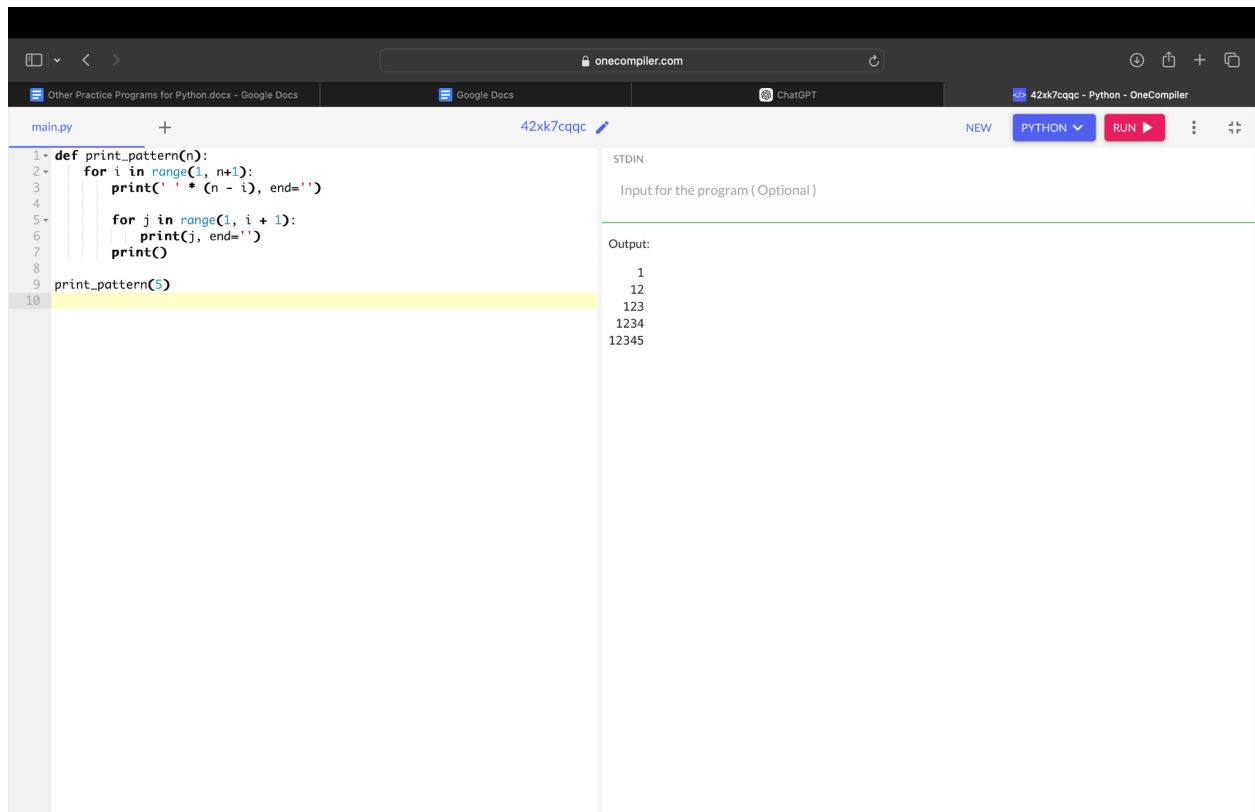


11. Write a program to print the following pattern.

```
1
12
123
1234
12345
```



The screenshot shows a web browser window with the URL `onecompiler.com`. The browser's address bar and tabs are visible at the top. The main content area is a Python IDE with a file named `main.py`. The code in the editor is as follows:

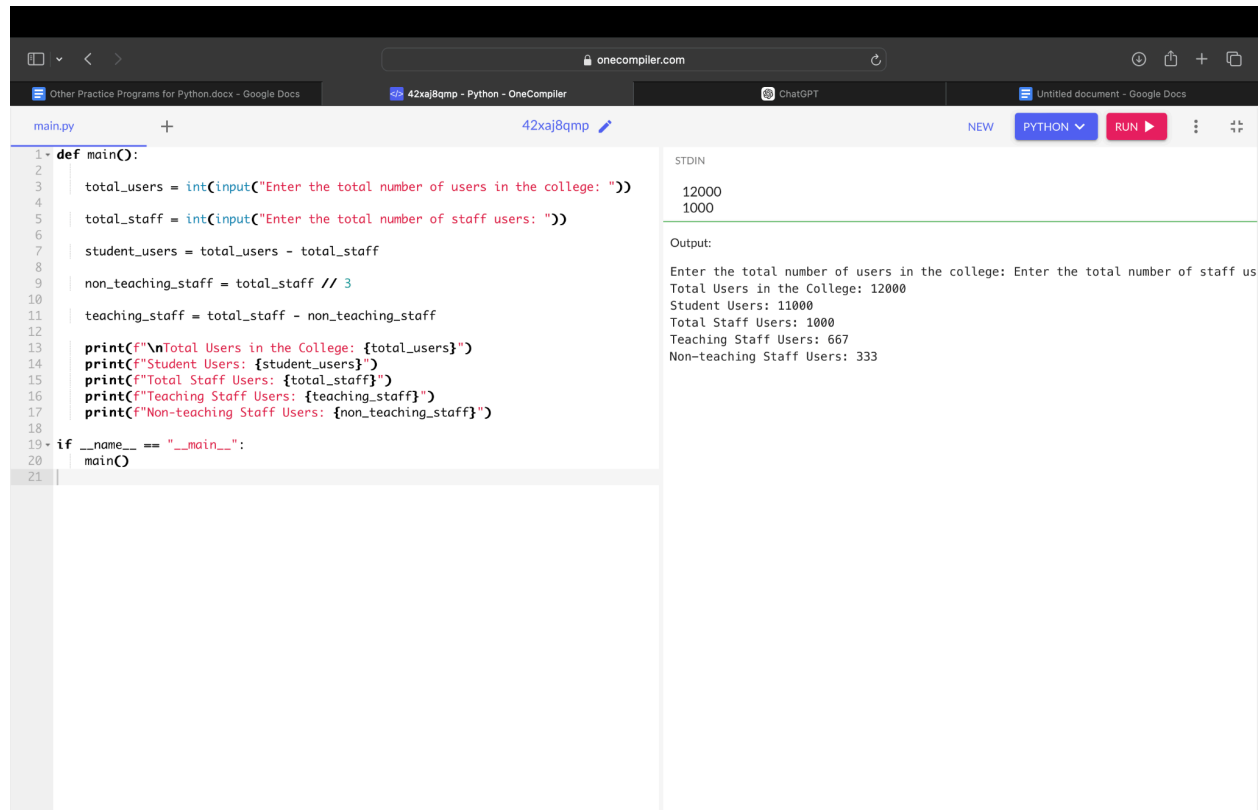
```
1 def print_pattern(n):
2     for i in range(1, n+1):
3         print(' ' * (n - i), end='')
4
5         for j in range(1, i + 1):
6             print(j, end='')
7         print()
8
9 print_pattern(5)
```

The line `print_pattern(5)` is highlighted in yellow. To the right of the code editor is a panel with the following sections:

- STDIN**: A text input field with the placeholder text "Input for the program (Optional)".
- Output:** A text area displaying the output of the program, which matches the pattern shown in the problem statement:

```
1
12
123
1234
12345
```

12. Write a program to find the number of student users in the college, get the total users, staff users details from the client. Note for every 3 staff user there is one Non teaching staff user assigned by default.



The screenshot shows a web-based Python IDE. The editor on the left contains a Python script named `main.py`. The script prompts the user for the total number of users and staff, then calculates the number of student users, teaching staff, and non-teaching staff based on the given constraints. The output on the right shows the results of the program execution.

```
1 def main():
2     total_users = int(input("Enter the total number of users in the college: "))
3
4     total_staff = int(input("Enter the total number of staff users: "))
5
6     student_users = total_users - total_staff
7
8     non_teaching_staff = total_staff // 3
9
10    teaching_staff = total_staff - non_teaching_staff
11
12
13    print(f"\nTotal Users in the College: {total_users}")
14    print(f"Student Users: {student_users}")
15    print(f"Total Staff Users: {total_staff}")
16    print(f"Teaching Staff Users: {teaching_staff}")
17    print(f"Non-teaching Staff Users: {non_teaching_staff}")
18
19 if __name__ == "__main__":
20     main()
21
```

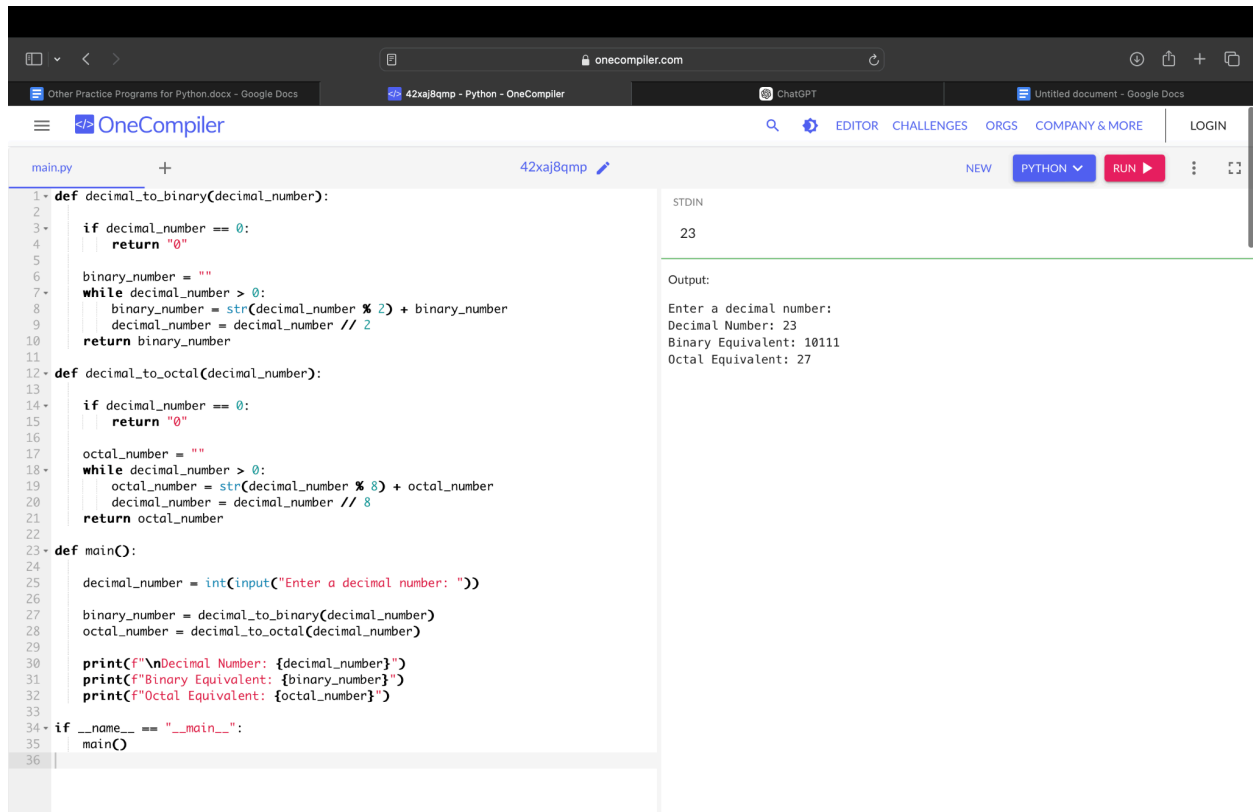
STDIN

```
12000
1000
```

Output:

```
Enter the total number of users in the college: Enter the total number of staff us
Total Users in the College: 12000
Student Users: 11000
Total Staff Users: 1000
Teaching Staff Users: 667
Non-teaching Staff Users: 333
```

13. Write a program to convert Decimal number equivalent to Binary number and octal numbers?



```
1 def decimal_to_binary(decimal_number):
2
3     if decimal_number == 0:
4         return "0"
5
6     binary_number = ""
7     while decimal_number > 0:
8         binary_number = str(decimal_number % 2) + binary_number
9         decimal_number = decimal_number // 2
10    return binary_number
11
12 def decimal_to_octal(decimal_number):
13
14     if decimal_number == 0:
15         return "0"
16
17     octal_number = ""
18     while decimal_number > 0:
19         octal_number = str(decimal_number % 8) + octal_number
20         decimal_number = decimal_number // 8
21     return octal_number
22
23 def main():
24
25     decimal_number = int(input("Enter a decimal number: "))
26
27     binary_number = decimal_to_binary(decimal_number)
28     octal_number = decimal_to_octal(decimal_number)
29
30     print(f"\nDecimal Number: {decimal_number}")
31     print(f"Binary Equivalent: {binary_number}")
32     print(f"Octal Equivalent: {octal_number}")
33
34 if __name__ == "__main__":
35     main()
36
```

STDIN

23

Output:

Enter a decimal number:
Decimal Number: 23
Binary Equivalent: 10111
Octal Equivalent: 27

14. Write a program to print the below pattern?

```

          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1  1   4   6   4   1
```

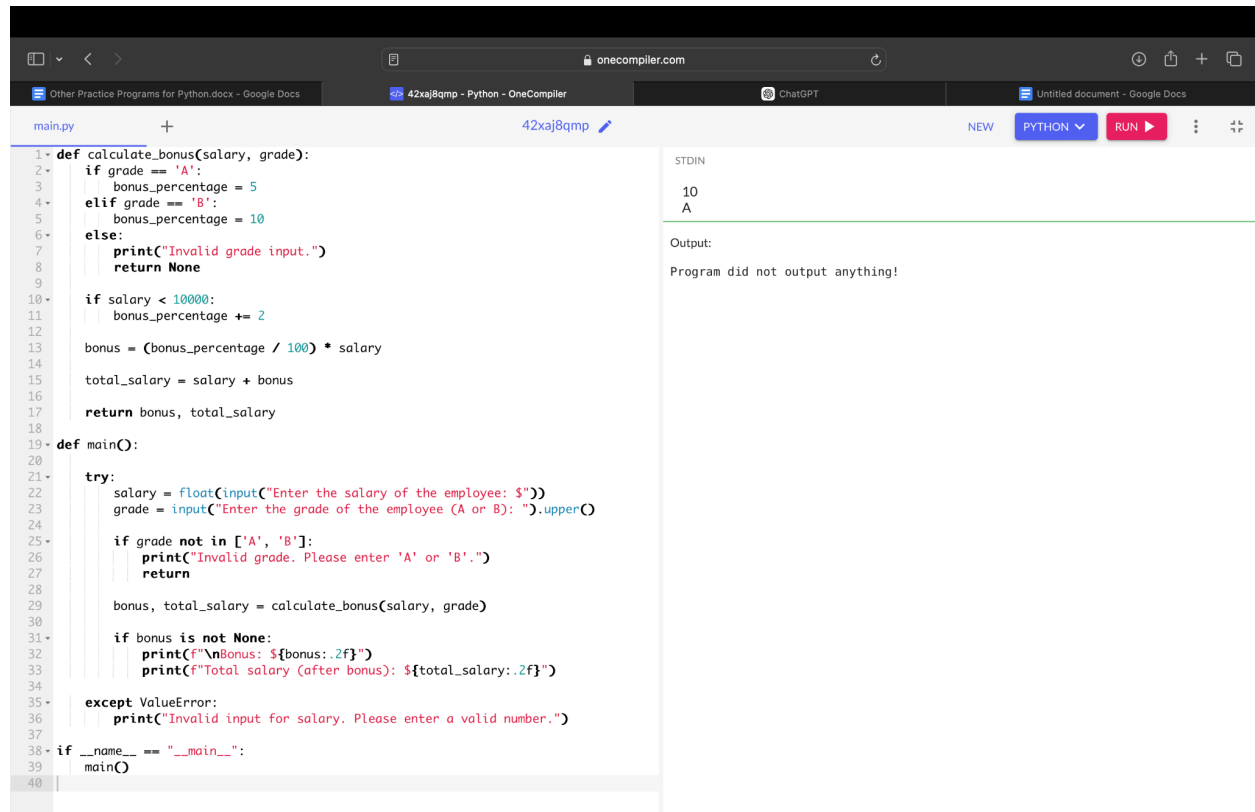
The screenshot shows a web-based Python IDE with a browser window at the top displaying 'onecompiler.com'. Below the browser, there are tabs for 'main.py', '42xk7cqqc', and 'NEW'. The 'main.py' tab is active, showing the following Python code:

```
1 def print_pascals_triangle(n):
2     triangle = []
3
4     for i in range(n):
5         row = [1] * (i + 1)
6
7         for j in range(1, i):
8             row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j]
9
10        triangle.append(row)
11
12    for row in triangle:
13        print(" ".join(map(str, row)).center(n * 2))
14
15 print_pascals_triangle(5)
```

The code is executed, and the output is displayed in the 'Output' section on the right. The output is a Pascal's triangle with 5 rows, centered and formatted with spaces:

```
Output:
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

15. In an organization they decide to give bonus to all the employees on New Year. A 5% bonus on salary is given to the grade A workers and 10% bonus on salary to the grade B workers. Write a program to enter the salary and grade of the employee. If the salary of the employee is less than \$10,000 then the employee gets an extra 2% bonus on salary Calculate the bonus that has to be given to the employee and print the salary that the employee will get. in python program



The screenshot shows a web-based Python IDE interface. The left pane contains a Python script named `main.py` with the following code:

```
1 def calculate_bonus(salary, grade):
2     if grade == 'A':
3         bonus_percentage = 5
4     elif grade == 'B':
5         bonus_percentage = 10
6     else:
7         print("Invalid grade input.")
8         return None
9
10    if salary < 10000:
11        bonus_percentage += 2
12
13    bonus = (bonus_percentage / 100) * salary
14
15    total_salary = salary + bonus
16
17    return bonus, total_salary
18
19 def main():
20
21     try:
22         salary = float(input("Enter the salary of the employee: $"))
23         grade = input("Enter the grade of the employee (A or B): ").upper()
24
25         if grade not in ['A', 'B']:
26             print("Invalid grade. Please enter 'A' or 'B'.")
27             return
28
29         bonus, total_salary = calculate_bonus(salary, grade)
30
31         if bonus is not None:
32             print(f"\nBonus: ${bonus:.2f}")
33             print(f"Total salary (after bonus): ${total_salary:.2f}")
34
35     except ValueError:
36         print("Invalid input for salary. Please enter a valid number.")
37
38 if __name__ == "__main__":
39     main()
40
```

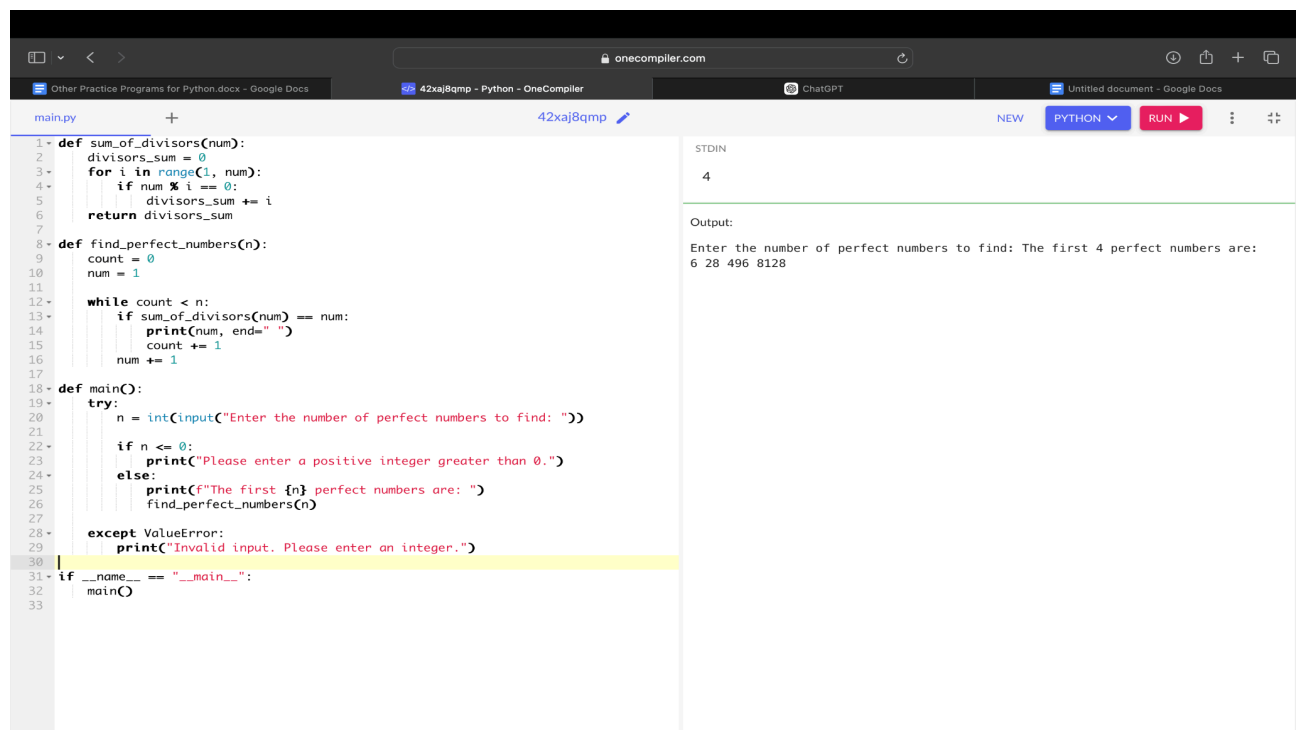
The right pane shows the execution environment. The **STDIN** input is:

```
10
A
```

The **Output:** section displays the message:

```
Program did not output anything!
```

16. Write a program to print the first n perfect numbers. (Hint Perfect number means a positive integer that is equal to the sum of its proper divisors)



```
1- def sum_of_divisors(num):
2-     divisors_sum = 0
3-     for i in range(1, num):
4-         if num % i == 0:
5-             divisors_sum += i
6-     return divisors_sum
7-
8- def find_perfect_numbers(n):
9-     count = 0
10-    num = 1
11-
12-    while count < n:
13-        if sum_of_divisors(num) == num:
14-            print(num, end=" ")
15-            count += 1
16-        num += 1
17-
18- def main():
19-     try:
20-         n = int(input("Enter the number of perfect numbers to find: "))
21-
22-         if n <= 0:
23-             print("Please enter a positive integer greater than 0.")
24-         else:
25-             print(f"The first {n} perfect numbers are: ")
26-             find_perfect_numbers(n)
27-
28-     except ValueError:
29-         print("Invalid input. Please enter an integer.")
30-
31- if __name__ == "__main__":
32-     main()
33-
```

STDIN

4

Output:

Enter the number of perfect numbers to find: The first 4 perfect numbers are:
6 28 496 8128

17. A Pythagorean triplet is a set of three integers a, b and c such that $a^2 + b^2 = c^2$. Given a limit, generate all Pythagorean Triples with values smaller than given limit?

The screenshot shows a web-based Python IDE with the following code in `main.py`:

```

1- def generate_pythagorean_triplets(limit):
2-     triplets = []
3-
4-     for c in range(1, limit):
5-         for a in range(1, c):
6-             for b in range(a, c):
7-                 if a*a + b*b == c*c:
8-                     triplets.append((a, b, c))
9-
10-    return triplets
11-
12- def main():
13-     try:
14-         limit = int(input("Enter the limit: "))
15-
16-         if limit <= 0:
17-             print("Please enter a positive integer greater than 0.")
18-         else:
19-             print(f"Pythagorean Triplets with values smaller than {limit}:")
20-             triplets = generate_pythagorean_triplets(limit)
21-
22-             if not triplets:
23-                 print("No Pythagorean triplets found.")
24-             else:
25-                 for triplet in triplets:
26-                     print(triplet)
27-
28-     except ValueError:
29-         print("Invalid input. Please enter a valid integer.")
30-
31- if __name__ == "__main__":
32-     main()
33-

```

The right-hand side of the IDE shows the execution results:

STDIN: 50

Output:

```

Enter the limit: Pythagorean Triplets with values smaller than 50:
(3, 4, 5)
(6, 8, 10)
(5, 12, 13)
(9, 12, 15)
(8, 15, 17)
(12, 16, 20)
(7, 24, 25)
(15, 20, 25)
(10, 24, 26)
(20, 21, 29)
(18, 24, 30)
(16, 30, 34)
(21, 28, 35)
(12, 35, 37)
(15, 36, 39)
(24, 32, 40)
(9, 40, 41)
(27, 36, 45)

```

18. Write a program using function to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered 12 percent rate of interest; for all other customers, the ROI is 10 percent.

The screenshot shows a web-based Python IDE with the following code in `main.py`:

```

1- def calculate_simple_interest(principal, time, is_senior_citizen):
2-     rate_of_interest = 12 if is_senior_citizen else 10
3-     return (principal * time * rate_of_interest) / 100
4-
5- principal = float(input("Enter principal: $"))
6- time = float(input("Enter time (in years): "))
7- is_senior_citizen = input("Is the customer a senior citizen? (yes/no): ").lower() == 'yes'
8-
9- interest = calculate_simple_interest(principal, time, is_senior_citizen)
10- print(f"Simple Interest: ${interest:.2f}")
11-

```

The right-hand side of the IDE shows the execution results:

STDIN: 5.10, 6.5, no

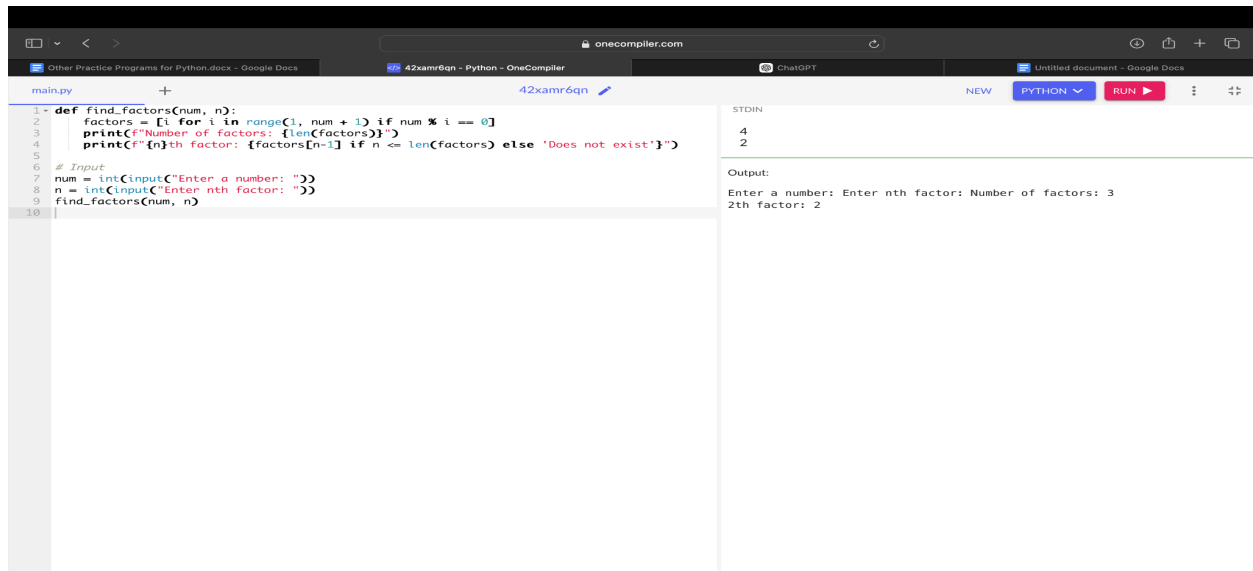
Output:

```

years): Is the customer a senior citizen? (yes/no): Simple Interest: $3.31

```

19. Write a program to print number of factors and to print nth factor of the given number.

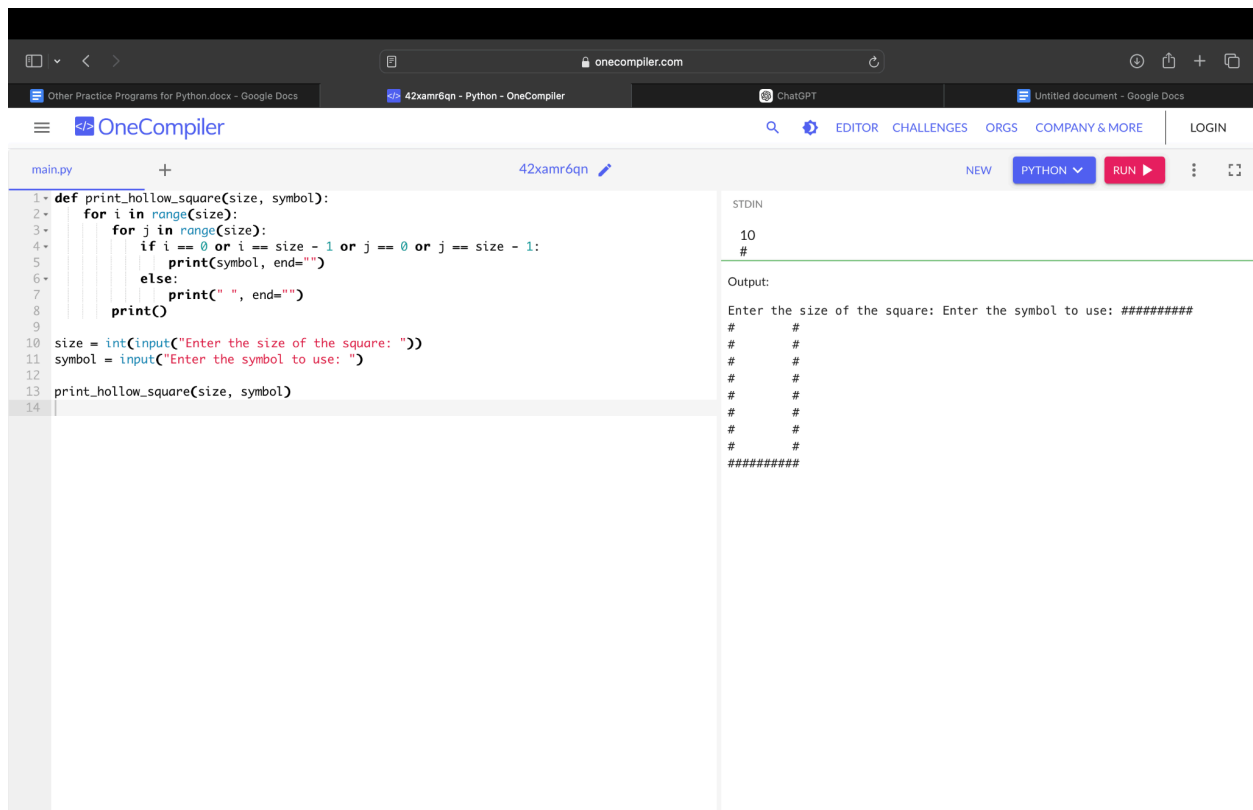


The screenshot shows a web browser with the OneCompiler website. The editor has a file named 'main.py' with the following Python code:

```
1 def find_factors(num, n):
2     factors = [i for i in range(1, num + 1) if num % i == 0]
3     print(f"Number of factors: {len(factors)}")
4     print(f"{n}th factor: {factors[n-1] if n <= len(factors) else 'Does not exist'}")
5
6 # Input
7 num = int(input("Enter a number: "))
8 n = int(input("Enter nth factor: "))
9 find_factors(num, n)
10
```

The right-hand side of the interface shows the 'STDIN' and 'Output' sections. The 'STDIN' section contains the input values '4' and '2'. The 'Output' section shows the program's execution results: 'Enter a number: Enter nth factor: Number of factors: 3' and '2th factor: 2'.

20. Write a program to print hollow square symbol pattern? Get the symbol input from the user



The screenshot shows the OneCompiler website with a Python program to print a hollow square symbol pattern. The code in 'main.py' is as follows:

```
1 def print_hollow_square(size, symbol):
2     for i in range(size):
3         for j in range(size):
4             if i == 0 or i == size - 1 or j == 0 or j == size - 1:
5                 print(symbol, end=" ")
6             else:
7                 print(" ", end=" ")
8         print()
9
10 size = int(input("Enter the size of the square: "))
11 symbol = input("Enter the symbol to use: ")
12
13 print_hollow_square(size, symbol)
14
```

The 'STDIN' section shows the inputs '10' and '#'. The 'Output' section displays the resulting hollow square pattern using the '#' symbol, with a size of 10. The pattern consists of 10 rows and 10 columns, with the border characters being '#' and the interior characters being spaces.