Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Програмування**

**Лабораторна робота №2**

«Типи даних, змінні та оператори мови програмування Python»

Виконав:

студент групи ІВ-81

Федорусов І.М.

Залікова книжка № ІВ-8128

Перевірив Новотарський М. А.

Київ - 2018 р.

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

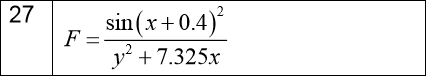
**Загальне завдання:**

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

**Індивідуальні завдання:**

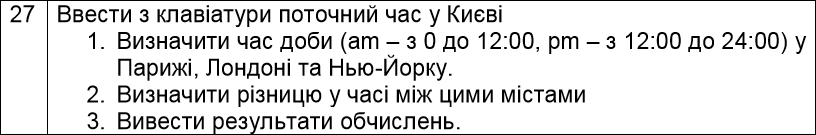
Завдання 1:

1. Відповідно до номера у списку вибрати номер варіанта виразу.
2. Написати програму обчислення виразу.
3. Забезпечити ввід значень для всіх змвнних, що знаходяться в правій частині виразу, з клавіатури комп’ютера.
4. Программа повинна працювати з дійсними числами.
5. Забезпечити вивід результатів обчислень.
6. Програма обчислення виразу повинна коректно працювати при всіх можливих значеннях змінних.



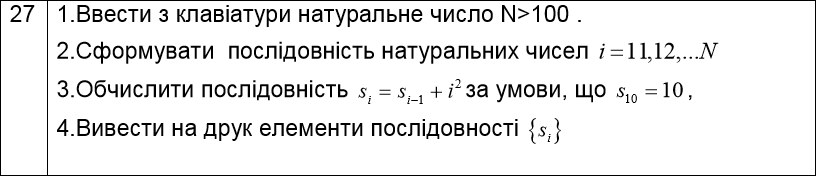
Завдання 2:

1. Відповідно до номера у списку вибрати індивідуальне завдання.
2. Написати програму на мові Python.
3. Забезпечити ввід даних з клавіатури комп’ютера
4. Забезпечити вивід результатів обчислень.
5. У звіті до лабораторної роботи описати алгоритм, за яким побудована програма.



Завдання 3:

Відповідно до номера 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.



**Теоретичні відомості:**

Основні типи даних:

* bool – логічний тип даних.
* NoneTуре – 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 з цифри
* привіт – не використовуємо кирилицю
* break – не повинна співпадати з ключовими словами
* !a, %b – не починаємо зі службових знаків
* x=23 – присвоєння значення
* х = int(input("x = "))– ввід значення
* х, y, z = 1, 2, 3 – позиційне присвоювання
* del х – видалення змінної

Математичні оператори

* + – додавання.
* - – віднімання.
* \* – множення.
* / – ділення.
* / / – Ділення з округленням униз.
* % – остача від ділення.
* \*\* – піднесення до степеня.
* Унарний мінус (-) і унарний плюс ( +) .

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**

* <Початкове значення>
* while <Условие>:
  + <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 А 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 (х //y, х % y)

Модуль math. Математичні функції

* Pi – повертає число π
* е – 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.
* ехр() – 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>,<Степінь>) – підносить <Число> до <Степеня>.
* fabs ( ) – абсолютне значення.
* fmod()– залишок від ділення.
* factorial () – факторіал числа.
* fsum(<Cписок чисел>) – 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(<cписок>[, <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.

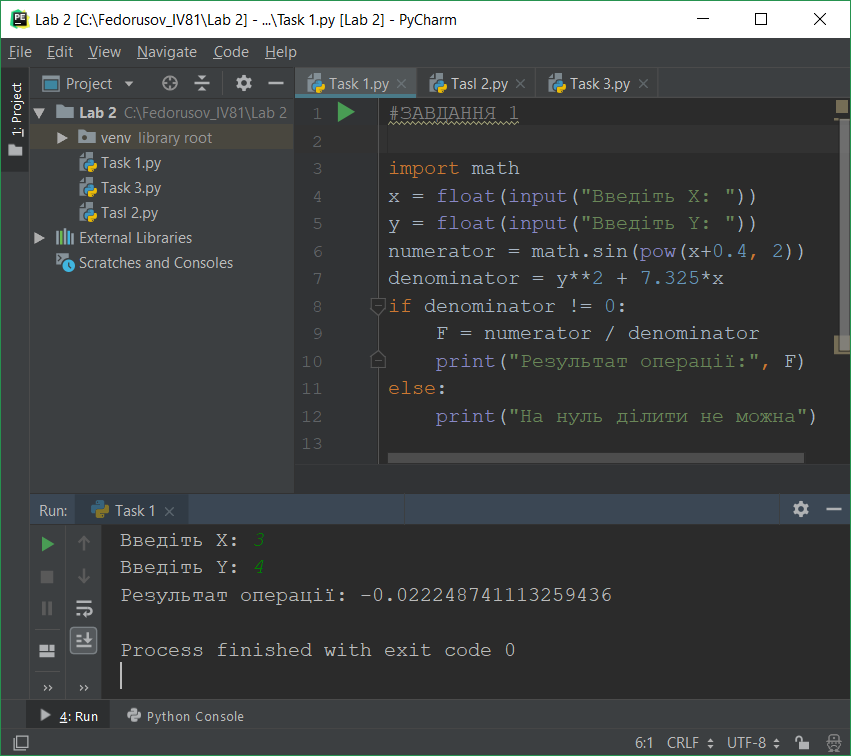
**Хід роботи:**

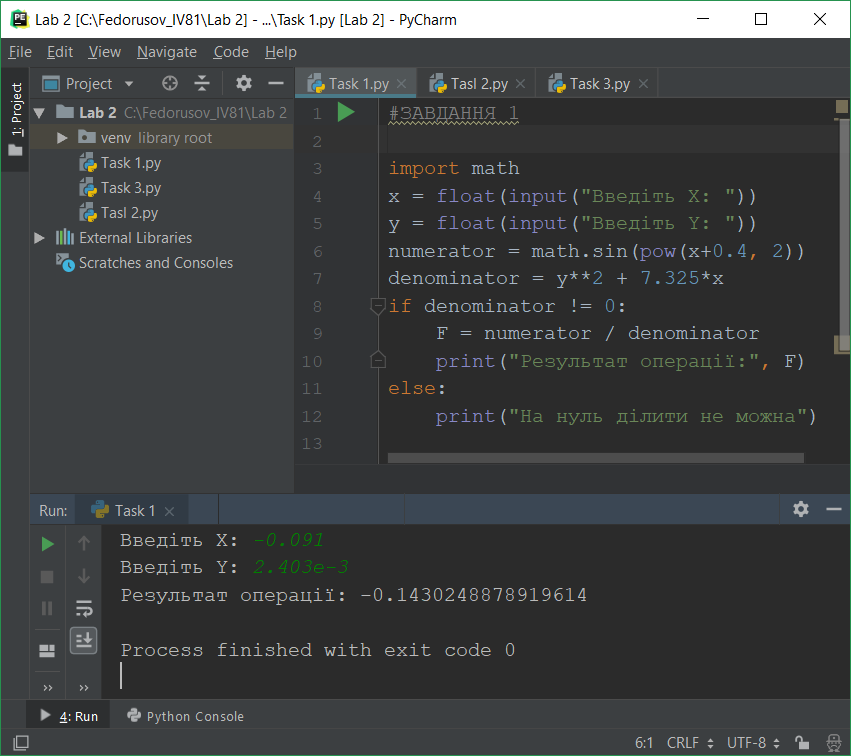
Завдання 1:

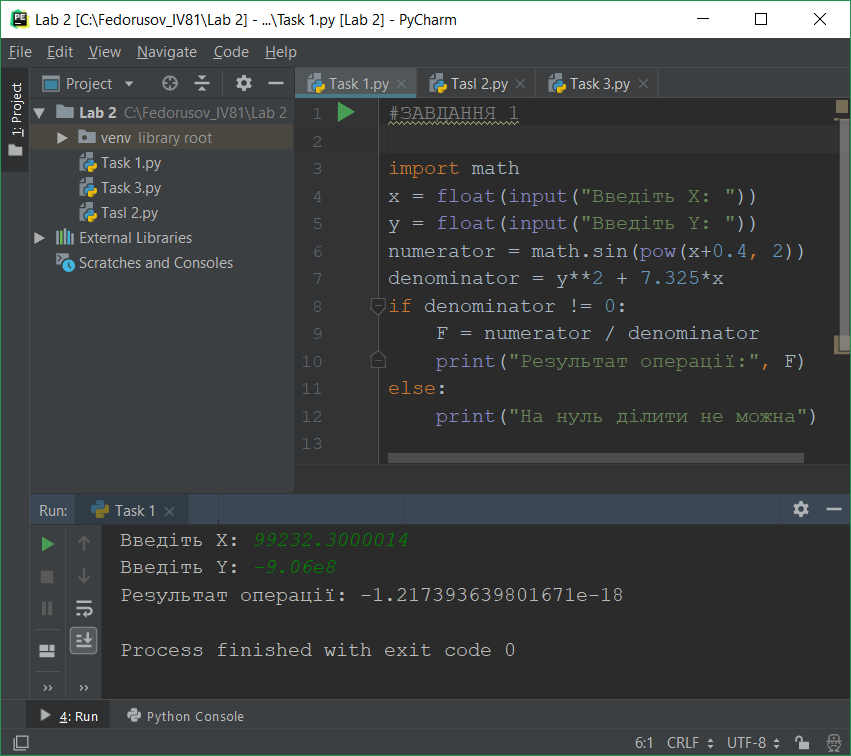
Текст програми:

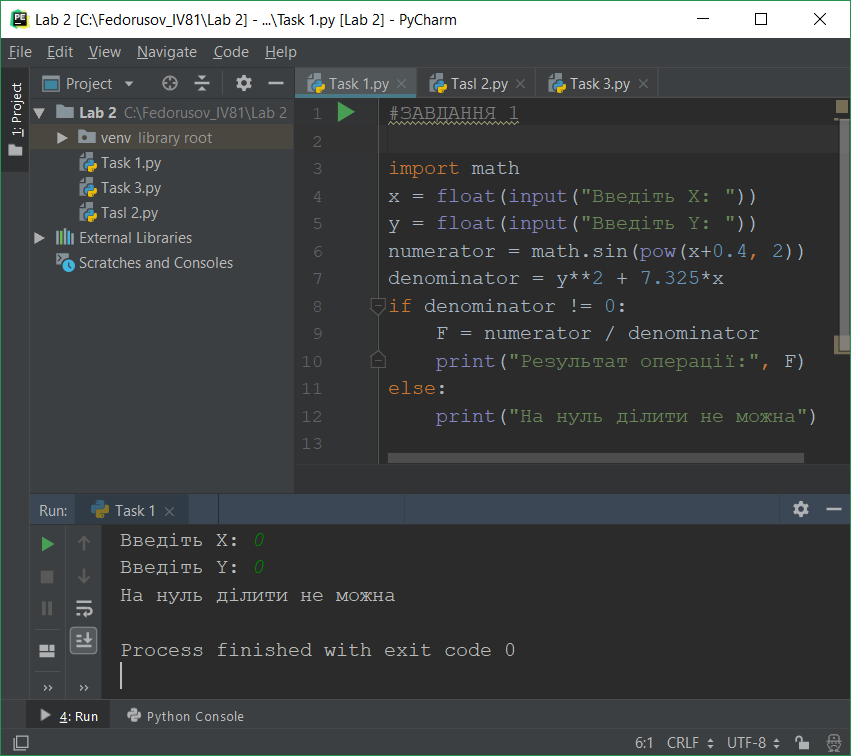
import math  
x = float(input("Введіть X: "))  
y = float(input("Введіть Y: "))  
numerator = math.sin(pow(x+0.4, 2))  
denominator = y\*\*2 + 7.325\*x  
F = numerator / denominator  
print("Результат операції:", F)

Перевірка на вхід різних варіантів значень:









Завдання 2:

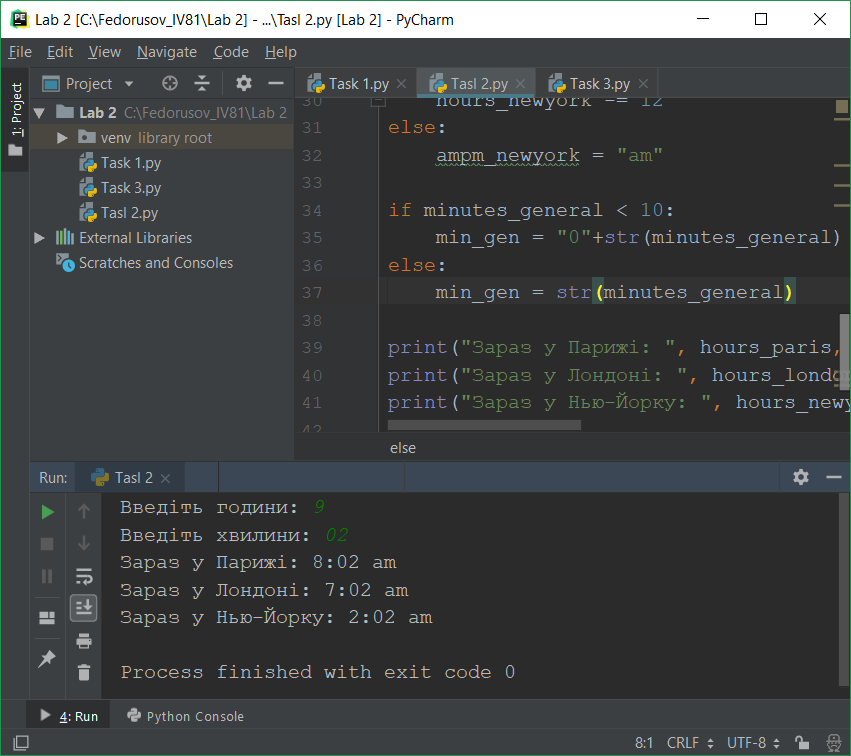
Текст програми:

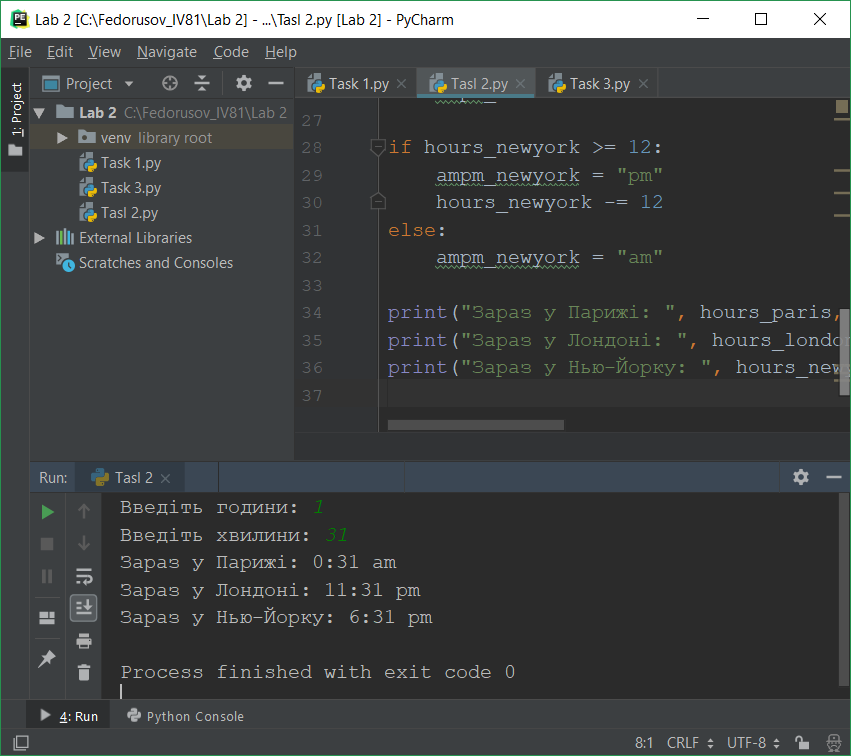
hours\_kiev = int(input("Введіть години: "))  
minutes\_general = int(input("Введіть хвилини: "))  
  
#вираховування часу в інших містах  
  
hours\_paris = hours\_kiev - 1  
if hours\_paris < 0:  
 hours\_paris+=24  
  
hours\_london = hours\_kiev - 2  
if hours\_london < 0:  
 hours\_london+=24  
  
hours\_newyork = hours\_kiev - 7  
if hours\_newyork < 0:  
 hours\_newyork+=24  
  
#перевірка на am,pm   
  
if hours\_paris >= 12:  
 ampm\_paris = "pm"  
 hours\_paris -= 12  
else:  
 ampm\_paris = "am"  
  
if hours\_london >= 12:  
 ampm\_london = "pm"  
 hours\_london -= 12  
else:  
 ampm\_london = "am"  
  
if hours\_newyork >= 12:  
 ampm\_newyork = "pm"  
 hours\_newyork -= 12  
else:  
 ampm\_newyork = "am"  
  
#якщо треба, зробити вигляд часу більш гарнішим  
  
if minutes\_general < 10:  
 min\_gen = "0"+str(minutes\_general)  
else:  
 min\_gen = str(minutes\_general)

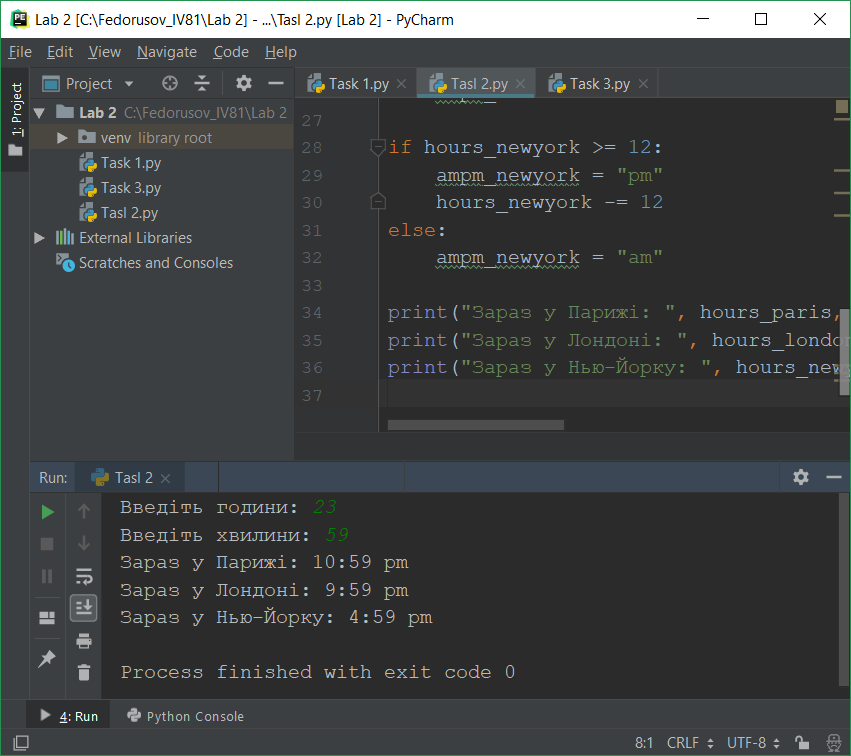
#вивід

print("Зараз у Парижі: ", hours\_paris, ":", min\_gen, " ",ampm\_paris, sep="")  
print("Зараз у Лондоні: ", hours\_london, ":", min\_gen, " ",ampm\_london, sep="")  
print("Зараз у Нью-Йорку: ", hours\_newyork, ":", min\_gen, " ",ampm\_newyork, sep="")

Перевірка на вхід різних варіантів значень:







Опис алгоритму, за яким працює програма:

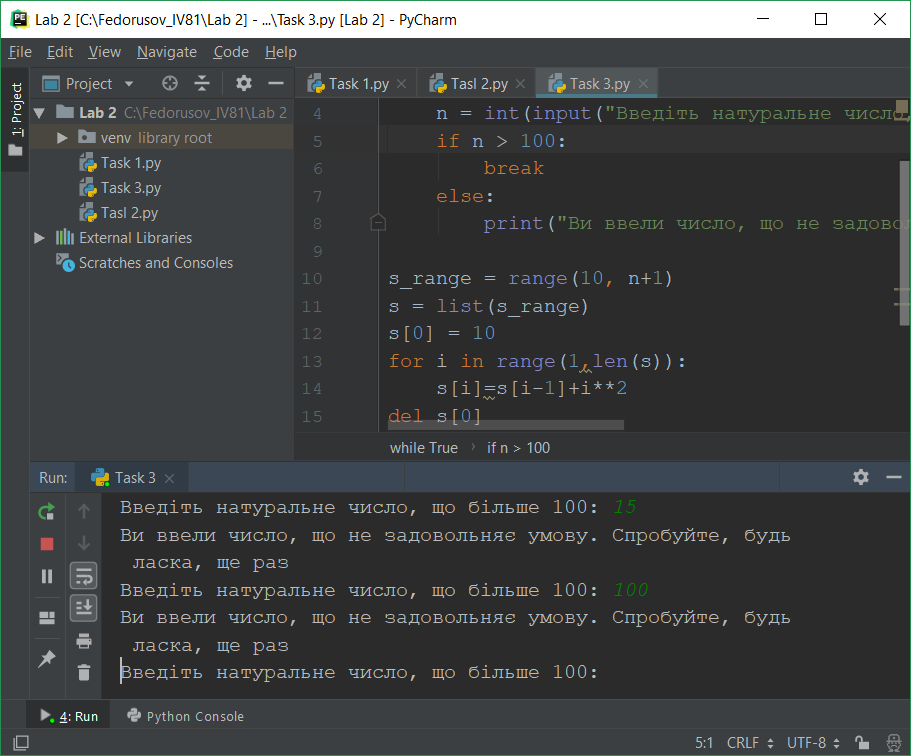
1. Програма отримує на вхід почергово значення hours\_kiev та minutes\_general, що відповідають за час(години) в Києві та хвилини(в усьому світі це прийнято узагальнити) відповідно.
2. Програма обчислює час(години) в інших містах(Париж, Лондон, Нью-Йору) згідно різниці часу між Києвом та цими містами, яку можна знайти у вільному доступі в Інтернеті. Якщо отримані дані менше нуля, тоді додати 24, щоб утворилося число згідно норм запису добового часу. Перевірка на норми запису відбувається кожного разу, коли програма вираховує час у інших містах.
3. Програма перевіряє час на позначку «am\pm», тобто «до полудня\після полудня» та форматує час згідно результату.
4. Якщо хвилини менше 10, то програма перетворює змінну з хвилинами в строку, щоб потім конкантенувати з ‘0’ перед нею, щоб отримати більш прийнятну форму запису часу
5. Програма виводить дані

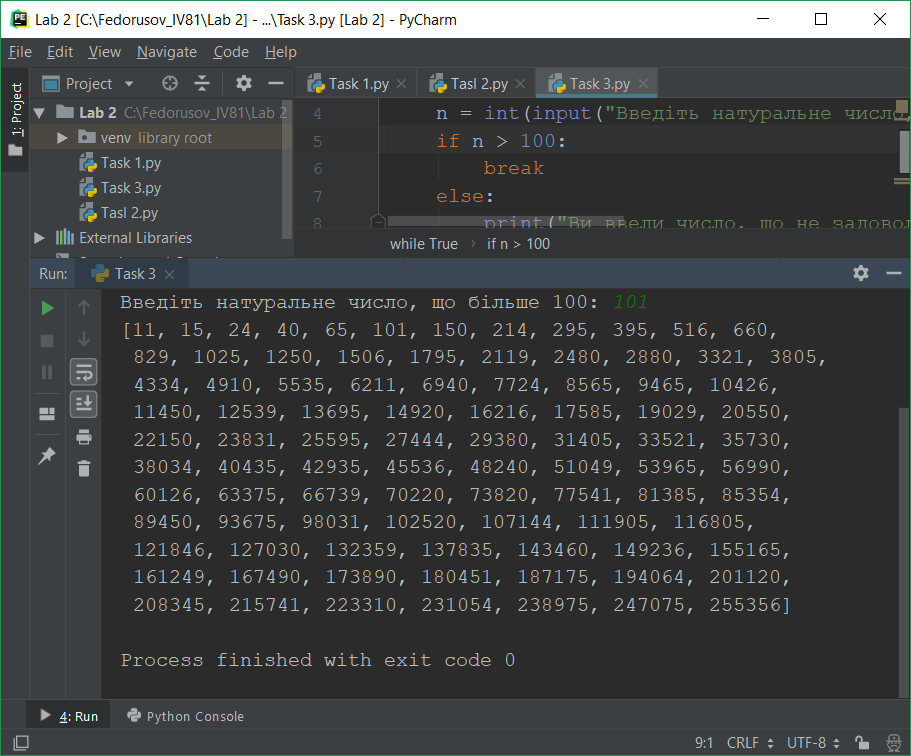
Завдання 3:

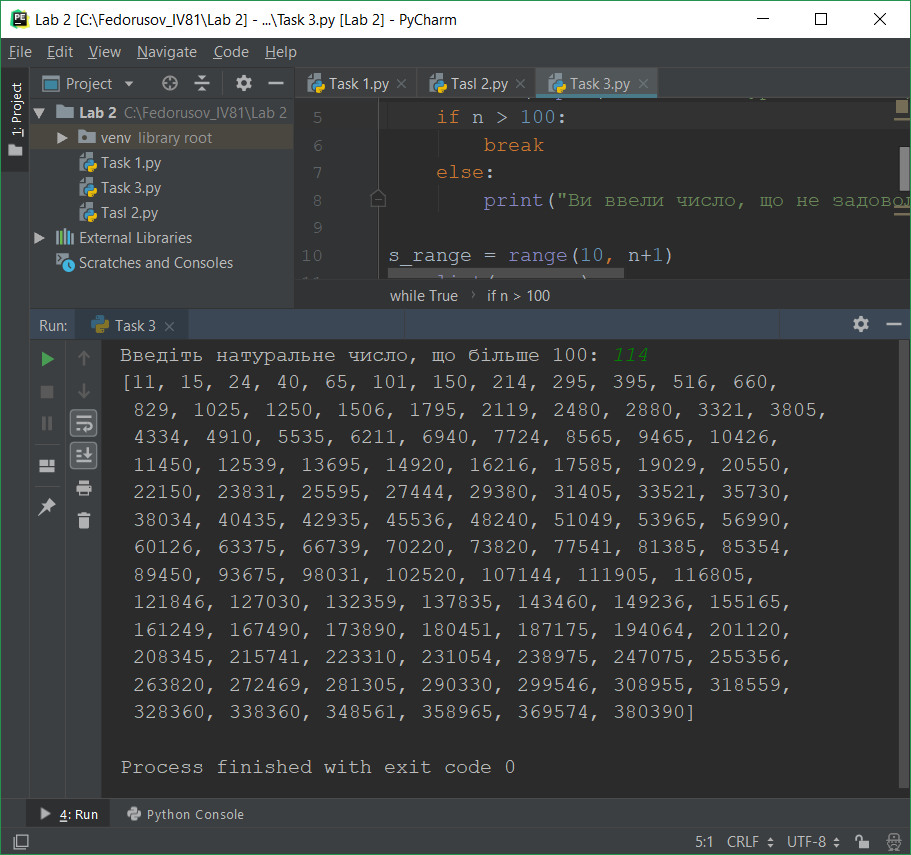
Текст програми

while True:  
 n = int(input("Введіть натуральне число, що більше 100: "))  
 if n > 100:  
 break  
 else:  
 print("Ви ввели число, що не задовольняє умову. Спробуйте, будь ласка, ще раз")  
  
s = list(range(10, n+1))  
s[0] = 10  
for i in range(1,len(s)):  
 s[i]=s[i-1]+i\*\*2  
del s[0]  
print(s)

Перевірка на вхід різних варіантів даних:







Алгоритм виконання програми:

1. Програма просить на вхід натуральне число N>100. Якщо користувач уводить число, що не відповідає цій умові, програма просить, щоб користувач спробував увести його ще раз.
2. Програма формує послідовність натуральних чисел i=10,11,12,…,N+1, попередньо створив діапазон цих чисел. Варто зазначити, що за умовою задачі, треба створити діапазон натуральних чисел i=11,12,…,N. Оскільки діапазон завжди викроює останній елемент вказаної умови, треба було вказати саме N+1. Далі за умовою, треба обчислити послідовність за відомою формулою, що передбачає використовування попереднього елементу послідовності. Для цього дано елемент S10 = 10, то ж для виконання наступного кроку алгоритма необхідно було ввести додатковий елемент 10 з 0 індексом.
3. Програма у циклі обчислює кожен елемент послідовності за формулою, що дана в умові. Оскільки програма передбачає користування попердніми елементами списку, то ж варто почати з 1 (другого) елемента, щоб уникнути повідомлення про помилку, попередньо ввівши додатковий елемент S10 = 10 (пояснення в попердньому кроці алгоритма).
4. Програма видаляє допоміжний елемент S10 = 10, щоб кінцева відповідь відповідала умові задачі, та виводитть на друк всі інші елементи послідовності.

**Аналіз проблем, які виникли при виконанні лабораторної роботи:**

Проблем не виникло. Складність була в структуруванні коду та приведенні його до гарнішого вигляду.