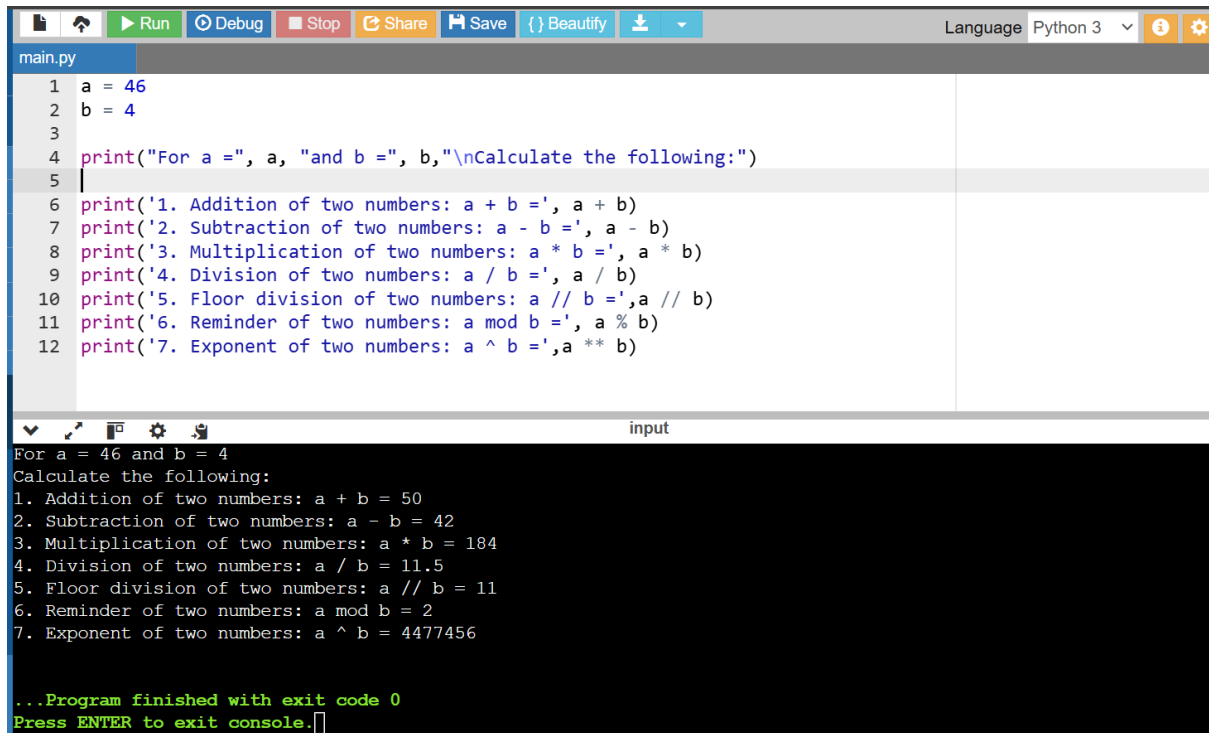


Python Operators

1. Arithmetic Operators:



The screenshot shows a Python IDE with a file named 'main.py'. The code defines two variables, 'a' and 'b', with values 46 and 4 respectively. It then prints a series of arithmetic operations and their results. The output window shows the results of these operations.

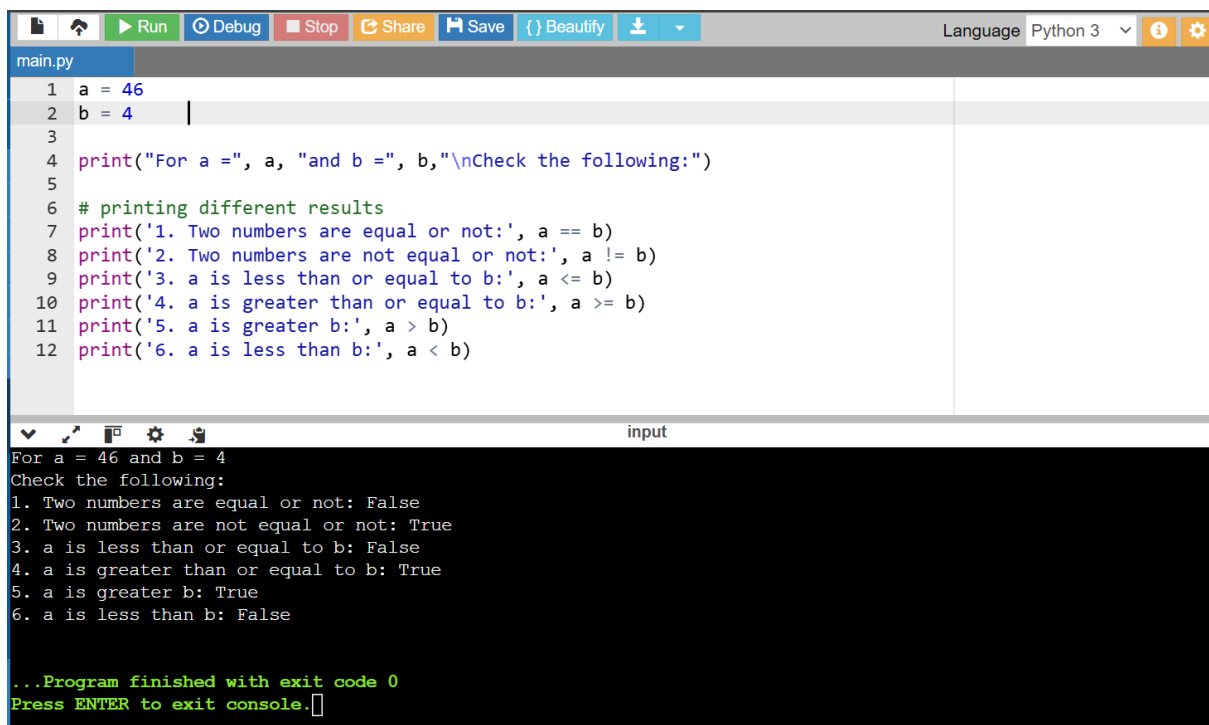
```
1 a = 46
2 b = 4
3
4 print("For a =", a, "and b =", b, "\nCalculate the following:")
5
6 print('1. Addition of two numbers: a + b =', a + b)
7 print('2. Subtraction of two numbers: a - b =', a - b)
8 print('3. Multiplication of two numbers: a * b =', a * b)
9 print('4. Division of two numbers: a / b =', a / b)
10 print('5. Floor division of two numbers: a // b =', a // b)
11 print('6. Remainder of two numbers: a mod b =', a % b)
12 print('7. Exponent of two numbers: a ^ b =', a ** b)
```

input

For a = 46 and b = 4
Calculate the following:
1. Addition of two numbers: a + b = 50
2. Subtraction of two numbers: a - b = 42
3. Multiplication of two numbers: a * b = 184
4. Division of two numbers: a / b = 11.5
5. Floor division of two numbers: a // b = 11
6. Remainder of two numbers: a mod b = 2
7. Exponent of two numbers: a ^ b = 4477456

...Program finished with exit code 0
Press ENTER to exit console.

2. Comparison Operators:



The screenshot shows a Python IDE with a file named 'main.py'. The code defines two variables, 'a' and 'b', with values 46 and 4 respectively. It then prints a series of comparison operations and their results. The output window shows the results of these operations.

```
1 a = 46
2 b = 4
3
4 print("For a =", a, "and b =", b, "\nCheck the following:")
5
6 # printing different results
7 print('1. Two numbers are equal or not:', a == b)
8 print('2. Two numbers are not equal or not:', a != b)
9 print('3. a is less than or equal to b:', a <= b)
10 print('4. a is greater than or equal to b:', a >= b)
11 print('5. a is greater b:', a > b)
12 print('6. a is less than b:', a < b)
```

input

For a = 46 and b = 4
Check the following:
1. Two numbers are equal or not: False
2. Two numbers are not equal or not: True
3. a is less than or equal to b: False
4. a is greater than or equal to b: True
5. a is greater b: True
6. a is less than b: False

...Program finished with exit code 0
Press ENTER to exit console.

3. Assignment Operators:



The screenshot shows a Python IDE with a file named 'main.py'. The code in the editor uses assignment operators to modify a variable 'a' starting from 34. The operations performed are addition, subtraction, multiplication, division, modulus, and exponentiation, each followed by a print statement. The output console shows the results of these operations: 40, 28, 204, 5.666666666666667, 4, and 1544804416. The program ends with a message indicating it finished with exit code 0.

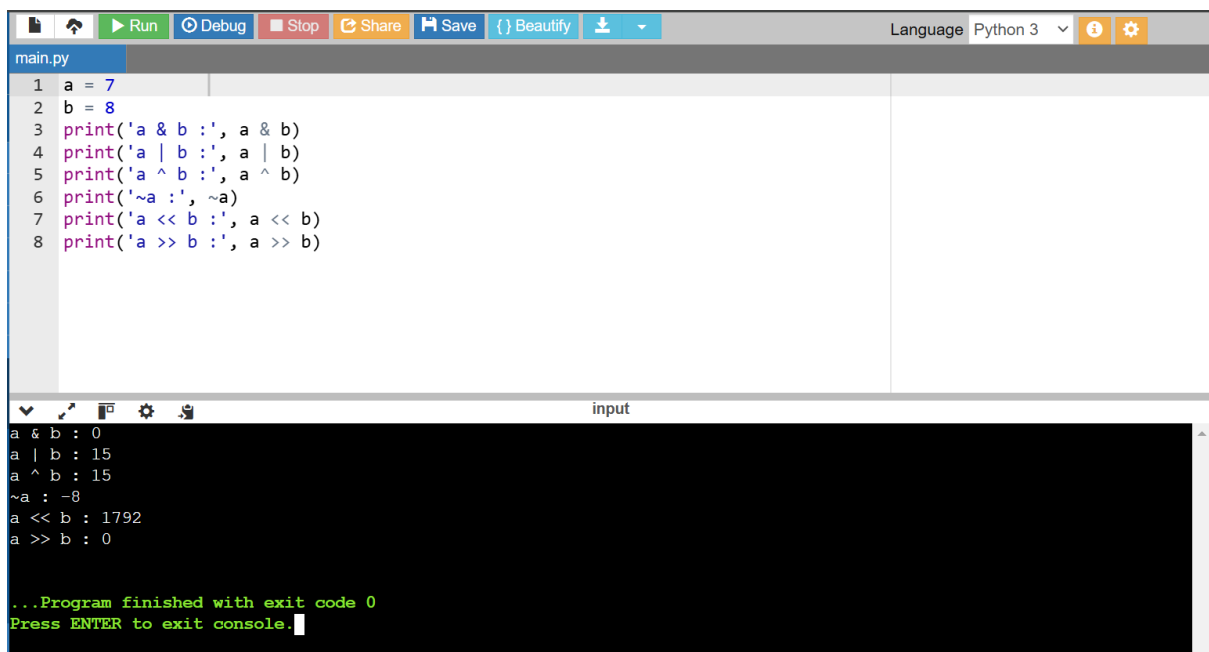
```
1 a = 34
2 b = 6
3 print('a += b:', a + b)
4 print('a -= b:', a - b)
5 print('a *= b:', a * b)
6 print('a /= b:', a / b)
7 print('a %= b:', a % b)
8 print('a **= b:', a ** b)
9 print('a //= b:', a // b)
```

input

```
a += b: 40
a -= b: 28
a *= b: 204
a /= b: 5.666666666666667
a %= b: 4
a **= b: 1544804416
a //= b: 5

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Bitwise Operators:



The screenshot shows a Python IDE with a file named 'main.py'. The code in the editor uses bitwise operators (&, |, ^, ~, <<, >>) on the numbers 7 and 8. The operations performed are bitwise AND, OR, XOR, NOT, left shift, and right shift, each followed by a print statement. The output console shows the results of these operations: 0, 15, 15, -8, 1792, and 0. The program ends with a message indicating it finished with exit code 0.

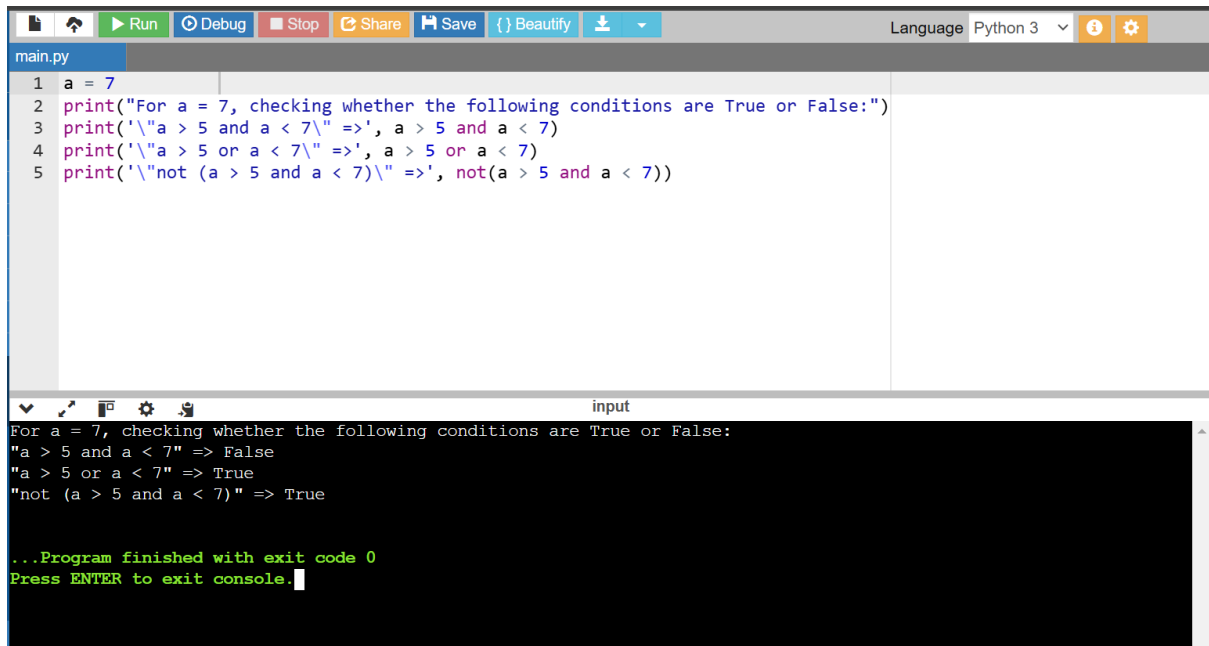
```
1 a = 7
2 b = 8
3 print('a & b:', a & b)
4 print('a | b:', a | b)
5 print('a ^ b:', a ^ b)
6 print('~a:', ~a)
7 print('a << b:', a << b)
8 print('a >> b:', a >> b)
```

input

```
a & b: 0
a | b: 15
a ^ b: 15
~a: -8
a << b: 1792
a >> b: 0

...Program finished with exit code 0
Press ENTER to exit console.
```

5. Logical Operators:



The screenshot shows a Python IDE with a file named `main.py`. The code defines a variable `a = 7` and prints several logical expressions. The output window shows the results of these expressions.

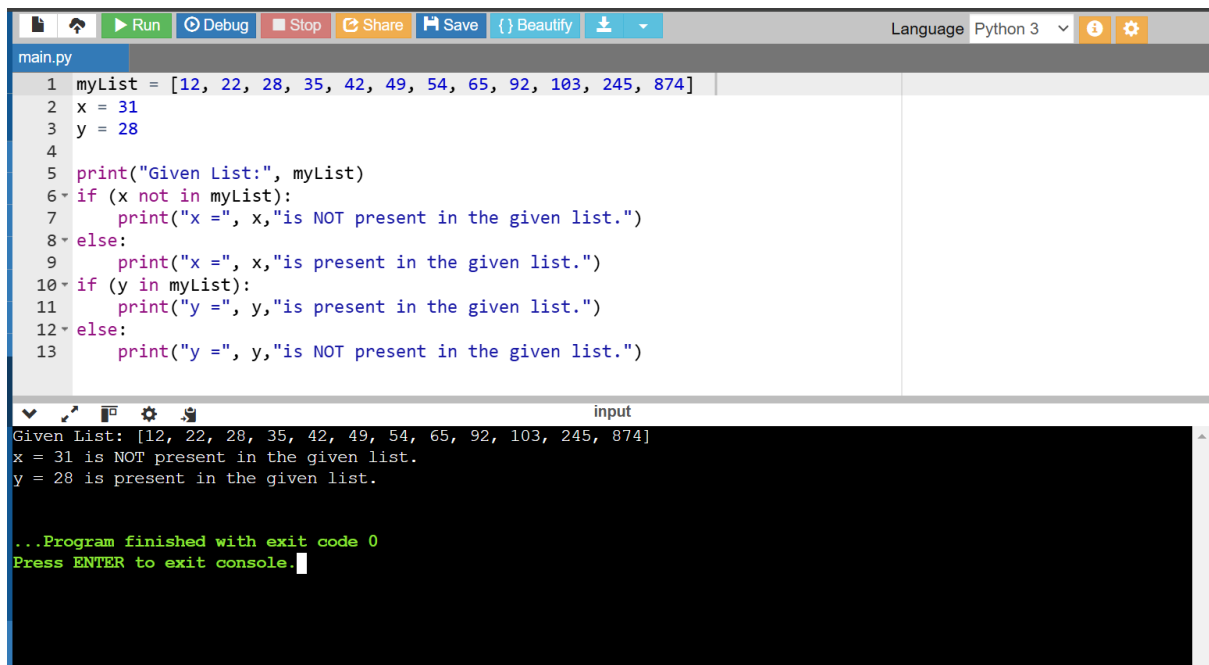
```
1 a = 7
2 print("For a = 7, checking whether the following conditions are True or False:")
3 print('\na > 5 and a < 7" =>', a > 5 and a < 7)
4 print('\na > 5 or a < 7" =>', a > 5 or a < 7)
5 print('\not (a > 5 and a < 7)" =>', not(a > 5 and a < 7))
```

input

```
For a = 7, checking whether the following conditions are True or False:
"a > 5 and a < 7" => False
"a > 5 or a < 7" => True
"not (a > 5 and a < 7)" => True

...Program finished with exit code 0
Press ENTER to exit console.
```

6. Membership Operators:



The screenshot shows a Python IDE with a file named `main.py`. The code creates a list `myList` and checks if variables `x` and `y` are members of it using the `in` and `not in` operators. The output window shows the results of these checks.

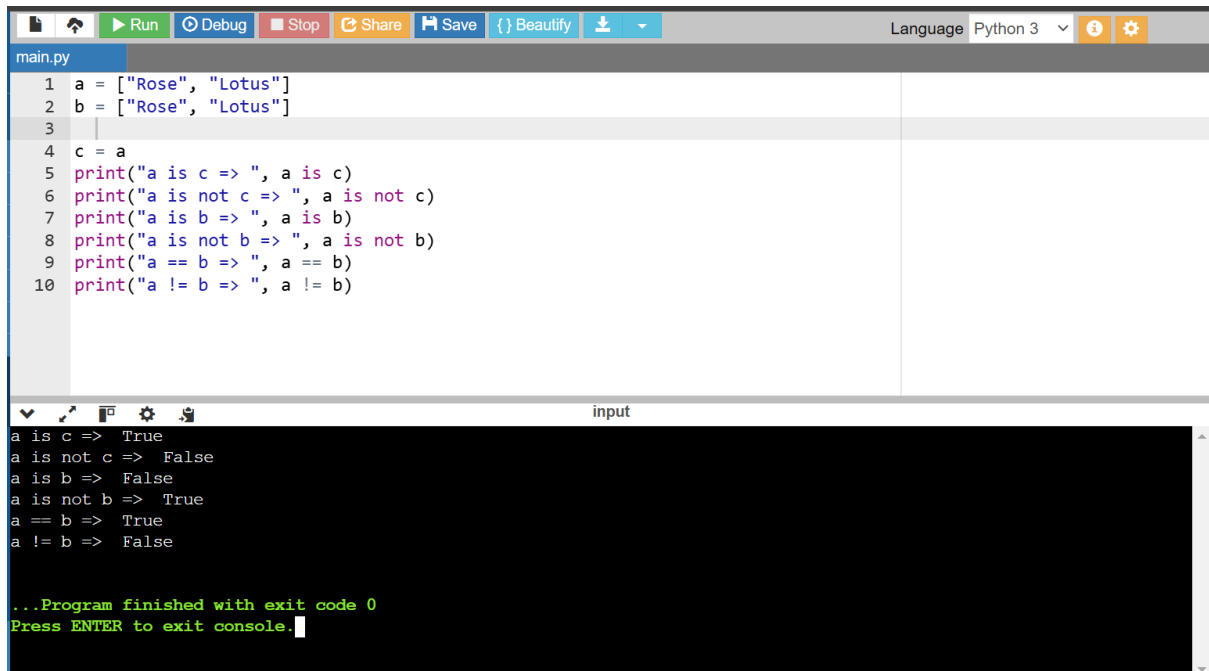
```
1 myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
2 x = 31
3 y = 28
4
5 print("Given List:", myList)
6 if (x not in myList):
7     print("x =", x, "is NOT present in the given list.")
8 else:
9     print("x =", x, "is present in the given list.")
10 if (y in myList):
11     print("y =", y, "is present in the given list.")
12 else:
13     print("y =", y, "is NOT present in the given list.")
```

input

```
Given List: [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
x = 31 is NOT present in the given list.
y = 28 is present in the given list.

...Program finished with exit code 0
Press ENTER to exit console.
```

7. Identity Operators:



The screenshot shows a Python IDE with a file named `main.py`. The code defines two lists, `a` and `b`, both containing the strings "Rose" and "Lotus". It then performs several identity and equality checks using `is`, `is not`, `==`, and `!=` operators. The output console shows the results of these checks.

```
1 a = ["Rose", "Lotus"]
2 b = ["Rose", "Lotus"]
3
4 c = a
5 print("a is c => ", a is c)
6 print("a is not c => ", a is not c)
7 print("a is b => ", a is b)
8 print("a is not b => ", a is not b)
9 print("a == b => ", a == b)
10 print("a != b => ", a != b)
```

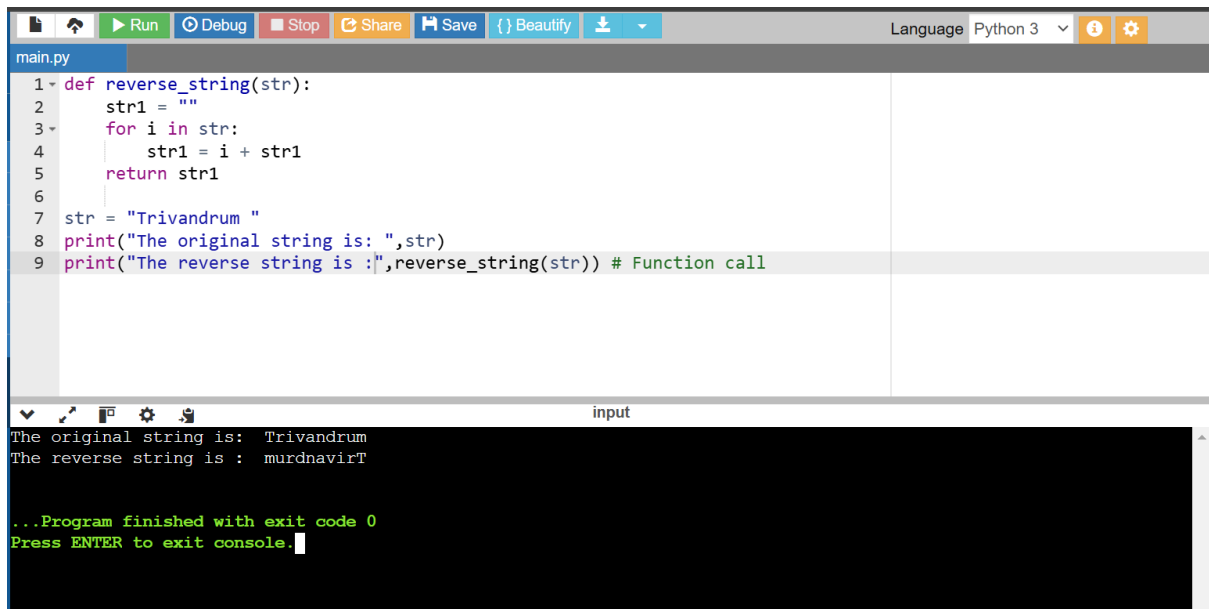
input

```
a is c => True
a is not c => False
a is b => False
a is not b => True
a == b => True
a != b => False

...Program finished with exit code 0
Press ENTER to exit console.
```

REVERSE A STRING IN PYTHON:

1. Using for loop:



The screenshot shows a Python IDE with a file named `main.py`. It defines a function `reverse_string` that iterates through each character of a string and builds a reversed string. The function is then called with the string "Trivandrum". The output console shows the original and reversed strings.

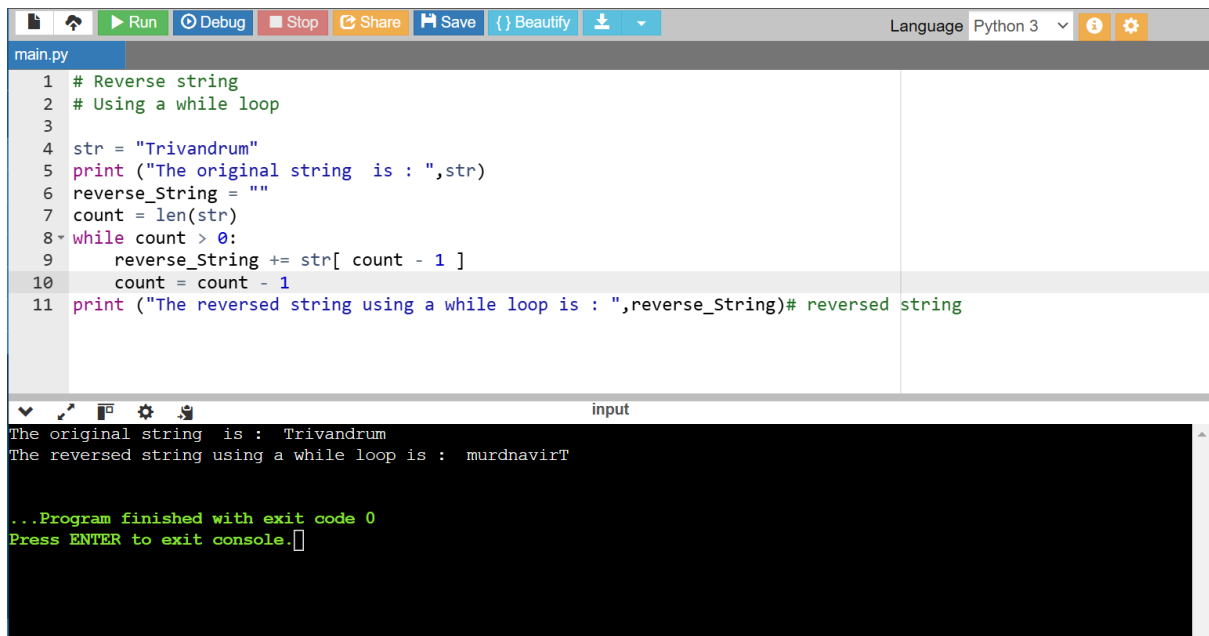
```
1 def reverse_string(str):
2     str1 = ""
3     for i in str:
4         str1 = i + str1
5     return str1
6
7 str = "Trivandrum "
8 print("The original string is: ",str)
9 print("The reverse string is :",reverse_string(str)) # Function call
```

input

```
The original string is: Trivandrum
The reverse string is : murdnairvT

...Program finished with exit code 0
Press ENTER to exit console.
```

2. Using while loop:



The screenshot shows a Python IDE with a file named 'main.py'. The code uses a while loop to reverse the string 'Trivandrum'. The console output shows the original string, the reversed string 'murdnaviT', and a message indicating the program finished with exit code 0.

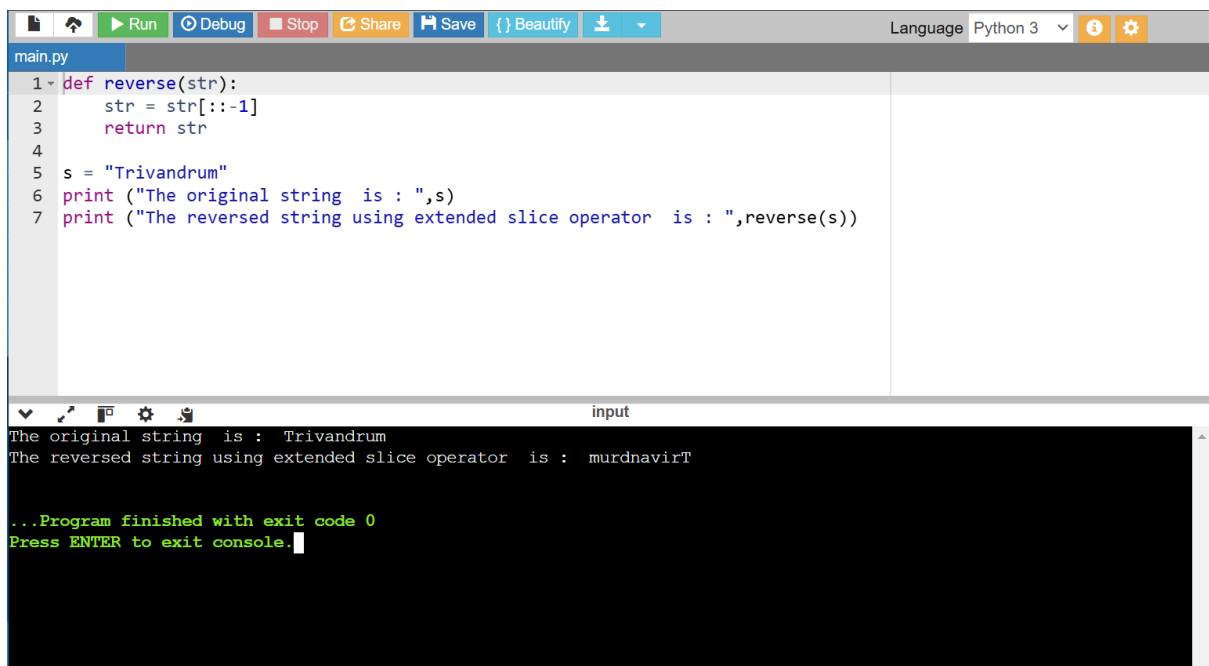
```
1 # Reverse string
2 # Using a while loop
3
4 str = "Trivandrum"
5 print ("The original string is : ",str)
6 reverse_String = ""
7 count = len(str)
8 while count > 0:
9     reverse_String += str[ count - 1 ]
10    count = count - 1
11 print ("The reversed string using a while loop is : ",reverse_String)# reversed string
```

input

```
The original string is : Trivandrum
The reversed string using a while loop is : murdnaviT

...Program finished with exit code 0
Press ENTER to exit console.
```

3. Using the slice ([]) operator:



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a function 'reverse' that uses the slice operator '::-1' to reverse a string. The console output shows the original string, the reversed string 'murdnaviT', and a message indicating the program finished with exit code 0.


```
1 def reverse(str):
2     str = str[::-1]
3     return str
4
5 s = "Trivandrum"
6 print ("The original string is : ",s)
7 print ("The reversed string using extended slice operator is : ",reverse(s))
```

input

```
The original string is : Trivandrum
The reversed string using extended slice operator is : murdnaviT

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Using reverse function with join:

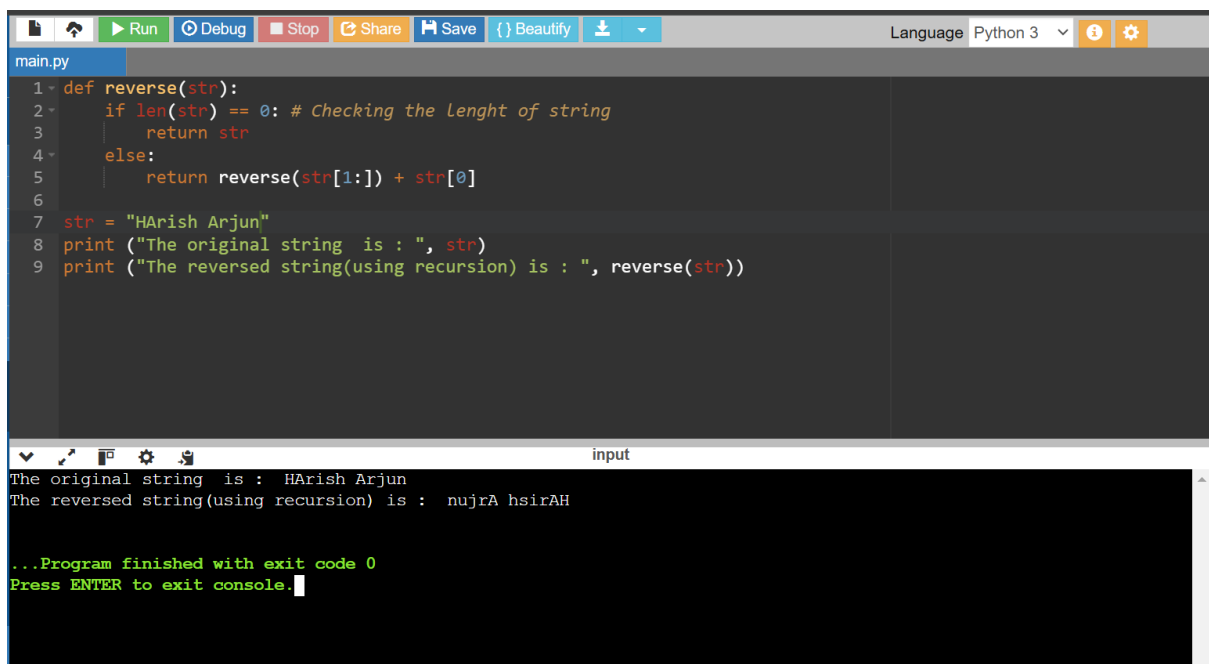


```
main.py
1 def reverse(str):
2     string = "".join(reversed(str)) # reversed() function inside the join() function
3     return string
4
5 s = "Trivandrum"
6
7 print ("The original string is : ",s)
8 print ("The reversed string using reversed() is : ",reverse(s) )

input
The original string is : Trivandrum
The reversed string using reversed() is : murdnavrT

...Program finished with exit code 0
Press ENTER to exit console.
```

5. Using recursion():

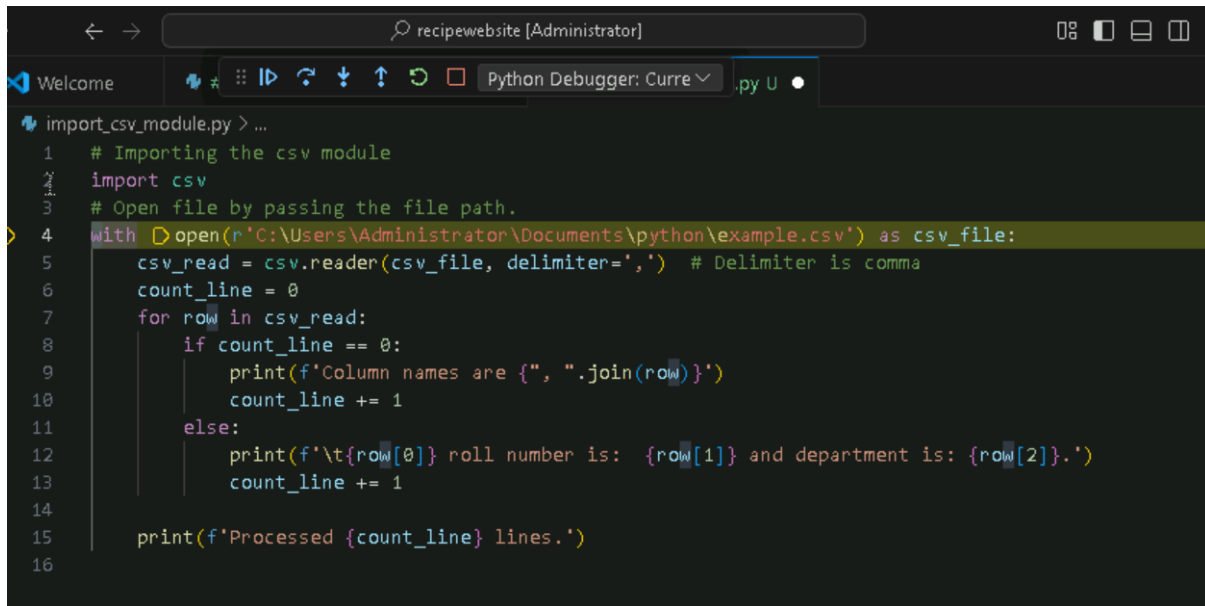


```
main.py
1 def reverse(str):
2     if len(str) == 0: # Checking the length of string
3         return str
4     else:
5         return reverse(str[1:]) + str[0]
6
7 str = "Harish Arjun"
8 print ("The original string is : ", str)
9 print ("The reversed string(using recursion) is : ", reverse(str))

input
The original string is : Harish Arjun
The reversed string(using recursion) is : nujrA hsiRAH

...Program finished with exit code 0
Press ENTER to exit console.
```

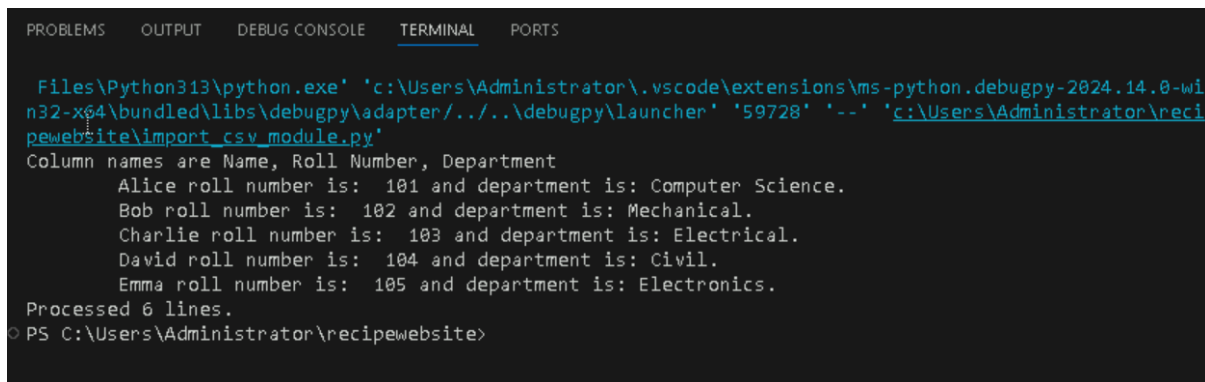
PYTHON CSV FILE :



The screenshot shows a Python script in a VS Code editor. The script is named `import_csv_module.py` and is located in the `recipewebsite` directory. The script imports the `csv` module and opens a file named `example.csv` at the path `r'C:\Users\Administrator\Documents\python\example.csv'`. It then reads the file using `csv.reader` with a comma as the delimiter. The script iterates over each row in the file. For the first row, it prints the column names. For subsequent rows, it prints the roll number and department for each student. Finally, it prints the total number of lines processed.

```
import_csv_module.py > ...
1  # Importing the csv module
2  import csv
3  # Open file by passing the file path.
4  with open(r'C:\Users\Administrator\Documents\python\example.csv') as csv_file:
5      csv_read = csv.reader(csv_file, delimiter=',') # Delimiter is comma
6      count_line = 0
7      for row in csv_read:
8          if count_line == 0:
9              print(f'Column names are {"", ".join(row)}')
10             count_line += 1
11          else:
12              print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}')
13              count_line += 1
14
15  print(f'Processed {count_line} lines.')
16
```

Output:

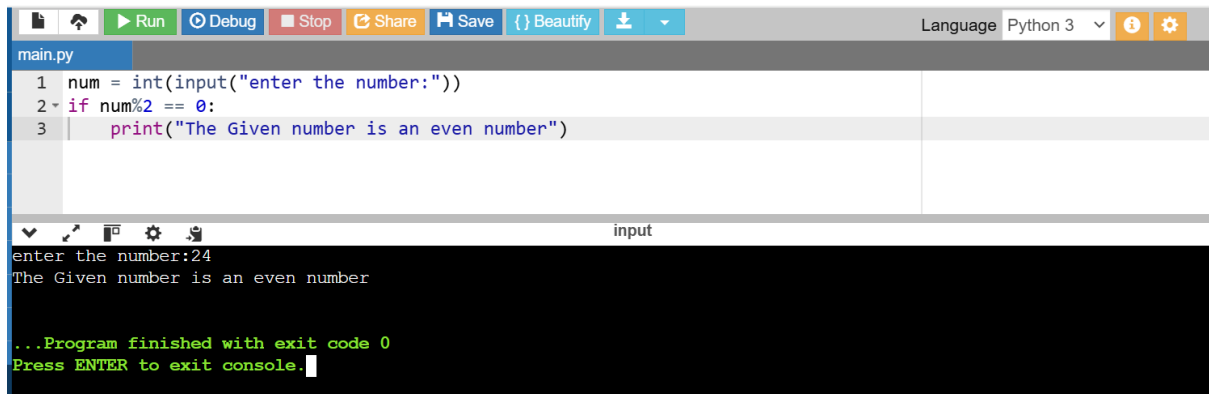


The screenshot shows the terminal output of the Python script. The output displays the column names and the roll number and department for each student. The output is as follows:

```
Files\Python313\python.exe' 'c:\Users\Administrator\.vscode\extensions\ms-python.debugpy-2024.14.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '59728' '--' 'c:\Users\Administrator\recipewebsite\import_csv_module.py'
Column names are Name, Roll Number, Department
Alice roll number is: 101 and department is: Computer Science.
Bob roll number is: 102 and department is: Mechanical.
Charlie roll number is: 103 and department is: Electrical.
David roll number is: 104 and department is: Civil.
Emma roll number is: 105 and department is: Electronics.
Processed 6 lines.
PS C:\Users\Administrator\recipewebsite>
```

Python IF-ELSE:

1. if statement:

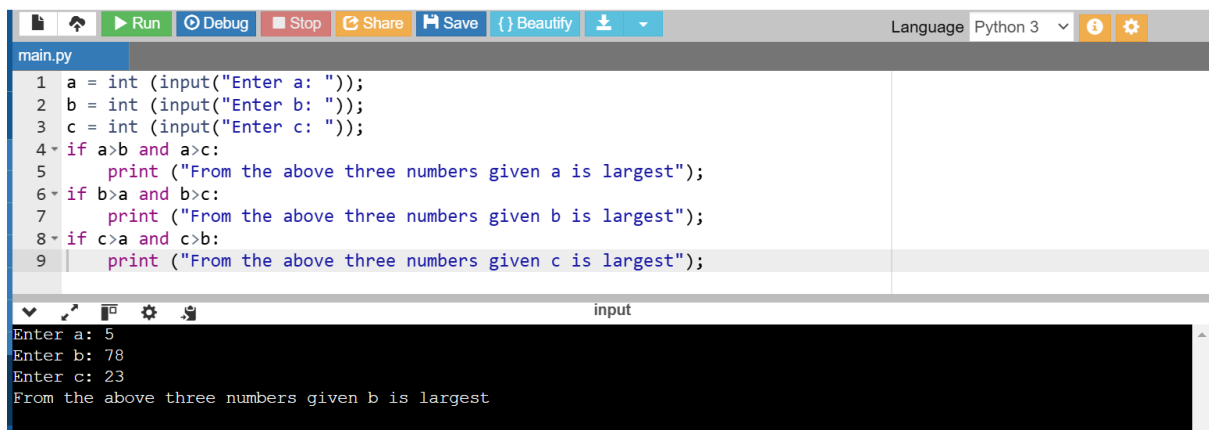


```
main.py
1 num = int(input("enter the number:"))
2 if num%2 == 0:
3     print("The Given number is an even number")

input
enter the number:24
The Given number is an even number

...Program finished with exit code 0
Press ENTER to exit console.
```

2. print the largest of the three numbers



```
main.py
1 a = int (input("Enter a: "));
2 b = int (input("Enter b: "));
3 c = int (input("Enter c: "));
4 if a>b and a>c:
5     print ("From the above three numbers given a is largest");
6 if b>a and b>c:
7     print ("From the above three numbers given b is largest");
8 if c>a and c>b:
9     print ("From the above three numbers given c is largest");

input
Enter a: 5
Enter b: 78
Enter c: 23
From the above three numbers given b is largest
```

If-Else statement:

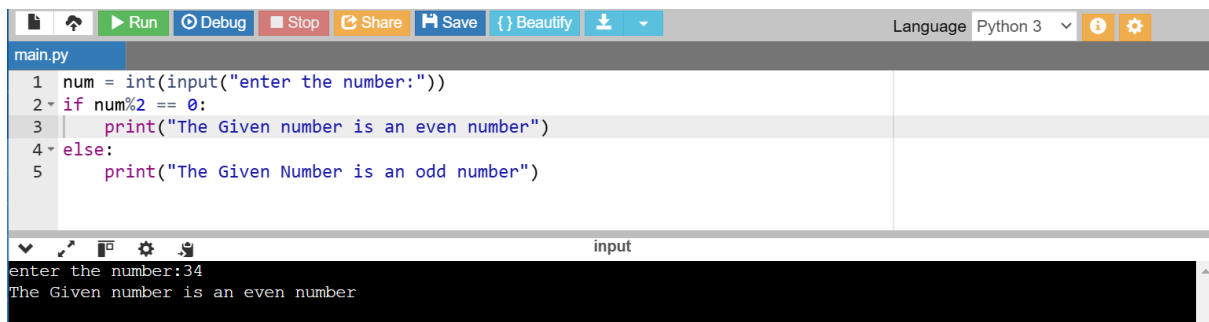
1.



```
main.py
1 age = int (input("Enter your age: "))
2 if age>=18:
3     print("You are eligible to vote !!");
4 else:
5     print("Sorry! you have to wait !!");

input
Enter your age: 76
You are eligible to vote !!
```


2. check whether a number is even or not:



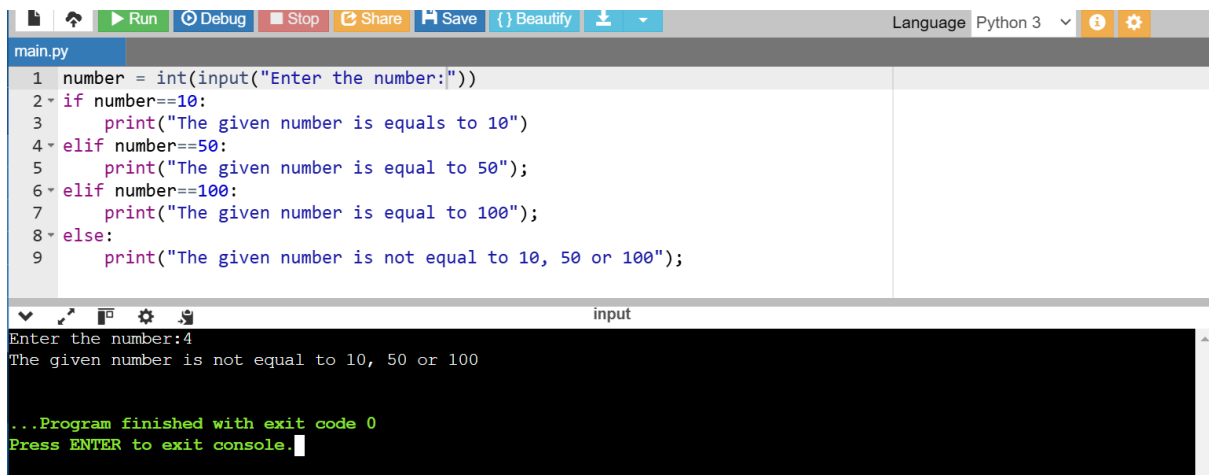
The screenshot shows a Python IDE with a file named 'main.py'. The code is as follows:

```
1 num = int(input("enter the number:"))
2 if num%2 == 0:
3     print("The Given number is an even number")
4 else:
5     print("The Given Number is an odd number")
```

The output console shows the input '34' and the output 'The Given number is an even number'.

Elif statement:

1.

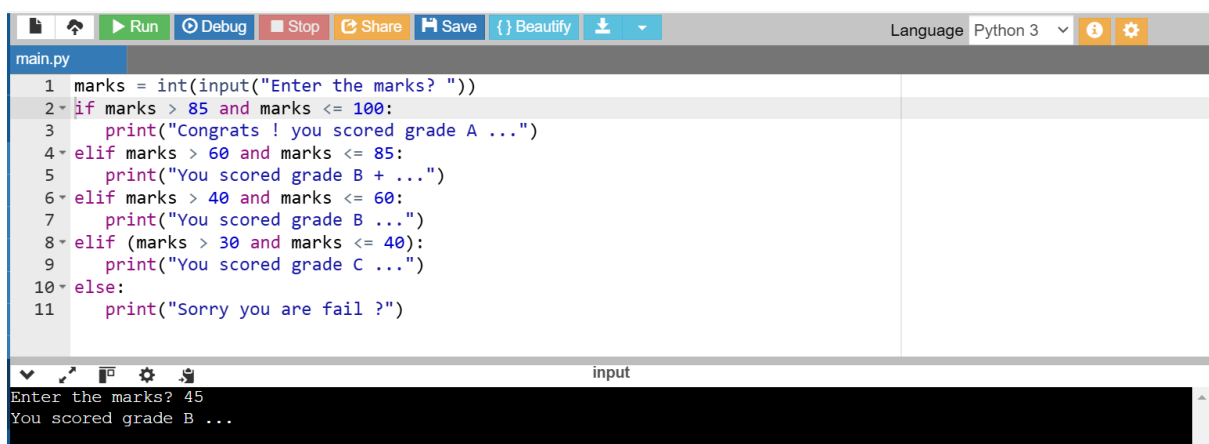


The screenshot shows a Python IDE with a file named 'main.py'. The code is as follows:

```
1 number = int(input("Enter the number:"))
2 if number==10:
3     print("The given number is equals to 10")
4 elif number==50:
5     print("The given number is equal to 50");
6 elif number==100:
7     print("The given number is equal to 100");
8 else:
9     print("The given number is not equal to 10, 50 or 100");
```

The output console shows the input '4' and the output 'The given number is not equal to 10, 50 or 100'. The program also shows a message: '...Program finished with exit code 0 Press ENTER to exit console.'

2.



The screenshot shows a Python IDE with a file named 'main.py'. The code is as follows:

```
1 marks = int(input("Enter the marks? "))
2 if marks > 85 and marks <= 100:
3     print("Congrats ! you scored grade A ...")
4 elif marks > 60 and marks <= 85:
5     print("You scored grade B + ...")
6 elif marks > 40 and marks <= 60:
7     print("You scored grade B ...")
8 elif (marks > 30 and marks <= 40):
9     print("You scored grade C ...")
10 else:
11     print("Sorry you are fail ?")
```

The output console shows the input '45' and the output 'You scored grade B ...'.

WHILE LOOPS:

1. sum of squares:



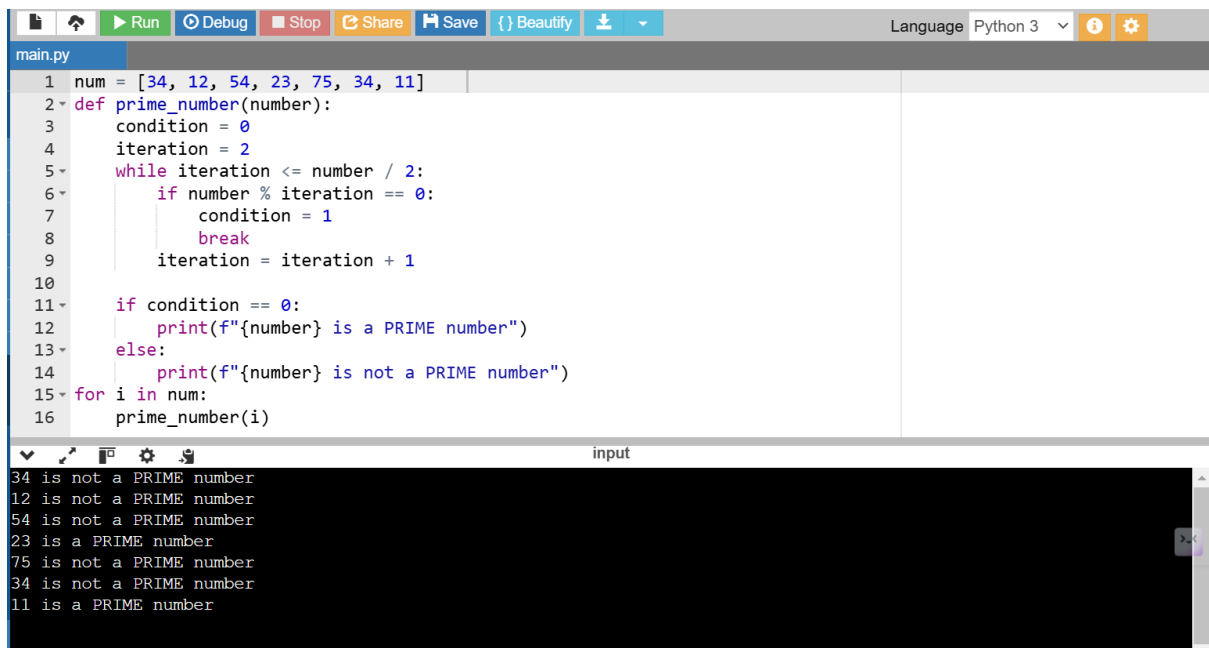
```
main.py
1 num = 21
2 summation = 0
3 c = 1
4
5 while c <= num:
6     summation = c**2 + summation
7     c = c + 1
8 print("The sum of squares is", summation)
```

input

The sum of squares is 3311

...Program finished with exit code 0
Press ENTER to exit console.

2. Prime Numbers:

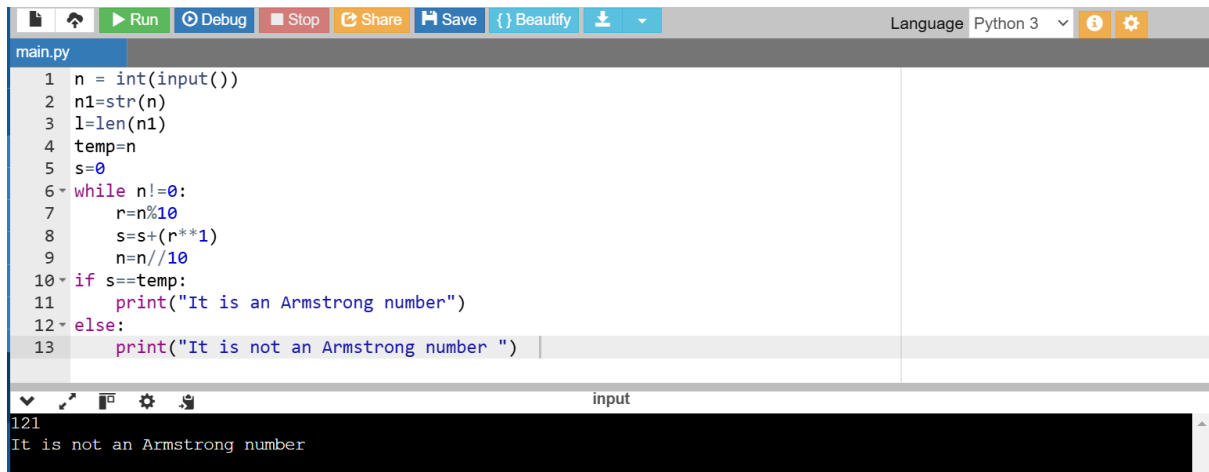


```
main.py
1 num = [34, 12, 54, 23, 75, 34, 11]
2 def prime_number(number):
3     condition = 0
4     iteration = 2
5     while iteration <= number / 2:
6         if number % iteration == 0:
7             condition = 1
8             break
9         iteration = iteration + 1
10
11     if condition == 0:
12         print(f"{number} is a PRIME number")
13     else:
14         print(f"{number} is not a PRIME number")
15 for i in num:
16     prime_number(i)
```

input

34 is not a PRIME number
12 is not a PRIME number
54 is not a PRIME number
23 is a PRIME number
75 is not a PRIME number
34 is not a PRIME number
11 is a PRIME number

3. Armstrong:

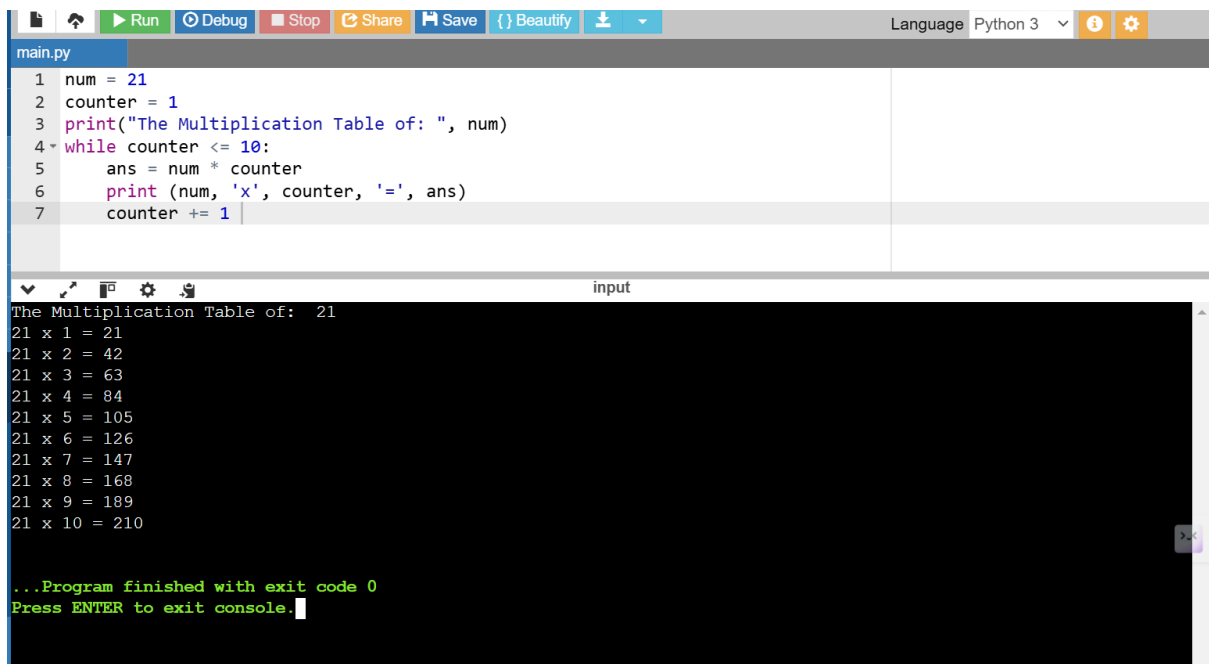


```
main.py
1 n = int(input())
2 n1=str(n)
3 l=len(n1)
4 temp=n
5 s=0
6 while n!=0:
7     r=n%10
8     s=s+(r**l)
9     n=n//10
10 if s==temp:
11     print("It is an Armstrong number")
12 else:
13     print("It is not an Armstrong number ")
```

input

```
121
It is not an Armstrong number
```

4. Multiplication Table using While Loop:



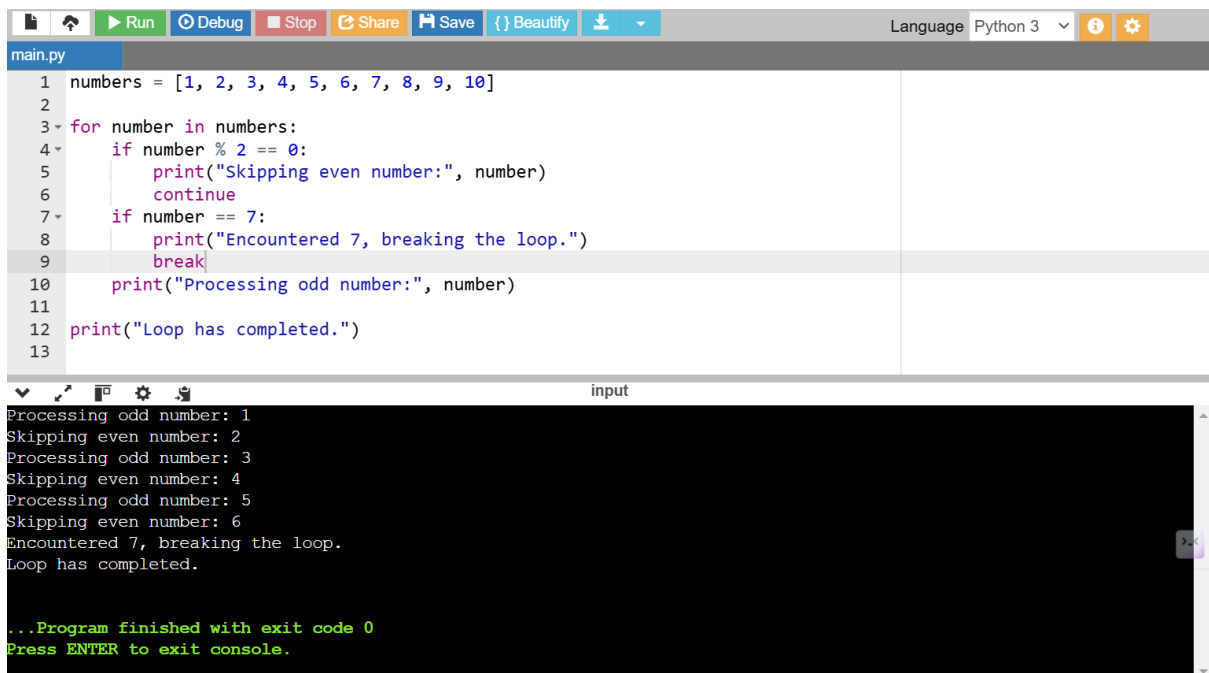
```
main.py
1 num = 21
2 counter = 1
3 print("The Multiplication Table of: ", num)
4 while counter <= 10:
5     ans = num * counter
6     print(num, 'x', counter, '=', ans)
7     counter += 1
```

input

```
The Multiplication Table of: 21
21 x 1 = 21
21 x 2 = 42
21 x 3 = 63
21 x 4 = 84
21 x 5 = 105
21 x 6 = 126
21 x 7 = 147
21 x 8 = 168
21 x 9 = 189
21 x 10 = 210

...Program finished with exit code 0
Press ENTER to exit console.
```

5. Break Statement & Continue Statement:



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a list 'numbers' with values [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. A for loop iterates over these numbers. If a number is even (number % 2 == 0), it prints 'Skipping even number:' followed by the number and then continues the loop. If a number is 7, it prints 'Encountered 7, breaking the loop.' and then breaks the loop. For odd numbers, it prints 'Processing odd number:' followed by the number. After the loop, it prints 'Loop has completed.'.

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 for number in numbers:
4     if number % 2 == 0:
5         print("Skipping even number:", number)
6         continue
7     if number == 7:
8         print("Encountered 7, breaking the loop.")
9         break
10    print("Processing odd number:", number)
11
12 print("Loop has completed.")
13
```

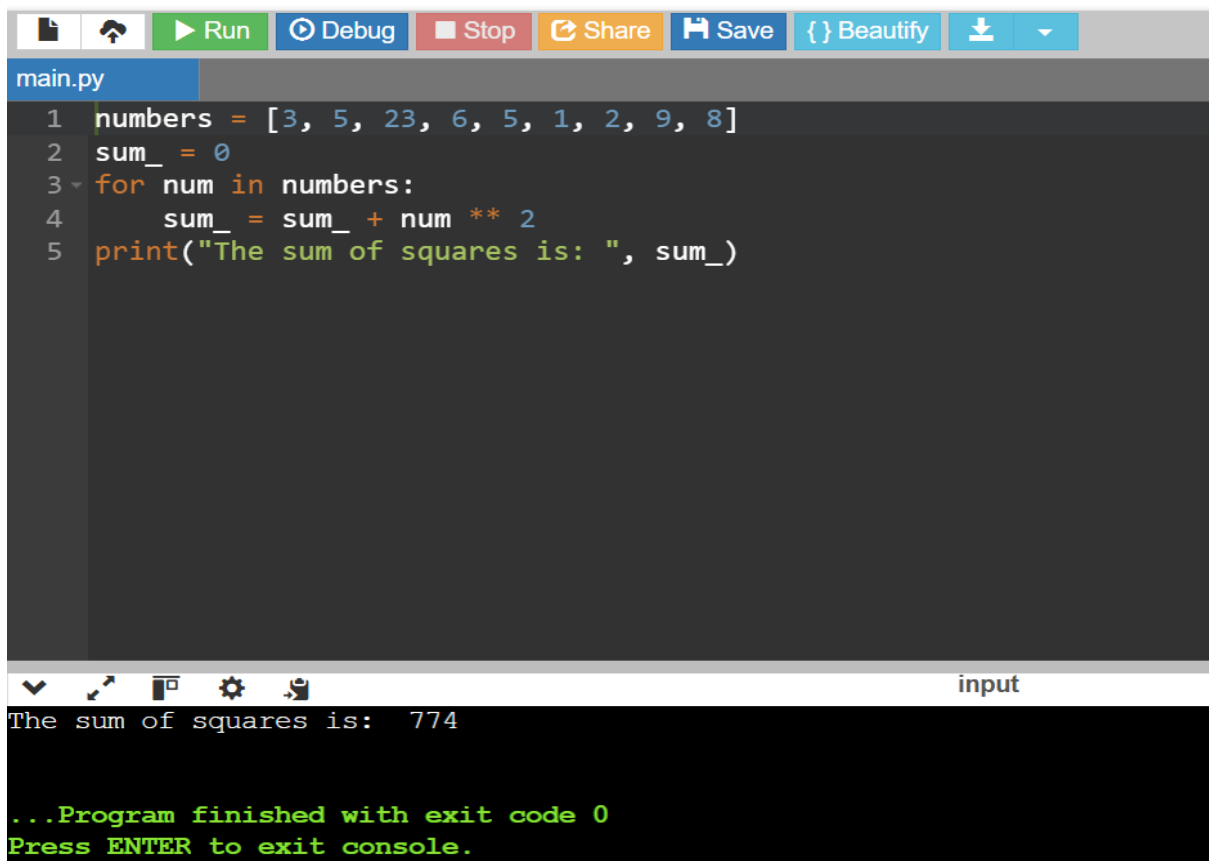
The console output shows the execution of the program:

```
Processing odd number: 1
Skipping even number: 2
Processing odd number: 3
Skipping even number: 4
Processing odd number: 5
Skipping even number: 6
Encountered 7, breaking the loop.
Loop has completed.

...Program finished with exit code 0
Press ENTER to exit console.
```

FOR LOOPS:

1.



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a list 'numbers' with values [3, 5, 23, 6, 5, 1, 2, 9, 8]. It initializes a variable 'sum_' to 0. A for loop iterates over each number in the list, calculating the square of the number (num ** 2) and adding it to 'sum_'. After the loop, it prints 'The sum of squares is: ' followed by the value of 'sum_'.

```
1 numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
2 sum_ = 0
3 for num in numbers:
4     sum_ = sum_ + num ** 2
5 print("The sum of squares is: ", sum_)
```

The console output shows the execution of the program:

```
The sum of squares is: 774

...Program finished with exit code 0
Press ENTER to exit console.
```

2.

```
main.py
1 my_list = [3, 5, 6, 8, 4]
2 for iter_var in range( len( my_list ) ):
3     my_list.append(my_list[iter_var] + 2)
4 print( my_list )
```

input

[3, 5, 6, 8, 4, 5, 7, 8, 10, 6]

...Program finished with exit code 0
Press ENTER to exit console.

3.

```
main.py
1 student_name_1 = 'Itika'
2 student_name_2 = 'Parker'
3 records = {'Itika': 90, 'Arshia': 92, 'Peter': 46}
4 def marks(student_name):
5     for a_student in records:
6         if a_student == student_name:
7             return records[a_student]
8         break
9     else:
10        return f'There is no student of name {student_name} in the records'
11 print(f"Marks of {student_name_1} are: ", marks(student_name_1))
12 print(f"Marks of {student_name_2} are: ", marks(student_name_2))
13
```

input

Marks of Itika are: 90
Marks of Parker are: There is no student of name Parker in the records

...Program finished with exit code 0
Press ENTER to exit console.

4.



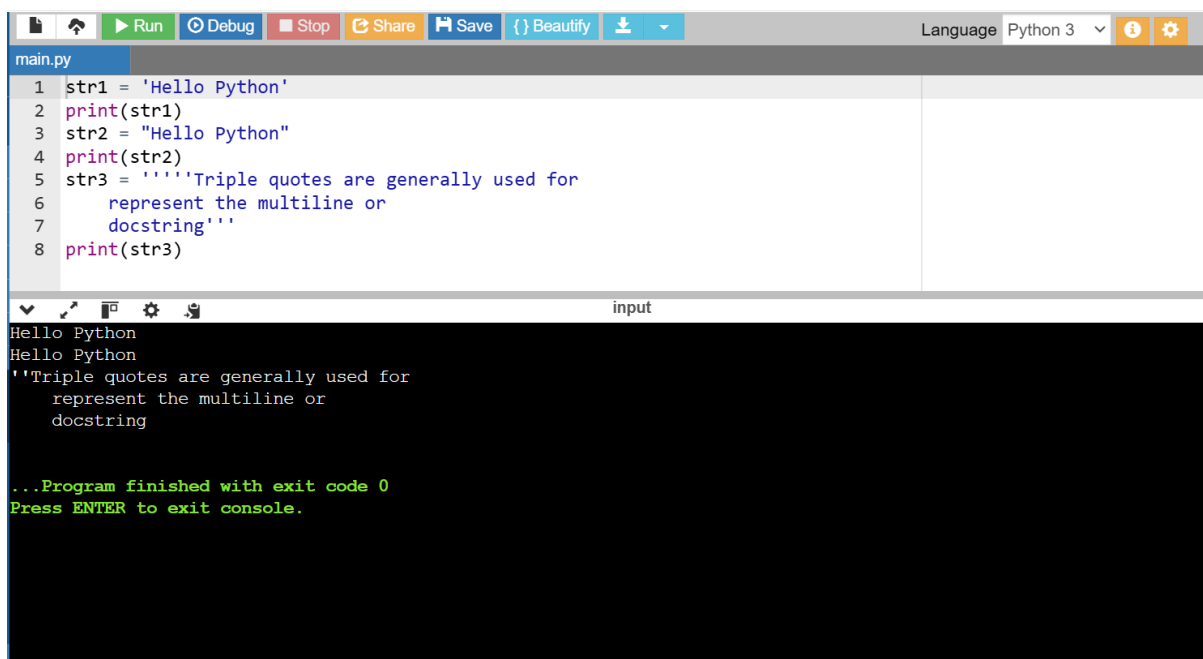
The screenshot shows a Python IDE with a toolbar at the top containing icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The editor window, titled 'main.py', contains the following Python code:

```
1 import random
2 numbers = [ ]
3 for val in range(0, 11):
4     numbers.append( random.randint( 0, 11 ) )
5 for num in range( 0, 11 ):
6     for i in numbers:
7         if num == i:
8             print( num, end = " " )
```

Below the editor is a console window with a toolbar and the label 'input'. It displays the output of the program as a sequence of numbers: 0 1 2 3 4 5 6 7 9 10. At the bottom of the console, it shows the message: "...Program finished with exit code 0 Press ENTER to exit console."

STRINGS

1.



The screenshot shows a Python IDE with a toolbar at the top. The editor window, titled 'main.py', contains the following Python code:

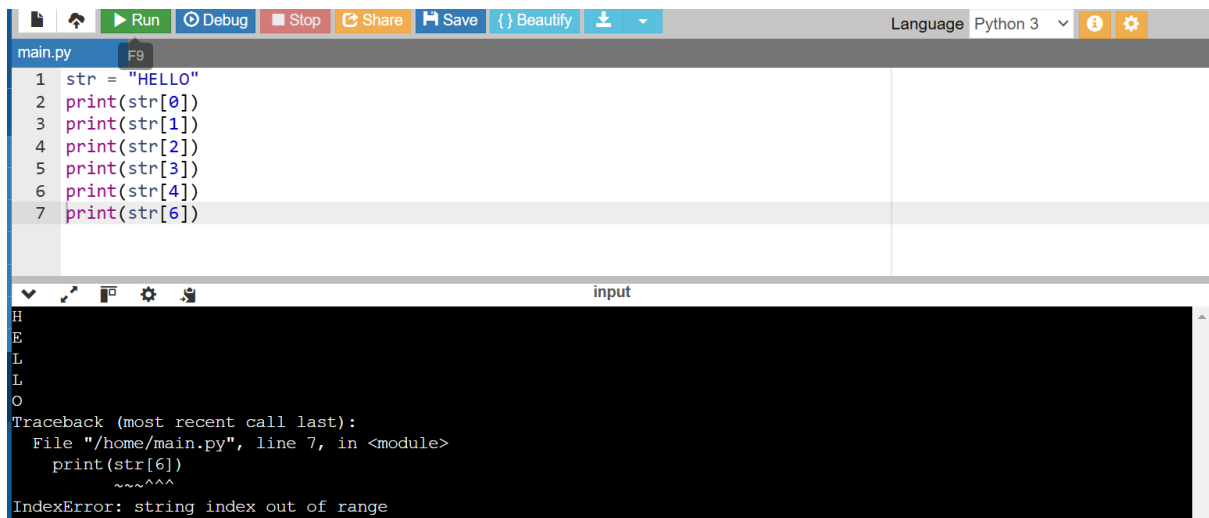
```
1 str1 = 'Hello Python'
2 print(str1)
3 str2 = "Hello Python"
4 print(str2)
5 str3 = '''Triple quotes are generally used for
6     represent the multiline or
7     docstring'''
8 print(str3)
```

Below the editor is a console window with a toolbar and the label 'input'. It displays the output of the program:

```
Hello Python
Hello Python
'''Triple quotes are generally used for
    represent the multiline or
    docstring'''
```

At the bottom of the console, it shows the message: "...Program finished with exit code 0 Press ENTER to exit console."

2. Indexing and Splitting:



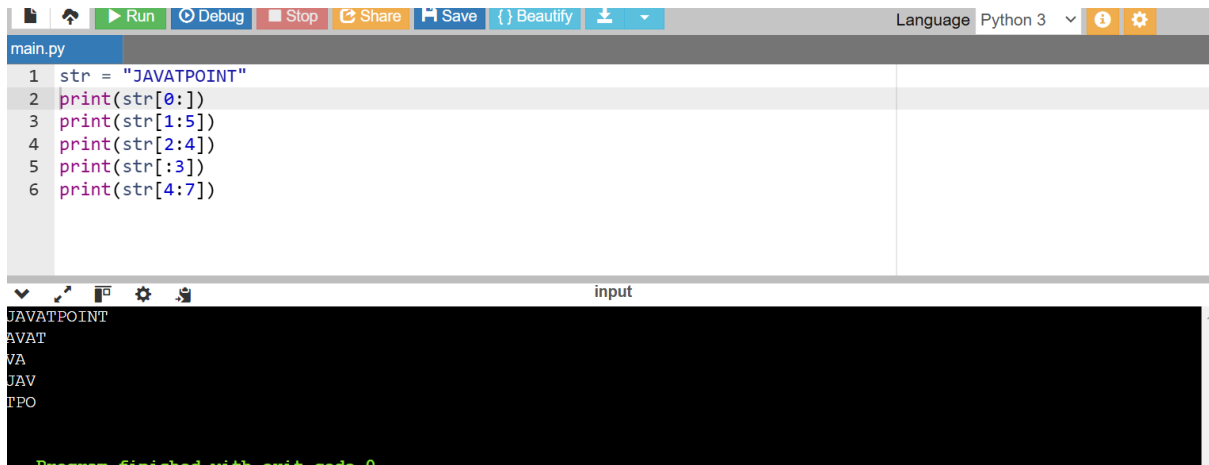
The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 str = "HELLO"  
2 print(str[0])  
3 print(str[1])  
4 print(str[2])  
5 print(str[3])  
6 print(str[4])  
7 print(str[6])
```

The IDE's output console shows the following error message:

```
H  
E  
L  
L  
O  
Traceback (most recent call last):  
  File "/home/main.py", line 7, in <module>  
    print(str[6])  
    ~~~^^^  
IndexError: string index out of range
```

3.



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 str = "JAVATPOINT"  
2 print(str[0:])  
3 print(str[1:5])  
4 print(str[2:4])  
5 print(str[:3])  
6 print(str[4:7])
```

The IDE's output console shows the following output:

```
JAVATPOINT  
AVAT  
VA  
JAV  
TPO  
Program finished with exit code 0
```

4.



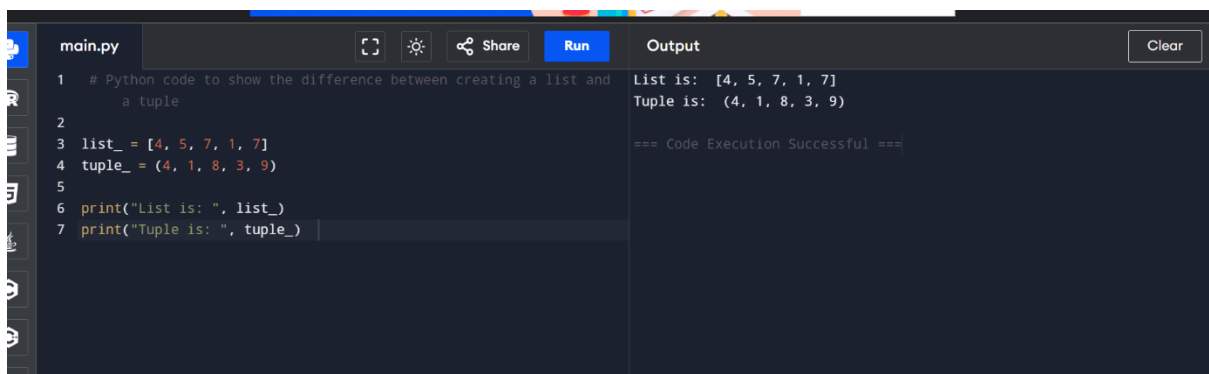
```
main.py
1 str = "Hello"
2 str1 = " world"
3 print(str*3)
4 print(str+str1)
5 print(str[4])
6 print(str[2:4]);
7 print('w' in str)
8 print('wo' not in str1)
9 print(r'C://python37')
10 print("The string str : %s"%(str))
```

input

```
HelloHelloHello
Hello world
o
ll
False
False
C://python37
The string str : Hello
```

Python List vs Tuple

1. List and Tuple Syntax Differences:



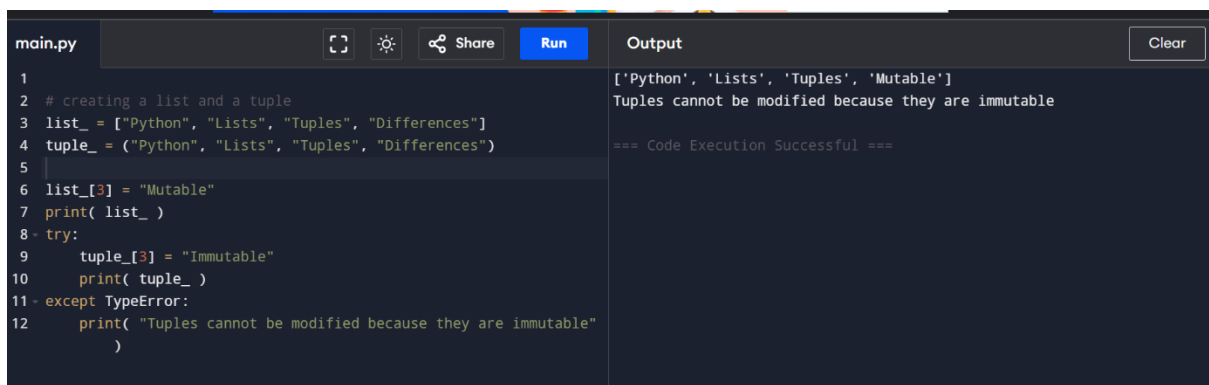
```
main.py
1 # Python code to show the difference between creating a list and a tuple
2
3 list_ = [4, 5, 7, 1, 7]
4 tuple_ = (4, 1, 8, 3, 9)
5
6 print("List is: ", list_)
7 print("Tuple is: ", tuple_)
```

Output

```
List is: [4, 5, 7, 1, 7]
Tuple is: (4, 1, 8, 3, 9)

=== Code Execution Successful ===
```

2. Updating the element of list and tuple at a particular index :



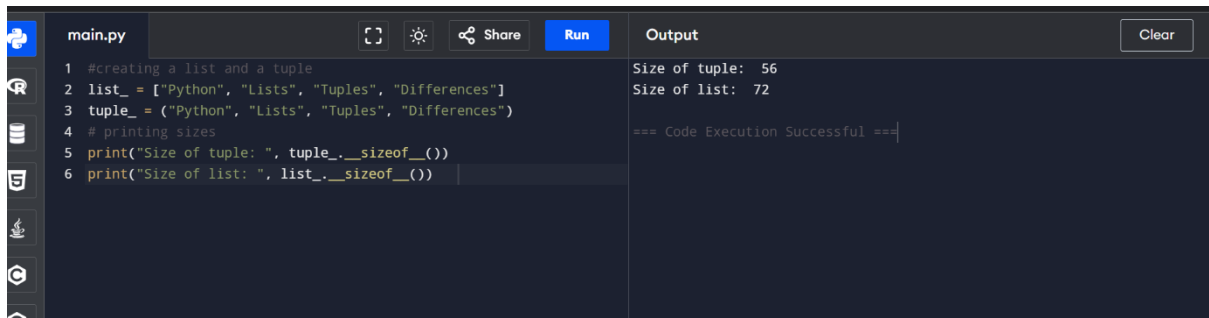
```
main.py
1
2 # creating a list and a tuple
3 list_ = ["Python", "Lists", "Tuples", "Differences"]
4 tuple_ = ("Python", "Lists", "Tuples", "Differences")
5
6 list_[3] = "Mutable"
7 print( list_ )
8 - try:
9     tuple_[3] = "Immutable"
10    print( tuple_ )
11 - except TypeError:
12    print( "Tuples cannot be modified because they are immutable" )
```

Output

```
['Python', 'Lists', 'Tuples', 'Mutable']
Tuples cannot be modified because they are immutable

=== Code Execution Successful ===
```


3. Code to show the difference in the size of a list and a tuple:



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with icons for file explorer, search, and other notebook functions. The main area is split into two panes. The left pane, titled 'main.py', contains the following Python code:

```
1 #creating a list and a tuple
2 list_ = ["Python", "Lists", "Tuples", "Differences"]
3 tuple_ = ("Python", "Lists", "Tuples", "Differences")
4 # printing sizes
5 print("Size of tuple: ", tuple_.__sizeof__())
6 print("Size of list: ", list_.__sizeof__())
```

The right pane, titled 'Output', shows the results of the code execution:

```
Size of tuple: 56
Size of list: 72

=== Code Execution Successful ===
```

At the top of the interface, there are buttons for 'Run' (in blue) and 'Clear'. A 'Share' button is also visible next to a settings icon.