

Arithmetic Operators

The screenshot shows a Python code editor interface. On the left, there's a sidebar with icons for various languages: Python (selected), R, SQL, HTML, CSS, JS, and Go. The main area has a file named "main.py" with the following code:

```
1 a=46
2 b=4
3
4 print("For a=",a,"and b=",b,"Calculate 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 of two numbers:a//b =',a//b)
11 print('6.Remainder of two numbers:a%b =',a%b)
12 print('7.Exponent of two numbers:a^b =',a**b)
```

At the top right, there are buttons for Run, Share, and a refresh icon. The "Run" button is highlighted in blue. To the right of the code area is a "Output" section containing the results of the execution:

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 of two numbers:a//b = 11
6.Remainder of two numbers:a%b = 2
7.Exponent of two numbers:a^b = 4477456
== Code Execution Successful ==

Comparison Operators

The screenshot shows a Python code editor interface. On the left, there's a sidebar with icons for various languages: Python (selected), R, SQL, CSS, JS, and Go. The main area has a file named "main.py" with the following code:

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

At the top right, there are buttons for Run, Share, and a refresh icon. The "Run" button is highlighted in blue. To the right of the code area is a "Output" section containing the results of the execution:

For a= 46 and b= 4
Calculate 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 than b: True
6.a is less than b: False
== Code Execution Successful ==

Assignment Operators

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 a=34
2 b=6
3
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)
10 print('a//=b',a//b)
```

The 'Output' tab shows the results of running the code:

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

== Code Execution Successful ==
```

Bitwise Operators

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

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

The 'Output' tab shows the results of running the code:

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

== Code Execution Successful ==
```

Logical Operators

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

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

The 'Output' tab shows the results of running the code, explaining the conditions for a = 7:

```
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

== Code Execution Successful ==
```

Membership Operators

The screenshot shows a Python code editor interface with the following details:

- File:** main.py
- Code:**

```
1 myList = [12, 22, 28, 35, 42, 49, 54, 65, 92, 103, 245, 874]
2
3 x = 31
4 y = 28
5
6 print("Given List:", myList)
7
8 if x not in myList:
9     print("x =", x,"is NOT present in the given list.")
10 else:
11     print("x =", x,"is present in the given list.")
12 if y in myList:
13     print("y =", y,"is present in the given list.")
14 else:
15     print("y =", y,"is NOT present in the given list.)
```
- Tools:** Includes icons for copy, paste, share, run, and clear.
- Output:**

```
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.

== Code Execution Successful ==
```

Identity Operators

The screenshot shows a Python code editor interface with the following details:

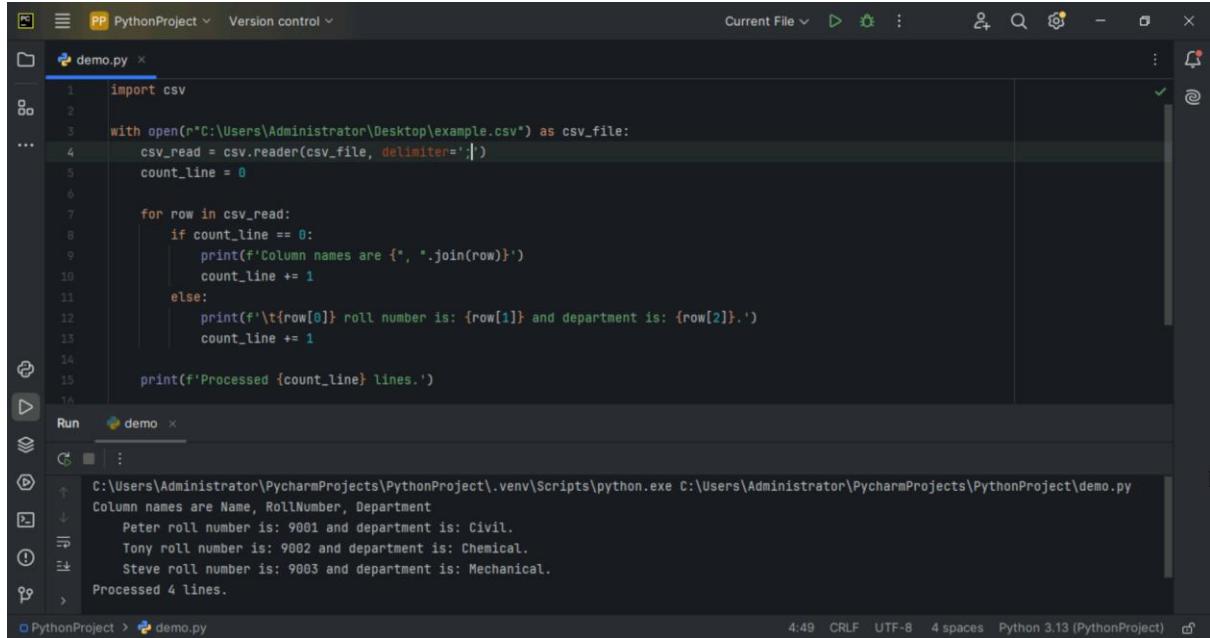
- File:** main.py
- Code:**

```
1 a = ["Rose", "Lotus"]
2 b = ["Rose", "Lotus"]
3 c = a
4
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)
```
- Tools:** Includes icons for copy, paste, share, run, and clear.
- Output:**

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

== Code Execution Successful ==
```

How to read CSV file in python?



The screenshot shows the PyCharm IDE interface. The code editor displays a Python script named 'demo.py' which reads a CSV file named 'example.csv'. The script prints the column names and then iterates through the rows, printing each row's data. The run tool window at the bottom shows the execution results, including the column names and three data rows: Peter, Tony, and Steve, along with their roll numbers and department.

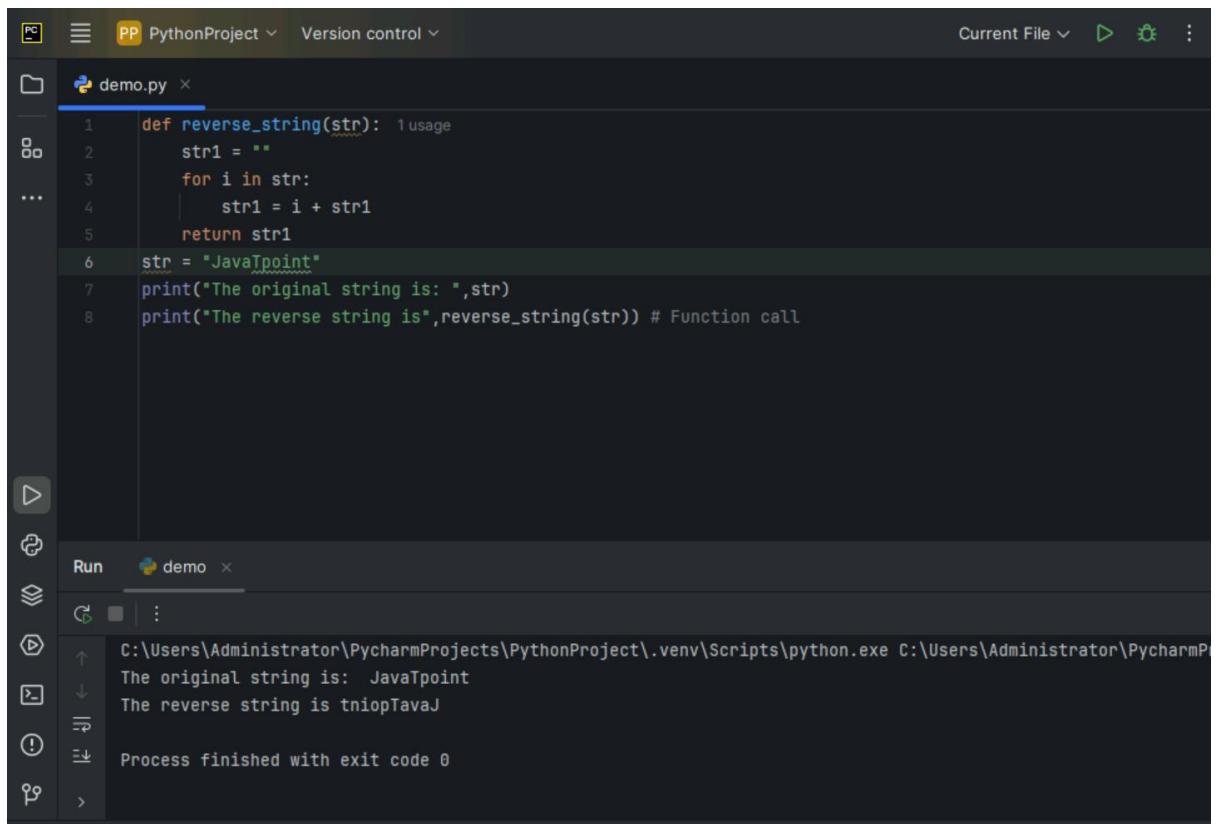
```
import csv
with open(r'C:\Users\Administrator\Desktop\example.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    count_line = 0
    for row in csv_reader:
        if count_line == 0:
            print(f'Column names are {", ".join(row)}')
            count_line += 1
        else:
            print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}.')
            count_line += 1
print(f'Processed {count_line} lines.'
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Column names are Name, RollNumber, Department
Peter roll number is: 9001 and department is: Civil.
Tony roll number is: 9002 and department is: Chemical.
Steve roll number is: 9003 and department is: Mechanical.
Processed 4 lines.
```

How to reverse a string in Python?

Using for loop



The screenshot shows the PyCharm IDE interface. The code editor displays a Python script named 'demo.py' which defines a function 'reverse_string' that takes a string as input and returns its reverse. The script then calls this function with the string 'JavaPoint' and prints both the original and reversed strings. The run tool window at the bottom shows the execution results, including the original string 'JavaPoint' and its reversal 'tniopTavaJ'.

```
def reverse_string(str): 1 usage
    str1 = ""
    for i in str:
        str1 = i + str1
    return str1
str = "JavaPoint"
print("The original string is: ",str)
print("The reverse string is",reverse_string(str)) # Function call
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmP
The original string is: JavaPoint
The reverse string is tniopTavaJ
Process finished with exit code 0
```

Using while loop

The screenshot shows the PyCharm IDE interface. The top bar displays project navigation and version control. The left sidebar shows a file tree with a selected file named 'demo.py'. The main code editor window contains the following Python code:

```
str = "JavaTpoint"
print ("The original string is : ",str)
reverse_String = ""
count = len(str)
while count > 0:
    reverse_String += str[ count - 1 ]
    count = count - 1
print ("The reversed string using a while loop is : ",reverse_String)
```

The bottom right panel shows the run output. It starts with the command: C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe demo.py. The output then displays the original string and the reversed string:

```
The original string is : JavaTpoint
The reversed string using a while loop is : tniopTavaJ
```

Using the slice ([]) operator

The screenshot shows a PyCharm interface with a dark theme. On the left is a toolbars and a file browser. The main area contains a code editor with a Python script named 'demo.py'. The code defines a function 'reverse' that takes a string 'str' and returns its reverse using the slice operator 'str[::-1]'. It then prints the original string 'JavaTpoint' and the reversed string 'tniopTavaJ'. The bottom panel shows the run output, which includes the command 'C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe demo', the printed strings, and a message 'Process finished with exit code 0'.

```
def reverse(str):    1 usage
    str = str[::-1]
    return str

s = "JavaTpoint"
print ("The original string is : ",s)
print ("The reversed string using extended slice operator is : |",reverse(s))
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe demo
The original string is : JavaTpoint
The reversed string using extended slice operator is : tniopTavaJ
Process finished with exit code 0
```

Using reverse function with join

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python script:

```
def reverse(str):    string = "".join(reversed(str)) # reversed() function inside the join() function
    return string
s = "JavaTpoint"
print ("The original string is : ",s)
print ("The reversed string using reversed() is : ",reverse(s) )
```

The run tool window at the bottom shows the execution results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\A
The original string is : JavaTpoint
The reversed string using reversed() is : tniopTavaJ
```

Using recursion()

The screenshot shows the PyCharm IDE interface. The top bar displays project and version control information. The left sidebar contains file navigation icons. The main editor window shows a Python script named 'demo.py' with the following code:

```
def reverse(str):    2 usages
    if len(str) == 0: # Checking the lenght of string
        return str
    else:
        return reverse(str[1:]) + str[0]
str = "Devansh Sharma"
print ("The original string is : ", str)
print ("The reversed string(using recursion) is : ", reverse(str))
```

The bottom panel is the 'Run' tool window, which includes a terminal tab. The terminal output shows the execution of the script and its results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python
The original string is : Devansh Sharma
The reversed string(using recursion) is : amrahS hsnaveD
```

Basic if conditions

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python script:

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

The "Run" tab is selected, showing the command "demo" and the output window below it. The output window displays the execution of the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject
enter the number:8
The Given number is an even number
Process finished with exit code 0
```

Program to print largest of three numbers

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor window contains the following Python script:

```
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");
```

The "Run" tab is selected, showing the command "demo" and the output window below it. The output window displays the following text:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\
Enter a: 10
Enter b: 32
Enter c: 21
From the above three numbers given b is largest
```

Program to check eligibility of voters

The screenshot shows the PyCharm IDE interface. The top bar displays 'PC', 'PythonProject', and 'Version control'. The left sidebar shows a file tree with 'demo.py'. The main editor window contains the following code:

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

The 'Run' tab is selected, showing the output of the program:

```
C:\Users\Administrator\PycharmProjects\PythonProject
Enter your age: 12
Sorry! you have to wait !!
Process finished with exit code 0
```

Program to check odd/even number

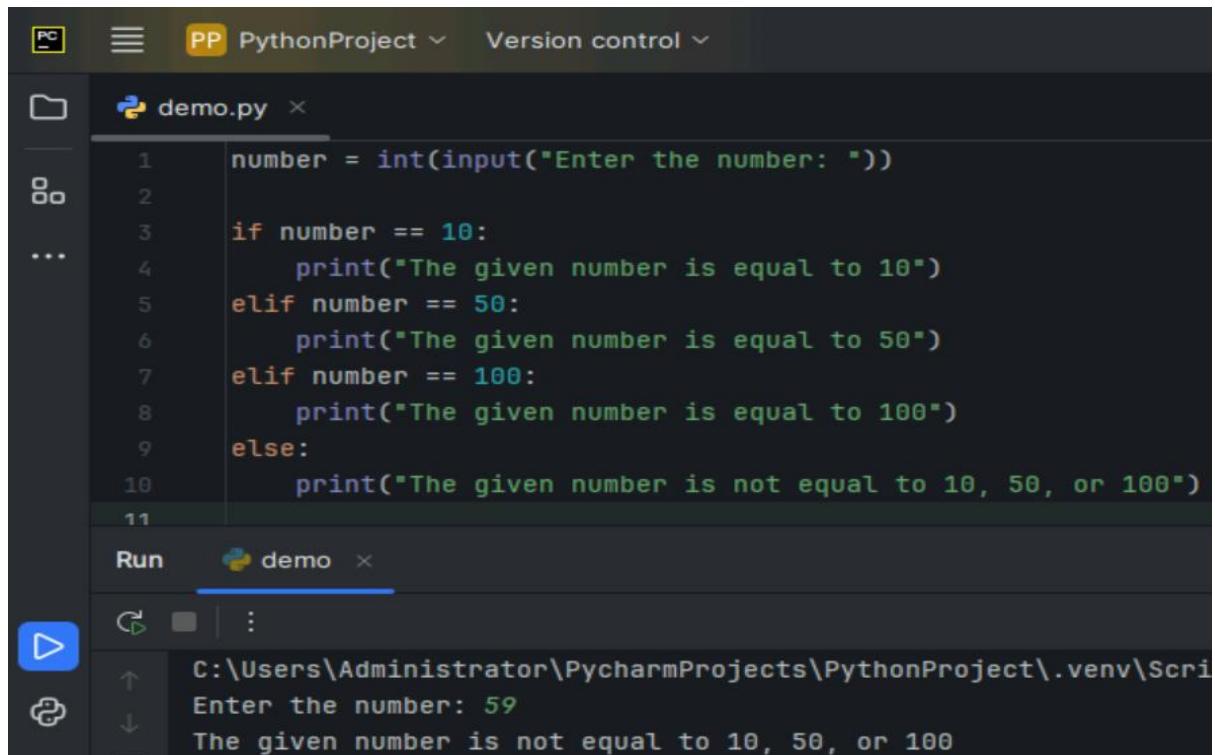
The screenshot shows the PyCharm IDE interface. The top bar displays 'PC', 'PythonProject', and 'Version control'. The left sidebar shows a file tree with 'demo.py'. The main editor window contains the following code:

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

The 'Run' tab is selected, showing the output of the program:

```
C:\Users\Administrator\PycharmProjects\PythonProject
Enter the number: 4
The given number is an even number
```

Elif basic program



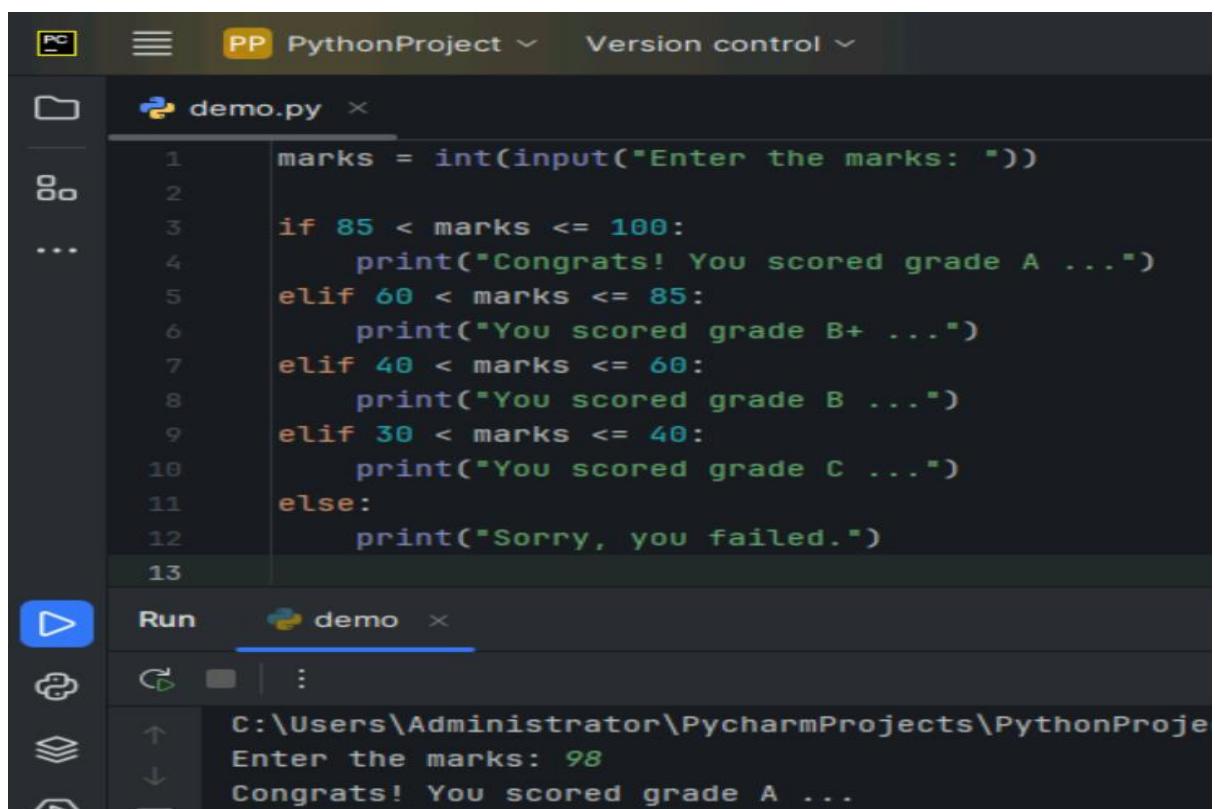
The screenshot shows the PyCharm IDE interface. The top bar displays project names "PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor contains the following Python code:

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

The run configuration "Run demo" is selected in the bottom toolbar. The terminal window shows the output of running the script with the input "59":

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Script
Enter the number: 59
The given number is not equal to 10, 50, or 100
```

Q



The screenshot shows the PyCharm IDE interface. The top bar displays project names "PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor contains the following Python code:

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

The run configuration "Run demo" is selected in the bottom toolbar. The terminal window shows the output of running the script with the input "98":

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Script
Enter the marks: 98
Congrats! You scored grade A ...
```

LOOPS

Working of for loop

The screenshot shows the PyCharm IDE interface. The top bar displays project names "PC", "PythonProject", and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor window contains the following Python code:

```
1 numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
2
3 squares = []
...
5 for value in numbers:
6     squares.append(value ** 2)
7
8 print("The list of squares is", squares)
9 |
```

Below the code editor is a "Run" tab with "demo" selected. The run output window shows the command "C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py" followed by the output: "The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]".

If-else in for loops

The screenshot shows the PyCharm IDE interface. The top bar displays project names "PC", "PythonProject", and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor window contains the following Python code:

```
1 string = "Python Loop"
2
3 for s in string:
4     if s == "o":
5         print("If block")
6     else:
7         print(s)
8
```

Below the code editor is a "Run" tab with "demo" selected. The run output window shows the command "C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py" followed by the output: "P" (on a new line), "y" (on a new line), "t" (on a new line), "h" (on a new line), "If block" (on a new line), "n" (on a new line), "L" (on a new line), "If block" (on a new line), "If block" (on a new line), and "P" (on a new line).

else in for loop

The screenshot shows the PyCharm IDE interface. The top bar displays 'PP PythonProject' and 'Version control'. The main editor window shows a file named 'demo.py' with the following code:

```
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)
for value in tuple_:
    if value % 2 != 0:
        print(value)
    else:
        print("These are the odd numbers present in the tuple")
```

The run output window below shows the execution results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects
3
9
3
9
7
These are the odd numbers present in the tuple
```

working of range function

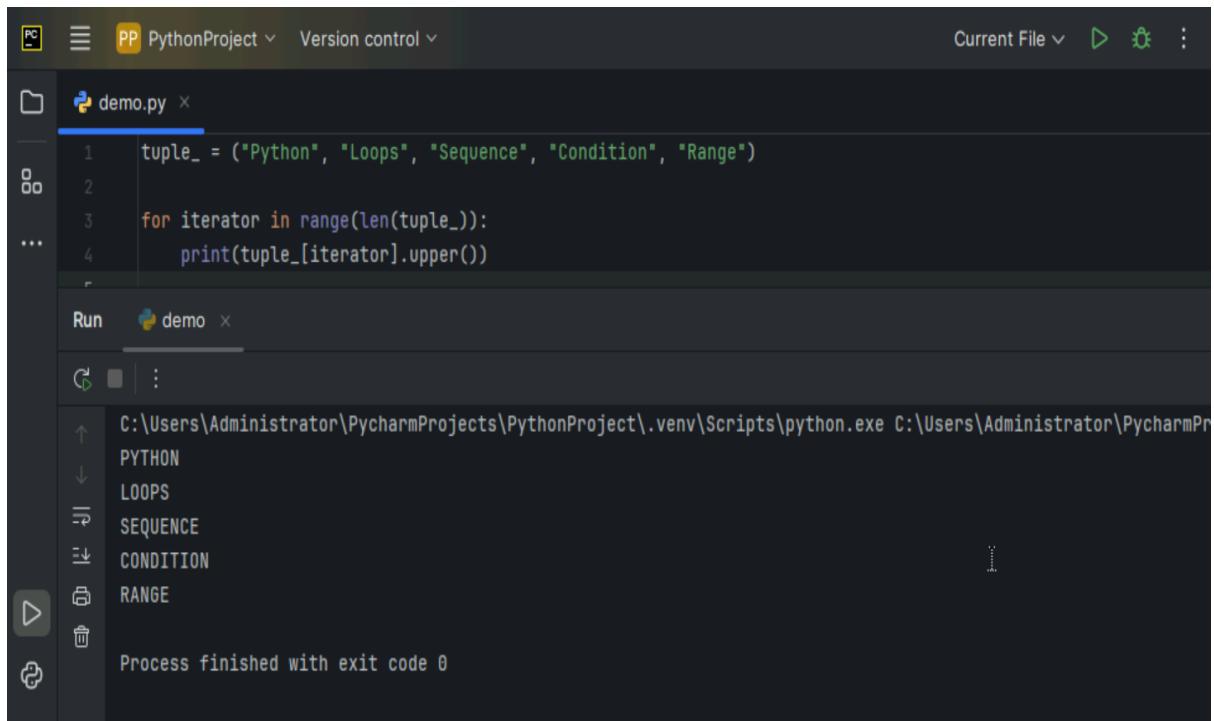
The screenshot shows the PyCharm IDE interface. The top bar displays 'PP PythonProject' and 'Version control'. The main editor window shows a file named 'demo.py' with the following code:

```
print(range(15))
print(list(range(15)))
print(list(range(4, 9)))
print(list(range(5, 25, 4)))
```

The run output window below shows the execution results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmP
range(0, 15)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[4, 5, 6, 7, 8]
[5, 9, 13, 17, 21]
Process finished with exit code 0
```

Indexing in for loop



The screenshot shows a PyCharm interface with a dark theme. In the top navigation bar, there are icons for PC, file, and project (PP PythonProject), followed by 'Version control'. On the right, there are buttons for 'Current File', a search icon, and a settings icon. The main area shows a code editor with a file named 'demo.py' containing the following code:

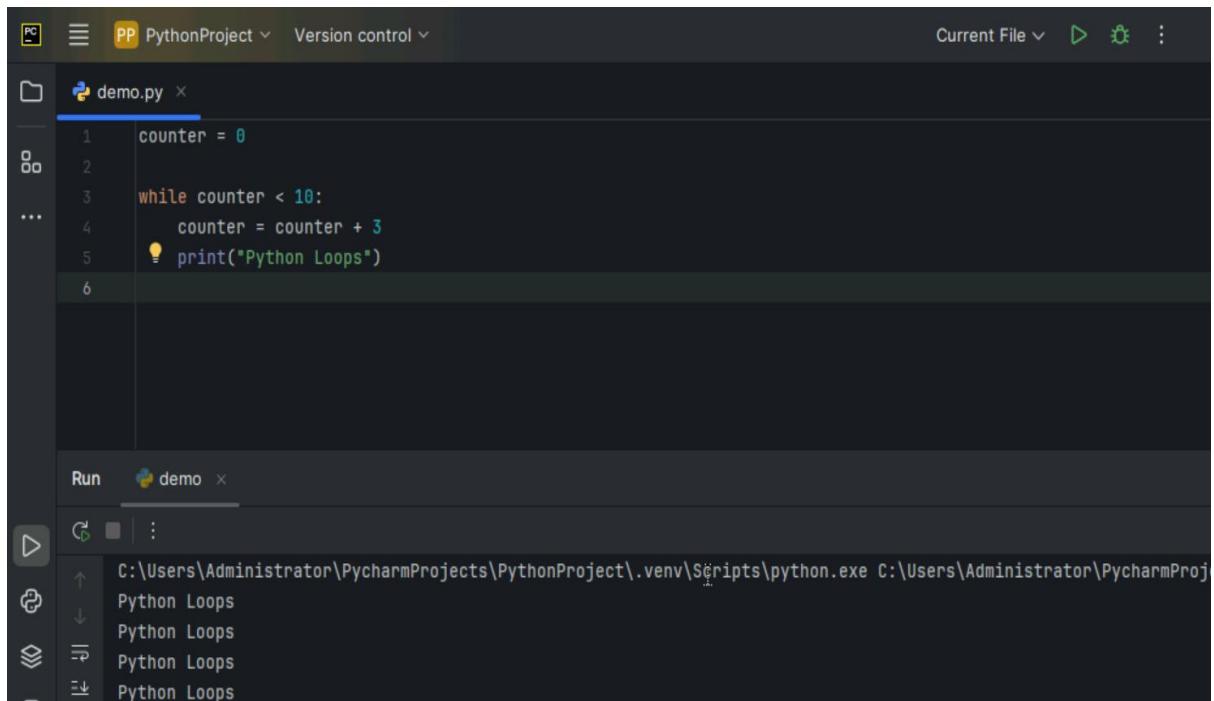
```
tuple_ = ("Python", "Loops", "Sequence", "Condition", "Range")
for iterator in range(len(tuple_)):
    print(tuple_[iterator].upper())
```

Below the code editor is a 'Run' section with a dropdown set to 'demo'. The run output window shows the following terminal output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProj
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE
```

At the bottom of the run output, it says 'Process finished with exit code 0'.

While loop counter



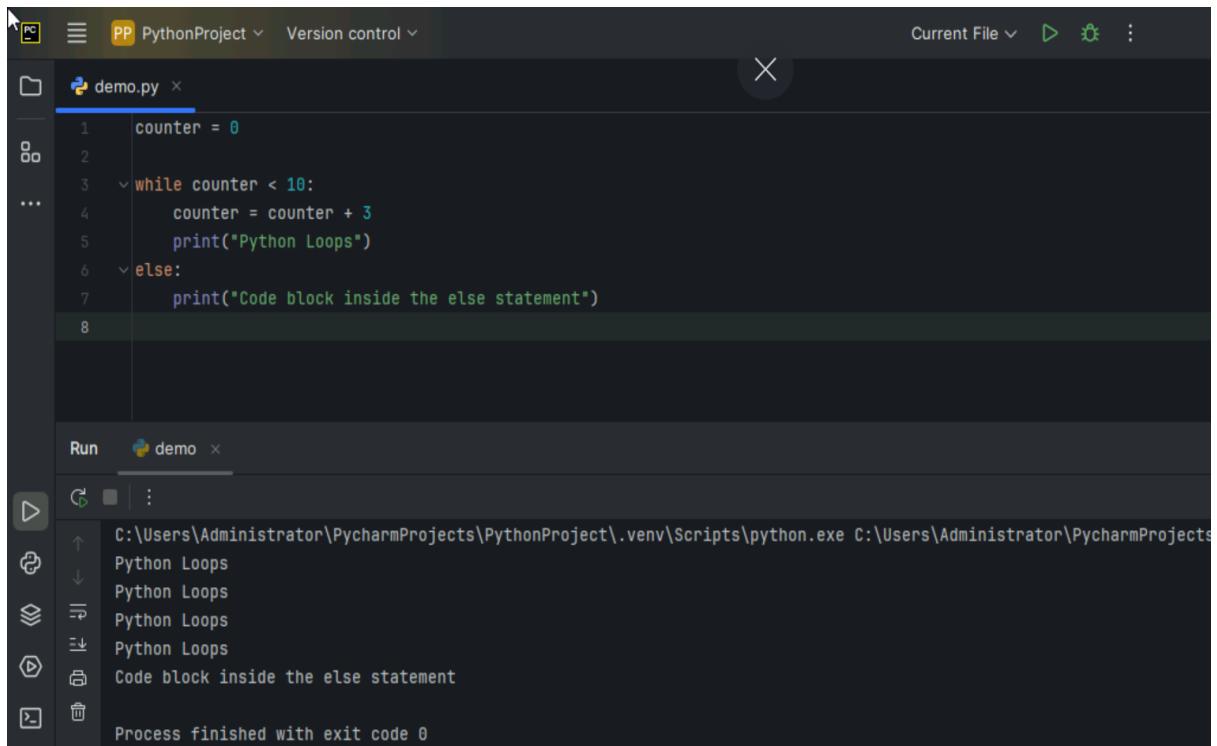
The screenshot shows a PyCharm interface with a dark theme. In the top navigation bar, there are icons for PC, file, and project (PP PythonProject), followed by 'Version control'. On the right, there are buttons for 'Current File', a search icon, and a settings icon. The main area shows a code editor with a file named 'demo.py' containing the following code:

```
counter = 0
while counter < 10:
    counter = counter + 3
    print("Python Loops")
```

Below the code editor is a 'Run' section with a dropdown set to 'demo'. The run output window shows the following terminal output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProj
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
Python Loops
```

Else statement in while loop

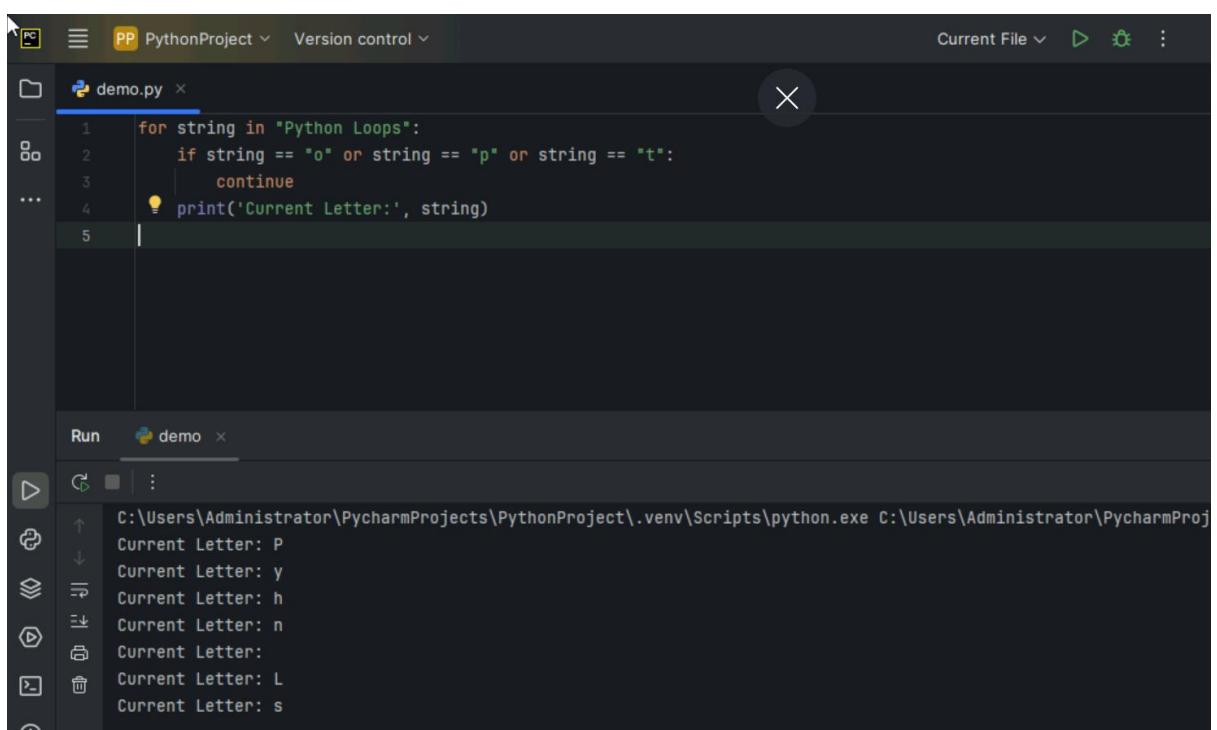


```
demo.py
1 counter = 0
2
3 while counter < 10:
4     counter = counter + 3
5     print("Python Loops")
6 else:
7     print("Code block inside the else statement")
8
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Python Loops
Code block inside the else statement
Process finished with exit code 0
```

Working of Continue statement



```
demo.py
1 for string in "Python Loops":
2     if string == "o" or string == "p" or string == "t":
3         continue
4     print('Current Letter:', string)
5
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Current Letter: P
Current Letter: y
Current Letter: h
Current Letter: n
Current Letter:
Current Letter: L
Current Letter: s
```

Working of break statement

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python code:

```
for string in "Python Loops":  
    if string == 'L':  
        break  
    print('Current Letter:', string)
```

The run output window at the bottom shows the execution results:

```
Current Letter: P  
Current Letter: y  
Current Letter: t  
Current Letter: h  
Current Letter: o  
Current Letter: n  
Current Letter:  
Process finished with exit code 0
```

Working of pass statement

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python code:

```
# Python program to show how the pass statement works  
for string in "Python Loops":  
    pass  
    print( 'Last Letter:', string)
```

The run output window at the bottom shows the execution results:

```
Last Letter: s  
Process finished with exit code 0
```

Sum of squares using for loop

The screenshot shows the PyCharm IDE interface. The top navigation bar includes icons for file operations, a project named "PythonProject", and version control. The main area displays a Python script named "demo.py". The code defines a list of numbers, initializes a sum variable to 0, and then iterates through the list using a for loop to calculate the sum of squares. The output window below shows the command run and the resulting output: "The sum of squares is: 774".

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

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject\demo.py
The sum of squares is: 774
Process finished with exit code 0
```

Creating a list using range()

The screenshot shows the PyCharm IDE interface. The top navigation bar includes icons for file operations, a project named "PythonProject", and version control. The main area displays a Python script named "demo.py". The code initializes a list "my_list" with values [3, 5, 6, 8, 4]. It then uses a for loop with "range(len(my_list))" to iterate over the list's length, appending the value at each index plus 2 to the end of the list. Finally, it prints the modified list. The output window shows the command run and the resulting output: "[3, 5, 6, 8, 4, 5, 7, 8, 10, 6]".

```
my_list = [3, 5, 6, 8, 4]
# Iterate over the original length of the list
for iter_var in range(len(my_list)):
    my_list.append(my_list[iter_var] + 2)
print(my_list)
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject\demo.py
[3, 5, 6, 8, 4, 5, 7, 8, 10, 6]
Process finished with exit code 0
```

Q

The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The main editor window contains the following Python code:

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

The run tab at the bottom shows the output of the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
The sum of squares is: 774
Process finished with exit code 0
```

Q

The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The main editor window contains the following Python code:

```
student_name_1 = 'Itika'
student_name_2 = 'Parker'

records = {'Itika': 90, 'Arshia': 92, 'Peter': 46}

def marks(student_name):
    for a_student in records:
        if a_student == student_name:
            return records[a_student]
    else:
        return f'There is no student of name {student_name} in the records'

print(f'Marks of {student_name_1} are: ', marks(student_name_1))
print(f'Marks of {student_name_2} are: ', marks(student_name_2))
```

The run tab at the bottom shows the output of the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Marks of Itika are: 90
Marks of Parker are: There is no student of name Parker in the records
```

Nested loops

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and various tool icons. The left sidebar shows a file tree with "demo.py" selected. The main editor window contains the following Python code:

```
import random
numbers = []
for val in range(0, 11):
    numbers.append(random.randint(a: 0, b: 11))
for num in range(0, 11):
    for i in numbers:
        if num == i:
            print(num, end=" ")
```

The code uses nested loops to generate a list of random numbers and then prints each number followed by a space. The run tab at the bottom shows the output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmPr
0 1 1 2 2 2 3 6 8 10
Process finished with exit code 0
```

While loop

The screenshot shows the PyCharm IDE interface. The top bar has icons for file operations, a project named "PythonProject", and version control. The main area displays a code editor with a file named "demo.py". The code contains a simple while loop that prints numbers from 1 to 10. The run tab at the bottom shows the output of the script.

```
i = 1
while i <= 10:
    print(i, end=' ')
    i += 1
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProj
1 2 3 4 5 6 7 8 9 10
Process finished with exit code 0
```

While loop with conditions

The screenshot shows the PyCharm IDE interface. The top bar has icons for file operations, a project named "PythonProject", and version control. The main area displays a code editor with a file named "demo.py". The code contains a while loop that prints numbers from 1 to 50, but only those that are divisible by 5 or 7. The run tab at the bottom shows the output of the script.

```
i=1
while i<51:
    if i%5 == 0 or i%7==0 :
        print(i, end=' ')
    i+=1
```

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\
5 7 10 14 15 20 21 25 28 30 35 40 42 45 49 50
Process finished with exit code 0
```

Summation

The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor window contains the following Python code:

```
1 num = 15
2 summation = 0
3 c = 1
...
5 while c <= num:
6     summation = c**2 + summation
7     c = c + 1
8
9 print("The sum of squares is", summation)
10
```

The bottom panel shows the "Run" tab with "demo" selected. The run output shows the result of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProj...
The sum of squares is 1240
```

Prime number via while loop

The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main code editor window contains the following Python code:

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

The bottom panel shows the "Run" tab with "demo" selected. The run output shows the results for each number in the list:

```
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
```

Armstrong number using while loop

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The main area shows a file named "demo.py" with the following code:

```
n = int(input("Enter a number: "))
print(n)
n1 = str(n)
l = len(n1)
temp = n
s = 0
while n != 0:
    r = n % 10
    s = s + (r ** l) # Raise each digit to the power of the number of digits
    n = n // 10
if s == temp:
    print("It is an Armstrong number")
else:
    print("It is not an Armstrong number")
```

The "Run" tab is selected, showing the output of running the script "demo". The output window displays:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Enter a number: 153
153
It is an Armstrong number
Process finished with exit code 0
```

Multiplication tables using while loop

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The main area shows a file named "demo.py" with the following code:

```
num = 21
counter = 1
print("The Multiplication Table of: ", num)
while counter <= 10:
    ans = num * counter
    print(num, 'x', counter, '=', ans)
    counter += 1
```

The "Run" tab is selected, showing the output of running the script "demo". The output window displays:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
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
```

Append

A screenshot of the PyCharm IDE interface. The top bar shows the project name "PythonProject" and various tool icons. The main area displays a Python file named "demo.py" with the following code:

```
list_ = [3, 5, 1, 4, 6]
squares = []
while list_:
    squares.append((list_.pop())**2)
print(squares)
```

The bottom panel shows the "Run" tab with the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\Pycharm[36, 16, 1, 25, 9]
Process finished with exit code 0
```

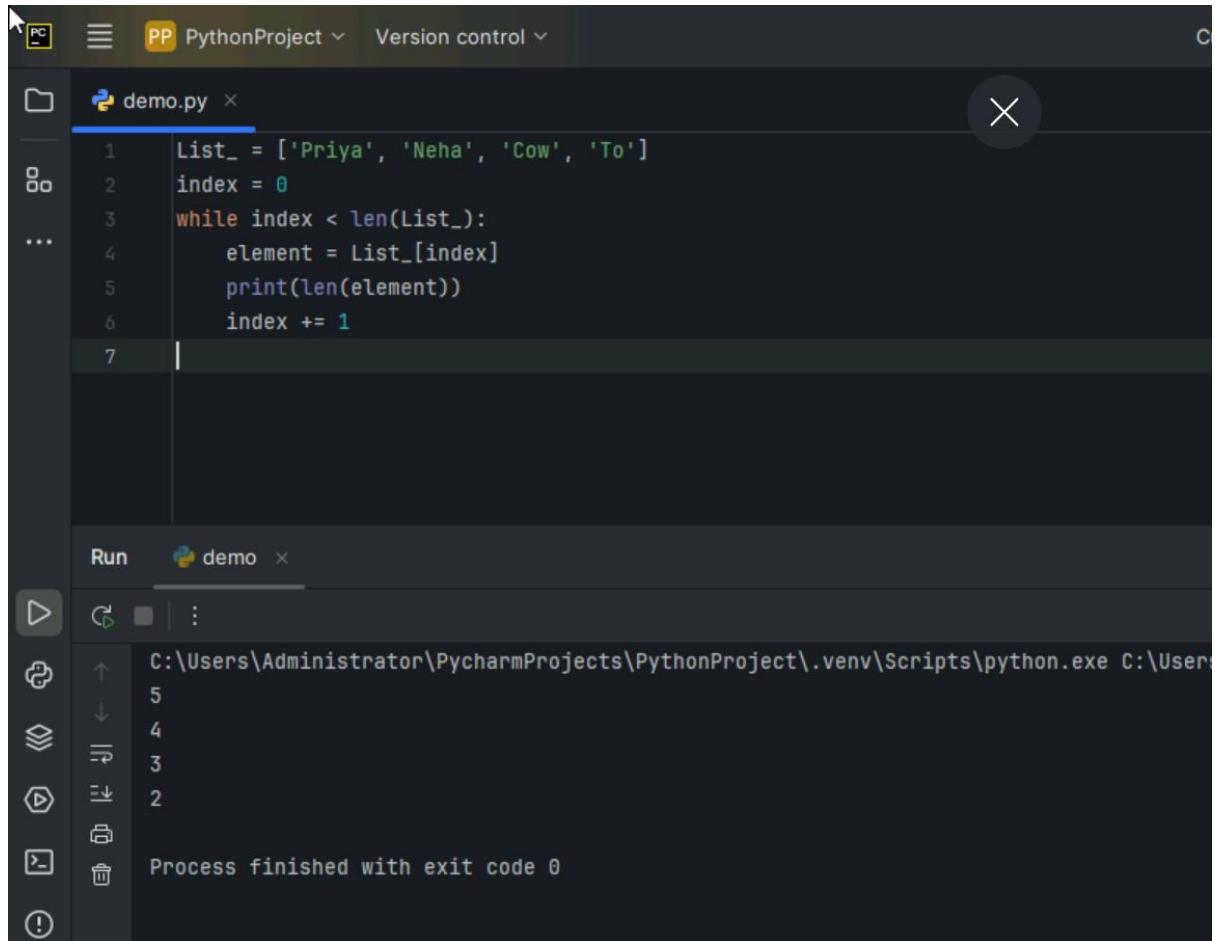
Even number check using while loop

A screenshot of the PyCharm IDE interface. The top bar shows the project name "PythonProject" and various tool icons. The main area displays a Python file named "demo.py" with the following code:

```
list_ = [3, 4, 8, 10, 34, 45, 67, 80]
index = 0
while index < len(list_):
    element = list_[index]
    if element % 2 == 0:
        print('It is an even number')
    else:
        print('It is an odd number')
    index += 1
```

The bottom panel shows the "Run" tab with the output of running the script, alternating between "It is an even number" and "It is an odd number" for each element in the list.

Letter check



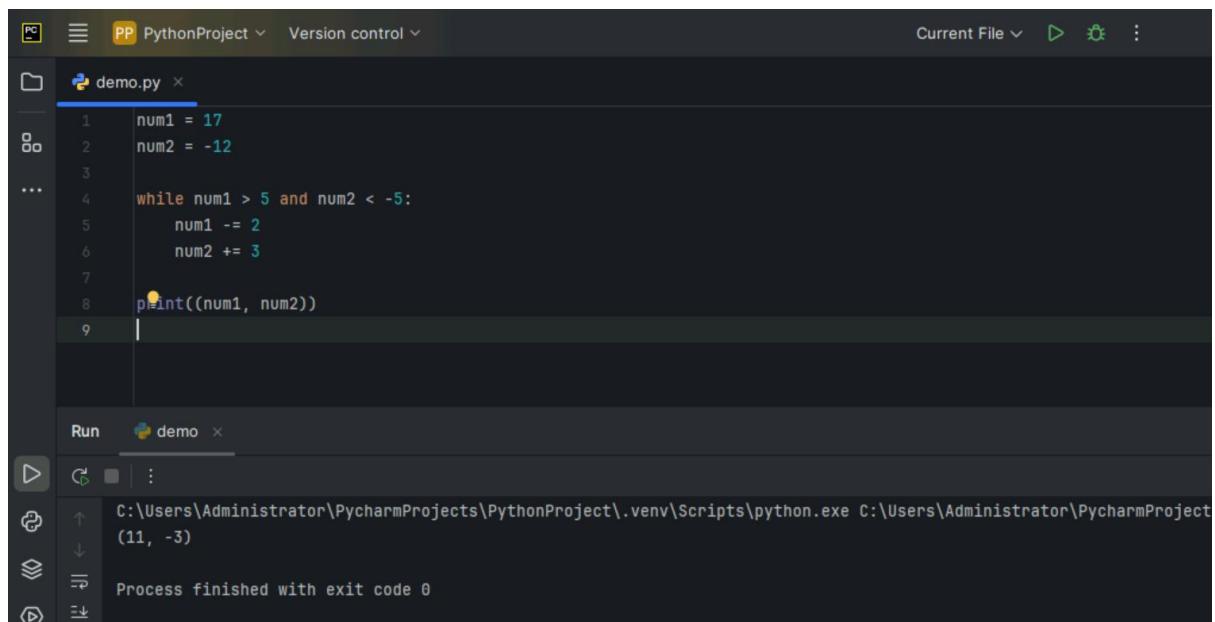
The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python script:

```
List_ = ['Priya', 'Neha', 'Cow', 'To']
index = 0
while index < len(List_):
    element = List_[index]
    print(len(element))
    index += 1
```

The run output window at the bottom shows the following results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
5
4
3
2
Process finished with exit code 0
```

And condition



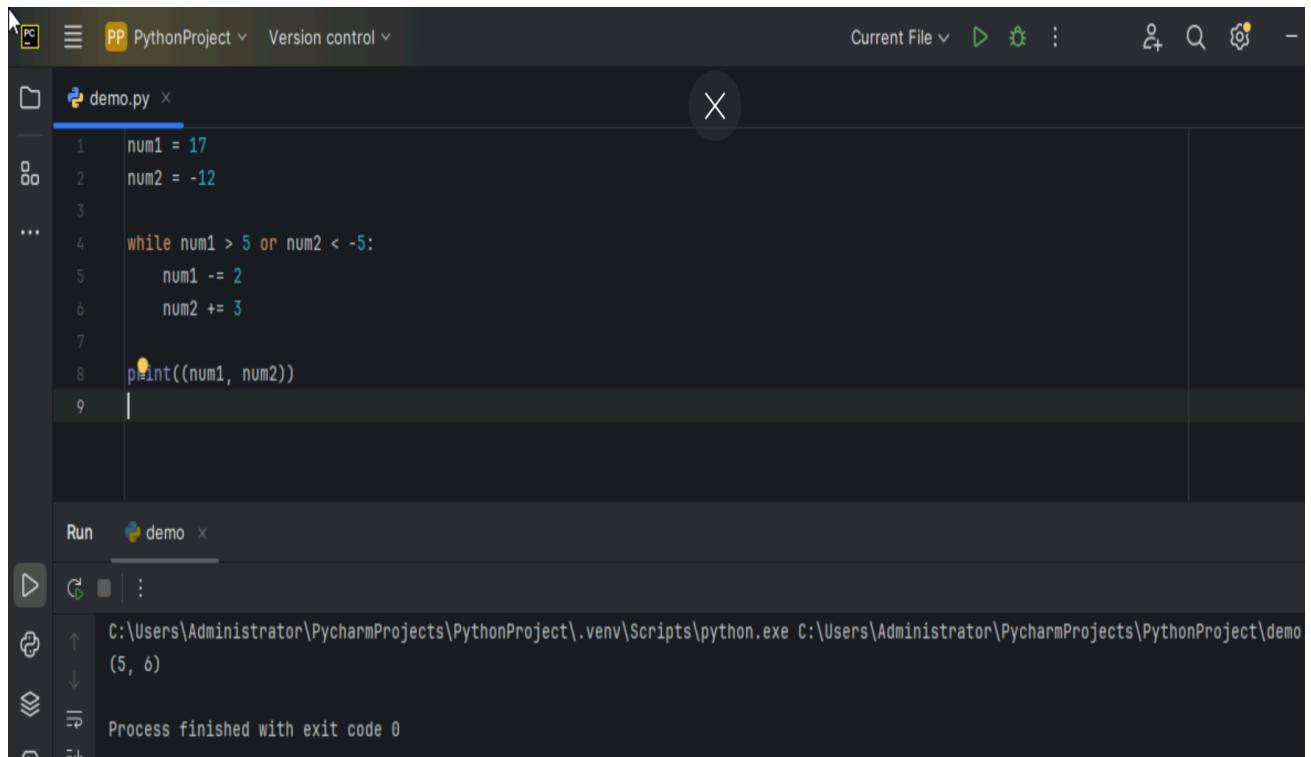
The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python script:

```
num1 = 17
num2 = -12
while num1 > 5 and num2 < -5:
    num1 -= 2
    num2 += 3
print((num1, num2))
```

The run output window at the bottom shows the following results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
(11, -3)
Process finished with exit code 0
```

Or condition



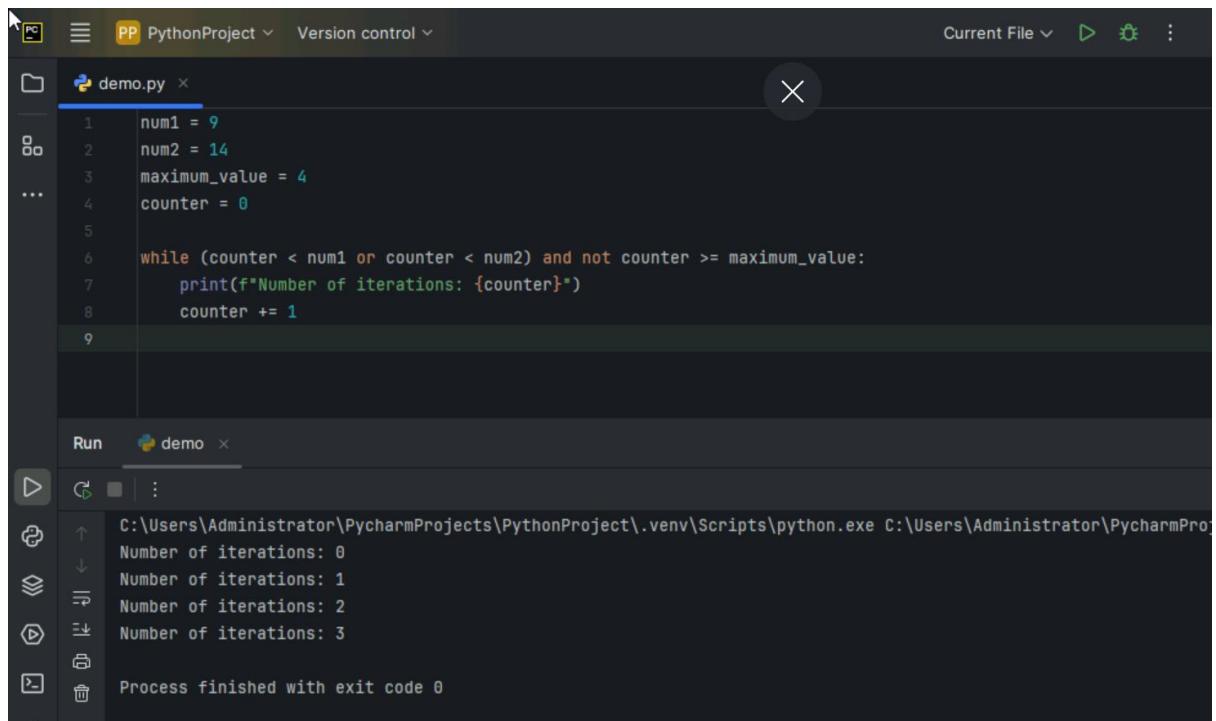
The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The main editor window contains the following code:

```
1 num1 = 17
2 num2 = -12
3
...
4 while num1 > 5 or num2 < -5:
5     num1 -= 2
6     num2 += 3
7
8 print((num1, num2))
9
```

The run tab at the bottom shows the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo
(5, 6)
Process finished with exit code 0
```

Multiple logical operators in while loop



The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The main editor window contains the following code:

```
1 num1 = 9
2 num2 = 14
3 maximum_value = 4
4 counter = 0
5
6 while (counter < num1 or counter < num2) and not counter >= maximum_value:
7     print(f"Number of iterations: {counter}")
8     counter += 1
9
```

The run tab at the bottom shows the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmPro
Number of iterations: 0
Number of iterations: 1
Number of iterations: 2
Number of iterations: 3
Process finished with exit code 0
```

Continue statement

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The left sidebar shows a project structure with a file named "demo.py". The code editor contains the following Python script:

```
for string in "While Loops":  
    if string == "o" or string == "i" or string == "e":  
        continue  
    print('Current Letter:', string)
```

The run output window at the bottom shows the execution results:

```
Current Letter: W  
Current Letter: h  
Current Letter: l  
Current Letter: L  
Current Letter: p  
Current Letter: s
```

The status bar at the bottom right indicates "Process finished with exit code 0".

Break Statement

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The left sidebar shows a project structure with a file named "demo.py". The code editor contains the following Python script:

```
for string in "Python Loops":  
    if string == 'n':  
        break  
    print('Current Letter: ', string)
```

The run output window at the bottom shows the execution results:

```
Current Letter: P  
Current Letter: y  
Current Letter: t  
Current Letter: h  
Current Letter: o
```

The status bar at the bottom right indicates "Process finished with exit code 0".

Pass

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python code:

```
for a_string in "Python Loops":  
    pass  
string = a_string  
print('The Last Letter of given string is:', string)
```

The run output window at the bottom shows the execution results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject\demo.py  
The Last Letter of given string is: s  
Process finished with exit code 0
```

Break example

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python code:

```
my_list = [1, 2, 3, 4]  
count = 1  
for item in my_list:  
    if item == 4:  
        print("Item matched")  
        count += 1  
        break  
print("Found at location", count)
```

The run output window at the bottom shows the execution results:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py  
Item matched  
Found at location 2  
Process finished with exit code 0
```

Break example

The screenshot shows the PyCharm IDE interface. The top bar displays "PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The code editor contains the following Python script:

```
my_str = "python"
for char in my_str:
    if char == 'o':
        break
    print(char)
```

The run tool window at the bottom shows the output of running the script:

```
Run demo
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject\demo.py
p
y
t
h
Process finished with exit code 0
```

Printing 2 table

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
main.py
```

```
1 n = 2
2 while True:
3     i = 1
4     while i <= 10:
5         print("%d X %d = %d\n" % (n, i, n * i))
6         i += 1
7     choice = int(input("Do you want to continue printing the table?
8         Press 0 for no: "))
9     if choice == 0:
10        print("Exiting the program...")
11        break
12    n += 1
13 print("Program finished successfully.")
```

To the right of the code, the output is displayed row-by-row:

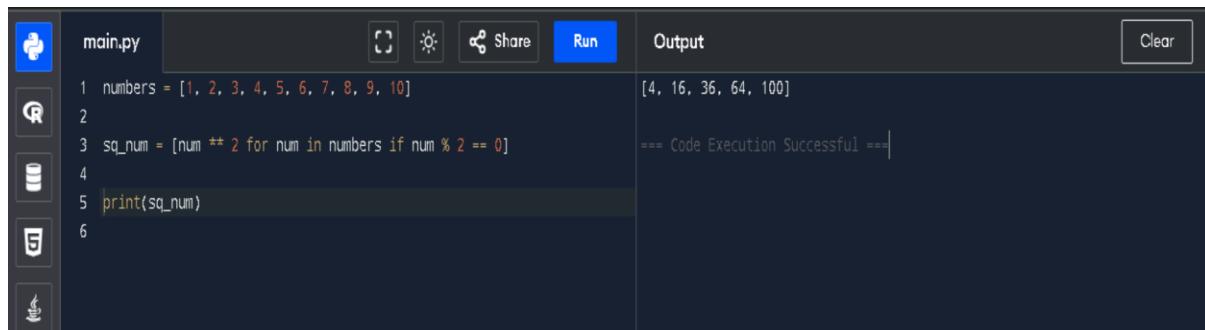
	Output
1	2 X 1 = 2
2	2 X 2 = 4
3	2 X 3 = 6
4	2 X 4 = 8
5	2 X 5 = 10
6	2 X 6 = 12
7	2 X 7 = 14
8	2 X 8 = 16
9	2 X 9 = 18
10	2 X 10 = 20

Continue



```
main.py
1 for iterator in range(10, 21):
2     if iterator == 15:
3         continue
4     print(iterator)
5
10
11
12
13
14
15
16
17
18
19
20
```

Square of numbers



```
main.py
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 sq_num = [num ** 2 for num in numbers if num % 2 == 0]
4
5 print(sq_num)
6
[4, 16, 36, 64, 100]
== Code Execution Successful ==
```

Creating basic strings

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with various icons: a blue square with a white plus sign, a QR code, a clipboard, a document, a gear, a coffee cup, a circular arrow, and two circular arrows. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 str1 = 'Hello Python'
2 print(str1)
3
4 str2 = "Hello Python"
5 print(str2)
6
7 str3 = '''Triple quotes are generally used for
8 representing multiline strings or
9 docstrings'''
10 print(str3)
11
```

The 'Output' tab shows the results of the code execution:

```
Hello Python
Hello Python
Triple quotes are generally used for
representing multiline strings or
docstrings
*** Code Execution Successful ***
```

String operations

The screenshot shows a Python code editor interface with a dark theme. The sidebar icons are the same as the previous screenshot. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
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 'Output' tab shows the results of the code execution, followed by an error message:

```
H
E
L
L
O
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 7, in <module>
    IndexError: string index out of range
*** Code Exited With Errors ***
```

String range

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
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])
7
```

The output pane shows the results of the code execution:

```
JAVATPOINT
AVAT
VA
JAV
TPO
== Code Execution Successful ==
```

String ranges

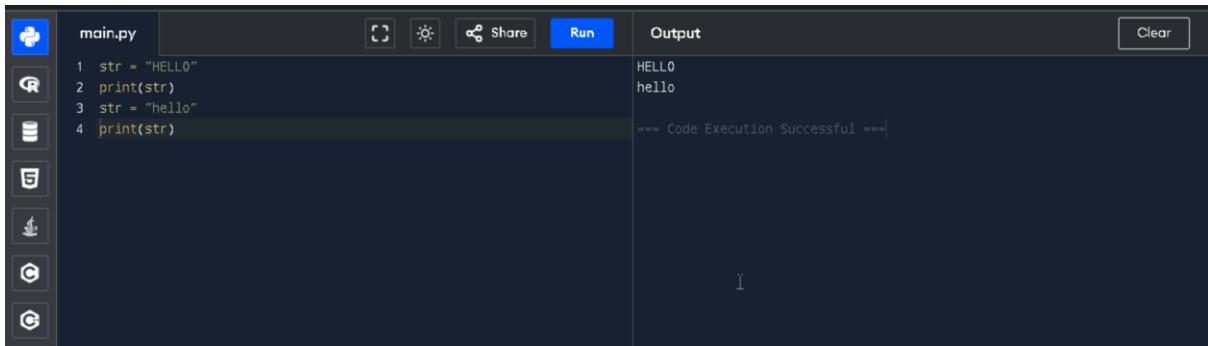
The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
1 str = 'JAVATPOINT'
2 print(str[-1])
3 print(str[-3])
4 print(str[-2])
5 print(str[-4:-1])
6 print(str[-7:-2])
7 print(str[::-1])
8 print(str[-12])
9
```

The output pane shows the results of the code execution, including an error message:

```
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 8, in <module>
    IndexError: string index out of range
== Code Exited With Errors ==
```

Reassigning strings



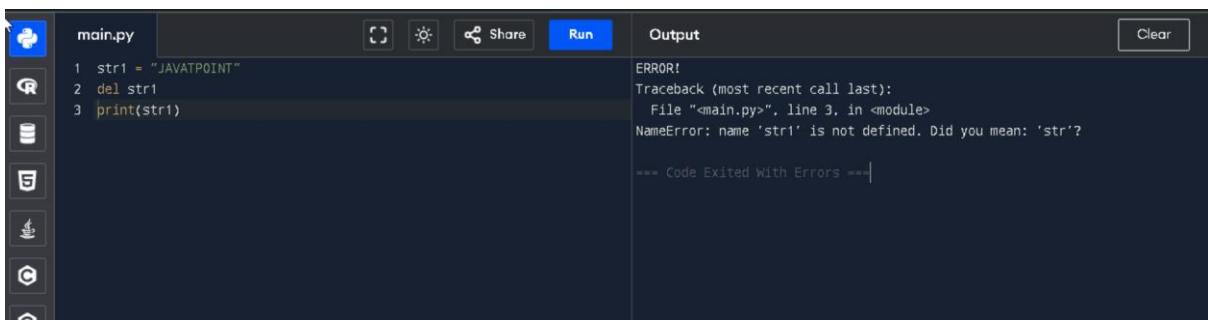
The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
1 str = "HELLO"
2 print(str)
3 str = "hello"
4 print(str)
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the code execution:

```
HELLO
hello
--- Code Execution Successful ---
```

Reassigning strings



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

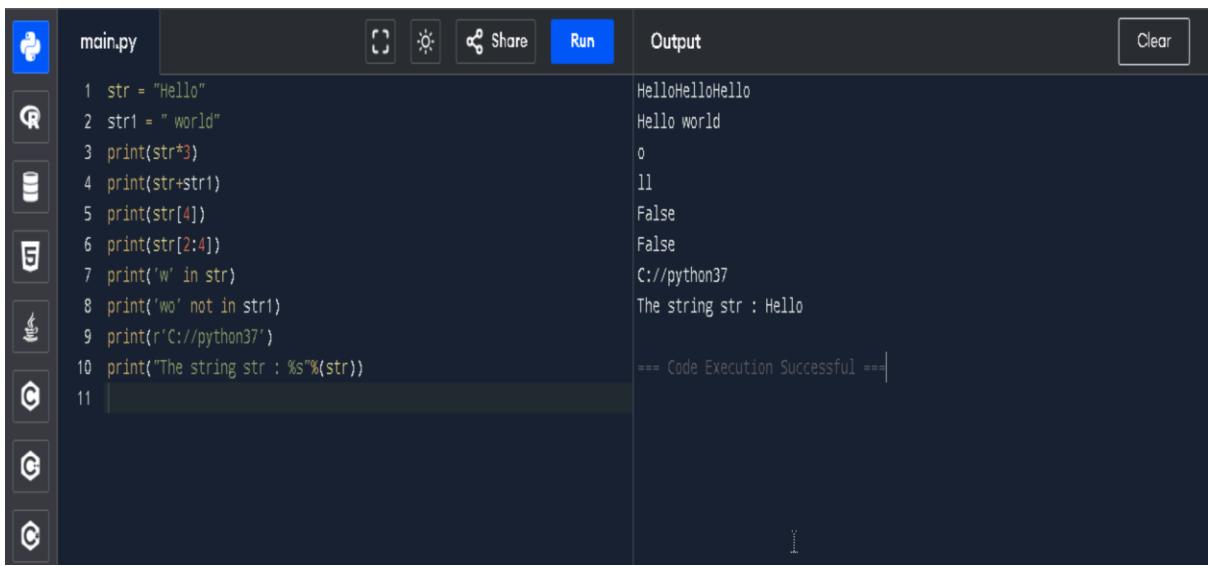
```
1 stri = "JAVATPOINT"
2 del stri
3 print(stri)
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the code execution:

```
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 3, in <module>
    NameError: name 'stri' is not defined. Did you mean: 'str'?

--- Code Exited With Errors ---
```

String formatting



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
1 str = "Hello"
2 stri = " world"
3 print(str*3)
4 print(str+stri)
5 print(str[4])
6 print(str[2:4])
7 print('w' in str)
8 print('wo' not in stri)
9 print(r'C://python37')
10 print("The string str : %s"%str)
11 |
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the code execution:

```
HelloHelloHello
Hello world
o
ll
False
False
C://python37
The string str : Hello
--- Code Execution Successful ---
```

Multiple quotes

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with icons for file operations like Open, Save, and Run. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 print("C:\\\\Users\\\\DEVANSH SHARMA\\\\Python32\\\\Lib")
2 print("This is the \\n multiline quotes")
3 print("This is \\x48\\x45\\x58 representation")
```

The 'Output' tab shows the results of running the code:

```
C:\\\\Users\\\\DEVANSH SHARMA\\\\Python32\\\\Lib
This is the
multiline quotes
This is HEX representation

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

String formatting, format() method

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with icons for file operations like Open, Save, and Run. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 print("{} and {} both are the best friend".format("Devansh",
2 "Abhishek"))
2 print("{} and {} best players ".format("Virat", "Rohit"))
3 print("{}, {}, {}".format(a="James", b="Peter", c="Ricky"))
4
```

The 'Output' tab shows the results of running the code:

```
Devansh and Abhishek both are the best friend
Rohit and Virat best players
James,Peter,Ricky

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

Q

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with icons for file operations like Open, Save, and Run. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 Integer = 10
2 Float = 1.290
3 String = "Devansh"
4
5 print("Hi I am Integer ... My value is %d\nHi I am float ... My value
is %f\nHi I am string ... My value is %s" % (Integer, Float,
String))
6
```

The 'Output' tab shows the results of running the code:

```
Hi I am Integer ... My value is 10
Hi I am float ... My value is 1.290000
Hi I am string ... My value is Devansh

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

Lists and Tuples

```
main.py
1 list_ = [4, 5, 7, 1, 7]
2 tuple_ = (4, 1, 8, 3, 9)
3
4 print("List is:", list_)
5 print("Tuple is:", tuple_)
6
```

List is: [4, 5, 7, 1, 7]
Tuple is: (4, 1, 8, 3, 9)
== Code Execution Successful ==

Mutable and immutable tuple

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

['Python', 'Lists', 'Tuples', 'Mutable']
Tuples cannot be modified because they are immutable
== Code Execution Successful ==

dir

```
main.py
1 print( dir(tuple), end = ", " )
2
```

'__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
 '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
 '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__',
 '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
 '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', 'count', 'index'].
== Code Execution Successful ==