок, виконуваний, якщо не використовувався оператор break>
]
```

Функції `range()` і `enumerate()`:

```
range ([<Початок>,) <Кінець> [, <Крок>])
enumerate ( <Об'єкт> [, start=0])
```

Оператор `continue:`

`continue` дозволяє перейти до наступної ітерації циклу до завершення виконання всіх інструкцій всередині циклу.

Оператор `break:`

`break` дозволяє перервати виконання циклу достроково.

Числа:

int – цілі числа.

float – дійсні числа.

complex – комплексні числа.

Двійкові числа починаються з комбінації символів 0b

Вісімкові числа починаються з нуля й наступної за ним латинської букви o (регістр не має значення) і містять цифри від 0 до 7.

Шістнадцяткові числа починаються з комбінації символів 0x (або 0X) і можуть містити цифри від 0 до 9 і букви від A до F.

Операції з фіксованою точністю:

```
>>> from decimal import Decimal
>>> Decimal("0.3") - Decimal("0.1") - Decimal("0.1") -
Decimal("0.1")
Decimal ('0.0')
```

Операції з дробами:

```
from fractions import Fraction
>>> Fraction(4, 5)
Fraction(4, 5)
```

Функції для роботи з числами:

```
bin(<Число>)
oct(<Число>)
hex(<Число>)
float([<Число або рядок>])
round(<Число>[, <Кількість знаків після точки>])
Abs(<Число>) Повертає абсолютне значення
pow(<Число>, <Степінь>[, <Дільник>])
max(<Список чисел через кому>)
min (<Список чисел через кому>)
sum (<Послідовність>[, <Початкове значення>])
divmod(x, y) Повертає кортеж із двох значень (x // y, x % y)
```

Модуль *math*. Математичні функції:

Pi – повертає число π

e – повертає значення константи e

sin(), cos(), tan() – стандартні тригонометричні функції

asin(), acos(), atan() – обернені тригонометричні функції

degrees() – перетворює радіани в градуси

radians() – перетворює градуси в радіани

exp() – експонента

`log(<число> [, <База>])` – логарифм по заданій базі
`log10()` – десятковий логарифм
`log2()` – логарифм по базі 2
`sqrt()` – квадратний корінь
`ceil()` – значення, округлене до найближчого більшого цілого
`floor()` – значення, округлене до найближчого меншого цілого
`pow(<Число>, <Степень>)` – підносить <Число> до <Степеня>
`fabs()` – абсолютне значення
`fmod()` – залишок від ділення
`factorial()` – факторіал числа
`fsum()` – повертає точну суму чисел із заданого списку

Модуль *random*. Генерація випадкових чисел:

```
import random  
from random import *
```

`random()` – повертає псевдовипадкове число від 0.0 до 1.0.
`seed([<параметр>] [, version=2])` – налаштовує генератор випадкових чисел на нову послідовність
`uniform(<початок>, <кінець>)` – повертає псевдовипадкове дійсне число в діапазоні від <Початок> до <Кінець>
`randint(<початок>, <кінець>)` – повертає псевдовипадкове ціле число в діапазоні від <Початок> до <Кінець>
`randrange([<початок>,] <кінець> [, <Крок>])` – повертає випадковий елемент із числової послідовності. Параметри аналогічні параметрам функції `range()`
`choice(<послідовність>)` – повертає випадковий елемент із заданої послідовності (рядка, списку, кортежу)
`shuffle(<список> [, <Число від 0.0 до 1.0>])` – перемішує елементи списку випадковим чином
`sample(<Послідовність>, <Кількість елементів>)` – повертає список із зазначеної кількості елементів, які будуть обрані випадковим чином із заданої послідовності

Індивідуальні завдання

Завдання 1:

Відповідно до номеру у списку вибрати вираз. Написати програму обчислення виразу. Забезпечити ввід даних з клавіатури комп'ютера та друк результатів

обчислень. Вираз: $F = \frac{4x^3 + \ln y}{e^{z+y} + 7,2 \sin y}$.

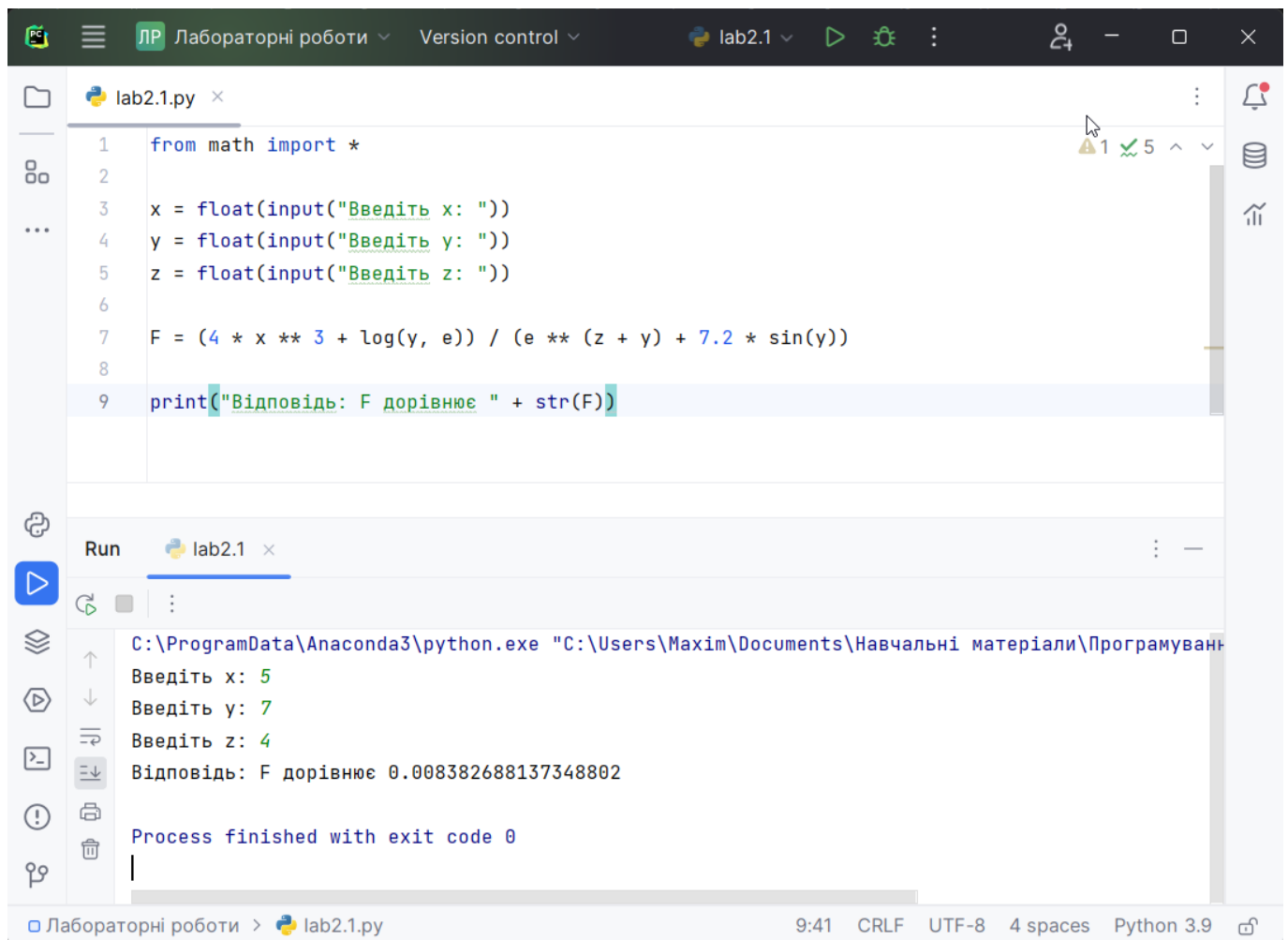
Роздруківка коду:

```
1  from math import *
2
3  x = float(input("Введіть x: "))
4  y = float(input("Введіть y: "))
5  z = float(input("Введіть z: "))
6
7  F = (4 * x ** 3 + log(y, e)) / (e ** (z + y) + 7.2 *
    sin(y))
8  print("Відповідь: F дорівнює " + str(F))
```

Код має відмінний від типового для цього документа шрифт для зручного читання.

Знімок екрана тексту програми:

Завдання програми було ввести змінні та вивести на екран результат обрахунку вираза.



The screenshot shows a code editor window with the file 'lab2.1.py' open. The code is as follows:

```
1  from math import *
2
3  x = float(input("Введіть x: "))
4  y = float(input("Введіть y: "))
5  z = float(input("Введіть z: "))
6
7  F = (4 * x ** 3 + log(y, e)) / (e ** (z + y) + 7.2 * sin(y))
8
9  print("Відповідь: F дорівнює " + str(F))
```

Below the code editor, the 'Run' output is displayed. It shows the execution of the program with the following input and output:

```
C:\ProgramData\Anaconda3\python.exe "C:\Users\Maxim\Documents\Навчальні матеріали\Програмуванн
Введіть x: 5
Введіть y: 7
Введіть z: 4
Відповідь: F дорівнює 0.008382688137348802

Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'lab2.1.py', the encoding is 'UTF-8', the line ending is 'CRLF', and the Python version is 'Python 3.9'.

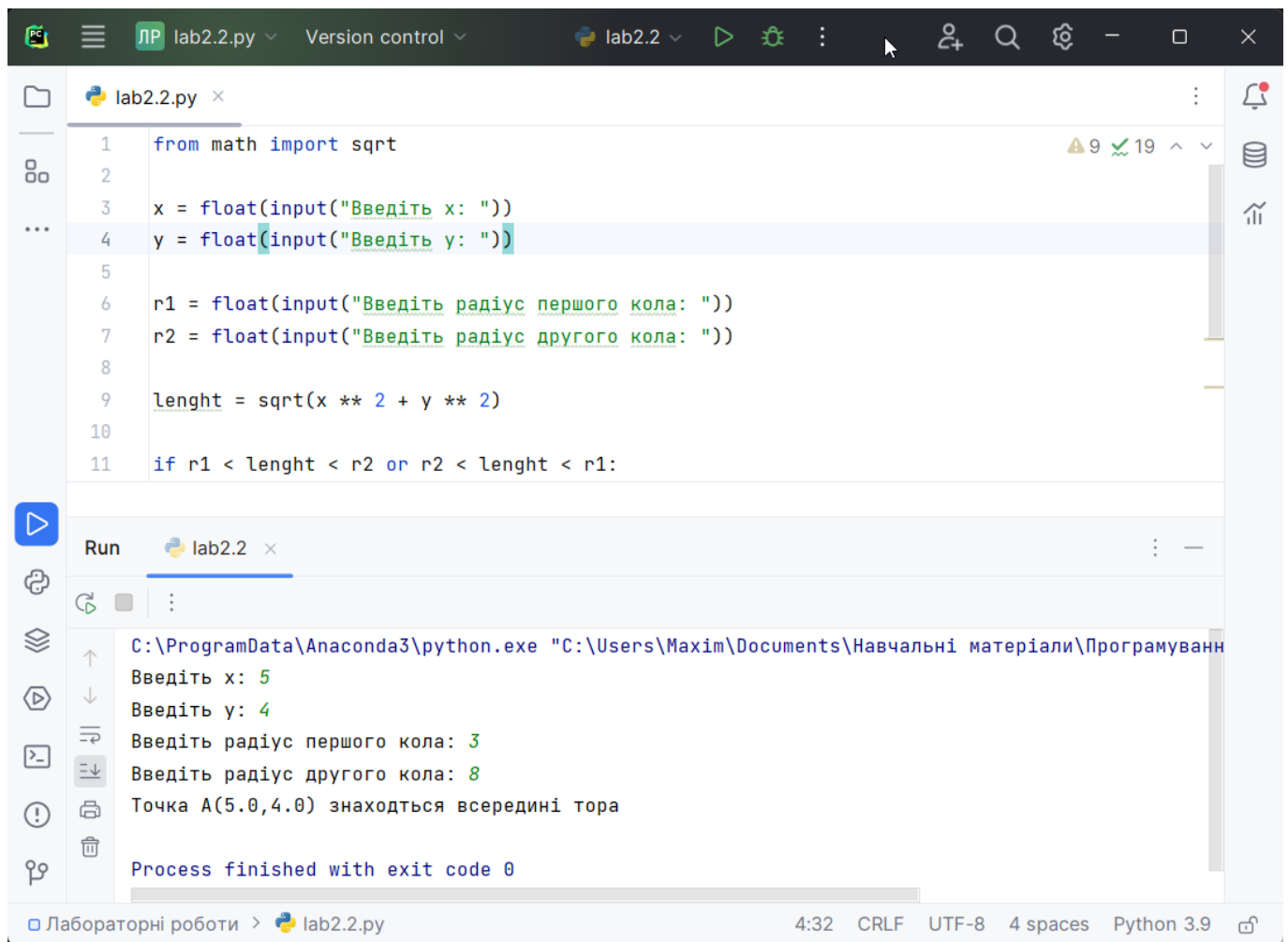
Завдання 2:

Відповідно до номеру у списку вибрати індивідуальне завдання. Написати програму на мові Python. Забезпечити ввід даних з клавіатури комп'ютера та друк результатів обчислень. У звіті до лабораторної роботи описати алгоритм, за яким побудована програма. Ввести з клавіатури координати точки $A(x,y)$. Визначити, чи лежить дана точка всередині тора, утвореного колами з радіусами r і R з центром в точці $O(0,0)$. Відповідь вивести у вигляді повідомлення.

Роздруківка коду:

```
1  from math import sqrt
2
3  x = float(input("Введіть x: "))
4  y = float(input("Введіть y: "))
5
6  r1 = float(input("Введіть радіус першого кола: "))
7  r2 = float(input("Введіть радіус другого кола: "))
8
9  lenght = sqrt(x ** 2 + y ** 2)
10
11 if r1 < lenght < r2 or r2 < lenght < r1:
12     print("Точка A(" +str(x)+ "," +str(y)+ ") знаходиться
    всередині тора")
13
14 else:
15     print("Точка A(" +str(x)+ "," +str(y)+ ") не
    знаходиться всередині тора")
```


Знімок екрана тексту програми:



```
1 from math import sqrt
2
3 x = float(input("Введіть x: "))
4 y = float(input("Введіть y: "))
5
6 r1 = float(input("Введіть радіус першого кола: "))
7 r2 = float(input("Введіть радіус другого кола: "))
8
9 lenght = sqrt(x ** 2 + y ** 2)
10
11 if r1 < lenght < r2 or r2 < lenght < r1:
```

Run lab2.2

```
C:\ProgramData\Anaconda3\python.exe "C:\Users\Maxim\Documents\Навчальні матеріали\Програмуванн
Введіть x: 5
Введіть y: 4
Введіть радіус першого кола: 3
Введіть радіус другого кола: 8
Точка A(5.0,4.0) знаходиться всередині тора
Process finished with exit code 0
```

Лабораторні роботи > lab2.2.py 4:32 CRLF UTF-8 4 spaces Python 3.9

Алгоритм побудови програми: Програма імпортує змінну `sqrt` з модуля `math`, потім користувачеві треба ввести координати точки та радіусів кіл, потім програма рахує довжину відрізка $[O, A]$ та порівнює її з радіусами кіл. Якщо довжина менше радіуса більшого кола та більше радіуса меншого кола, то програма виводить повідомлення, що точка A знаходиться усередині тора; інакше виводить повідомлення, що точка A не лежить усередині тора.

Завдання 3:

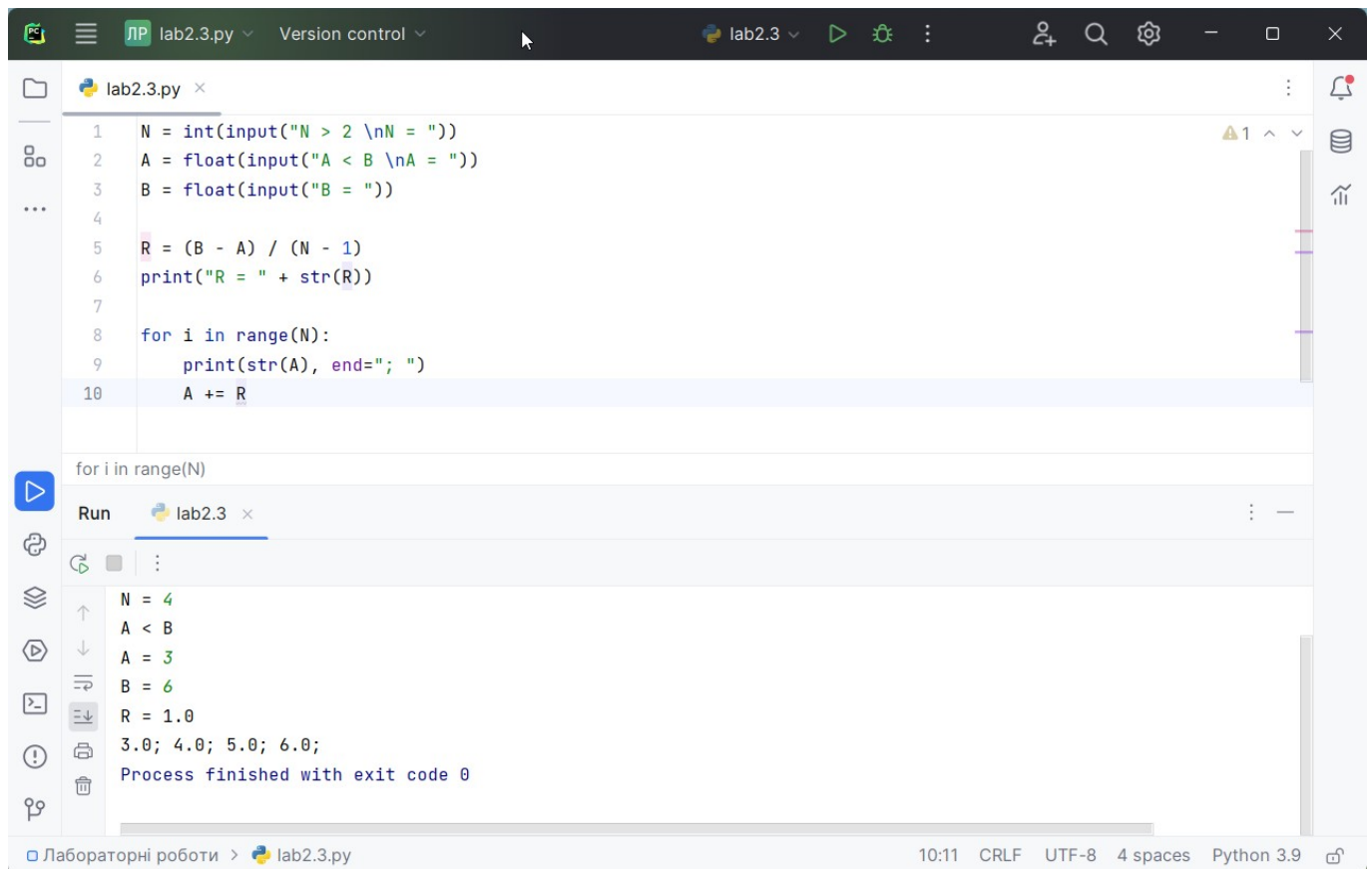
Відповідно до номеру у списку групи вибрати індивідуальне завдання. Написати програму на мові Python. Забезпечити ввід даних з клавіатури комп'ютера та друк результатів обчислень. У звіті до лабораторної роботи описати алгоритм, за яким побудована програма.

Ввести з клавіатури ціле число $N (> 2)$ і дві дійсні точки на числовій осі: A, B ($A < B$). Відрізок $[A, B]$ розбитий на рівні відрізки довжини H з кінцями в N точках з координатами $A, A + H, A + 2H, A + 3H, \dots, B$. Вивести значення H і набір з N точок, який утворює розбиття відрізка $[A, B]$.

Роздруківка коду:

```
1  N = int(input("N > 2 \nN = "))
2  A = float(input("A < B \nA = "))
3  B = float(input("B = "))
4
5  R = (B - A) / (N - 1)
6  print("R = " + str(R))
7
8  for i in range(N):
9      print(str(A), end="; ")
10     A += R
```

Знімок екрана тексту програми:



The screenshot shows a code editor with a dark theme. The top bar includes a file explorer, a search bar, and a version control dropdown. The main editor area displays the Python code from the previous block. Below the code editor is a 'Run' button and a console window. The console window shows the output of the program: 'N = 4', 'A < B', 'A = 3', 'B = 6', 'R = 1.0', and '3.0; 4.0; 5.0; 6.0;'. The console also indicates that the process finished with exit code 0. The bottom status bar shows the file path 'Лабораторні роботи > lab2.3.py', the time '10:11', and the encoding 'CRLF UTF-8 4 spaces Python 3.9'.

```
lab2.3.py x
1  N = int(input("N > 2 \nN = "))
2  A = float(input("A < B \nA = "))
3  B = float(input("B = "))
4
5  R = (B - A) / (N - 1)
6  print("R = " + str(R))
7
8  for i in range(N):
9      print(str(A), end="; ")
10     A += R

for i in range(N)
Run lab2.3 x
N = 4
A < B
A = 3
B = 6
R = 1.0
3.0; 4.0; 5.0; 6.0;
Process finished with exit code 0
Лабораторні роботи > lab2.3.py 10:11 CRLF UTF-8 4 spaces Python 3.9
```

Алгоритм побудови програми: Програма очікує поки користувач введе ціле число N та значень двох точок A та B , потім після цього, програма рахує довжину N та друкує її значення, а потім друкує рядок зі значеннями N точок, які ділять відрізок $[A, B]$ на відрізки, які рівні H .

Висновок: Виконавши цю лабораторну роботу, я зміг здобути відповідні навички в типах даних, змінних та операторах; зміг розробити тестову програму. Під час виконання лабораторної роботи проблем не виникало, а складність була в структуруванні коду та приведенні його до більш гарного вигляду.