Python Task 3

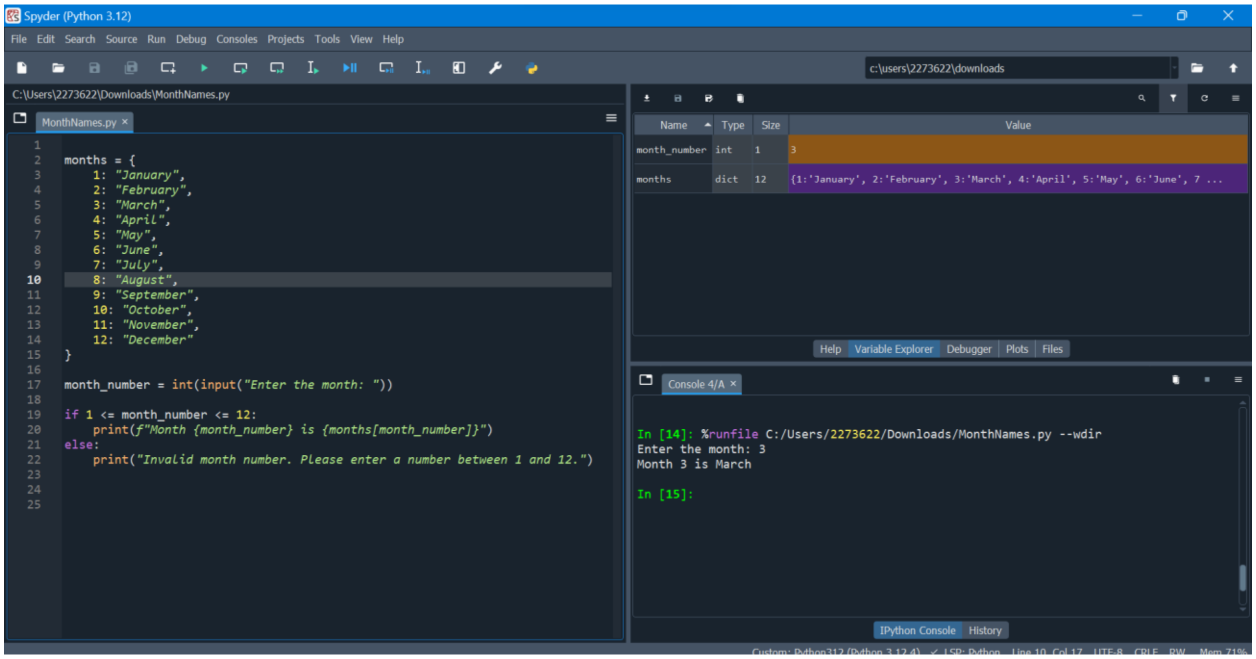
Exercise 1
Name your file: MonthNames.py
Write a program that reads an integer value between 1 and 12 from the user and prints output the corresponding month of the year.
An example run of the program (numbers in bold are typed in by the user)
Enter the month: 3
Month 3 is March

Exercise1: Month Conversion Program
Definition: A month conversion program is a simple application that takes an integer input from the user, representing a month of the year, and outputs the corresponding month name.
Purpose: The purpose of this program is to demonstrate basic input and output operations in programming, as well as the use of conditional statements to map numeric values to specific outputs.
Use Cases:

1. This program can be used in educational settings to teach beginners about basic programming concepts.
2. Useful in designing user interfaces where users need to select or input months.

Code 1:

```
months = {
    1: "January",
    2: "February",
    3: "March",
    4: "April",
    5: "May",
    6: "June",
    7: "July",
    8: "August",
    9: "September",
    10: "October",
    11: "November",
    12: "December"
}

month_number = int(input("Enter the month: "))

if 1 <= month_number <= 12:
    print(f"Month {month_number} is {months[month_number]}")
else:
    print("Invalid month number. Please enter a number between 1 and 12.")
```


Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\downloads
C:\Users\2273622\Downloads\MonthNames.py
MonthNames.py
Help Variable Explorer Debugger Plots Files
Console 4/A
IPython Console History

```python
months = {
    1: "January",
    2: "February",
    3: "March",
    4: "April",
    5: "May",
    6: "June",
    7: "July",
    8: "August",
    9: "September",
    10: "October",
    11: "November",
    12: "December"
}

month_number = int(input("Enter the month: "))

if 1 <= month_number <= 12:
    print(f"Month {month_number} is {months[month_number]}")
else:
    print("Invalid month number. Please enter a number between 1 and 12.")
```

Name	Type	Size	Value
month_number	int	1	3
months	dict	12	{1:'January', 2:'February', 3:'March', 4:'April', 5:'May', 6:'June', 7 ...

```
In [14]: %runfile C:/Users/2273622/Downloads/MonthNames.py --wdir
Enter the month: 3
Month 3 is March

In [15]:
```

Advantages:
A Python dictionary is a data structure that stores the value in **key: value** pairs. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be *immutable*.

Exercise 2
A certain cinema currently sells tickets for a full price of 6 pounds, but always sells tickets for half price to people who are less than 16 years old, and for a third of the price for people who are 60 years old or more.
An example run of the program (numbers in bold are typed in by the user)
Enter your age: 63
Your ticket costs £2.00

Definition: This program calculates the cost of a cinema ticket based on the age of the customer. The cinema offers discounted prices for children and seniors.

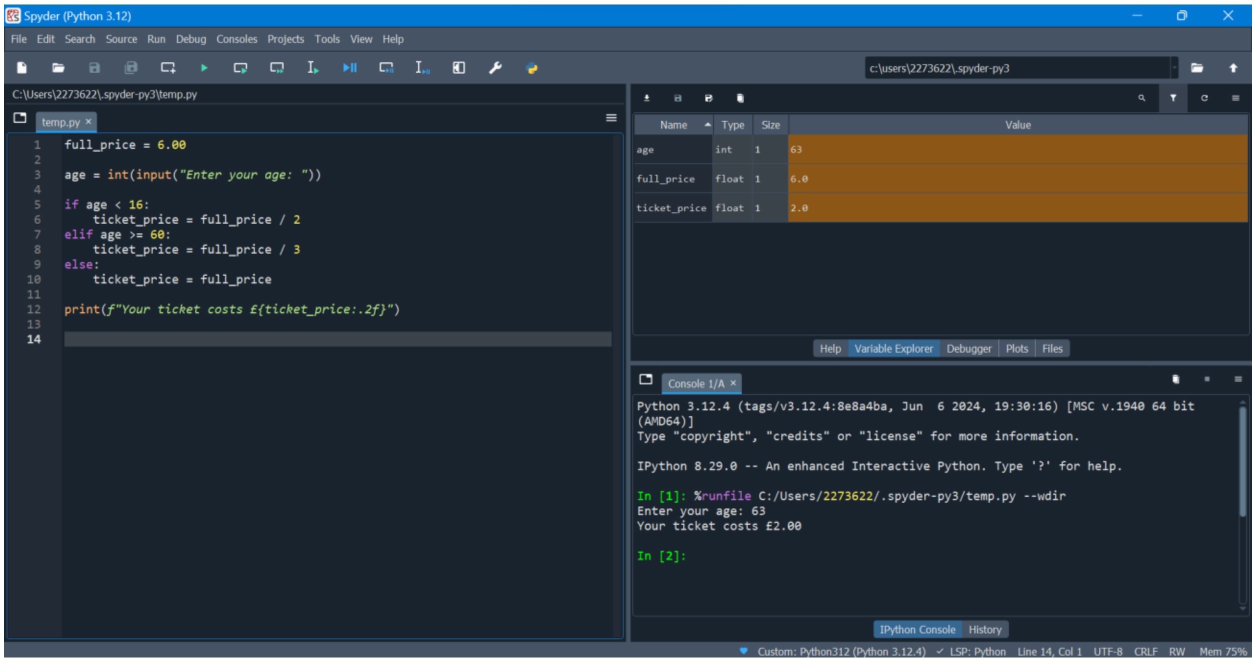
Purpose: The purpose of this program is to demonstrate how conditional statements can be used to apply different pricing rules based on user input. It also highlights the practical application of programming in real-world scenarios, such as ticket pricing.

Use Cases:

1. **Cinema Management:** This program can be used by cinema management to automate ticket pricing based on age.

Code2:

```
full_price = 6.00

age = int(input("Enter your age: "))

if age < 16:
    ticket_price = full_price / 2
elif age >= 60:
    ticket_price = full_price / 3
else:
    ticket_price = full_price

print(f"Your ticket costs £{ticket_price:.2f}")
```



```
full_price = 6.00

age = int(input("Enter your age: "))

if age < 16:
    ticket_price = full_price / 2
elif age >= 60:
    ticket_price = full_price / 3
else:
    ticket_price = full_price

print(f"Your ticket costs £{ticket_price:.2f}")
```

Name	Type	Size	Value
age	int	1	63
full_price	float	1	6.0
ticket_price	float	1	2.0

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %runfile C:/Users/2273622/.spyder-py3/temp.py --wdir
Enter your age: 63
Your ticket costs £2.00

In [2]:
```

Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\.spyder-py3
C:\Users\2273622\.spyder-py3\temp.py
temp.py
Help Variable Explorer Debugger Plots Files
Console 1/A
IPython Console History
Custom: Python312 (Python 3.12.4) LSP: Python Line 14, Col 1 UTF-8 CRLF RW Mem 75%

Advantages:
Conditional statements helps in decision making based on certain criteria.

Exercise 3

Name your file: BodyMassIndex.py
Write a program to calculate your BMI and give weight status. Body Mass Index (BMI) is an internationally used measurement to check if you are a healthy weight for your height.The metric BMI formula accepts weight in kilograms and height in meters:
BMI= weight(kg)/height2(m2)
BMI Weight Status Categories table

BMI range - kg/m2	Category
Below 18.5	Underweight
18.5 -24.9	Normal
25 - 29.9	Overweight
30 & Above	Obese

An example run of the program (numbers in bold are typed in by the user)

```
Enter your weight in (kg): 75
Enter your height in (m): 1.70
Your BMI is: 25.95
You are in the “overweight” range.
```

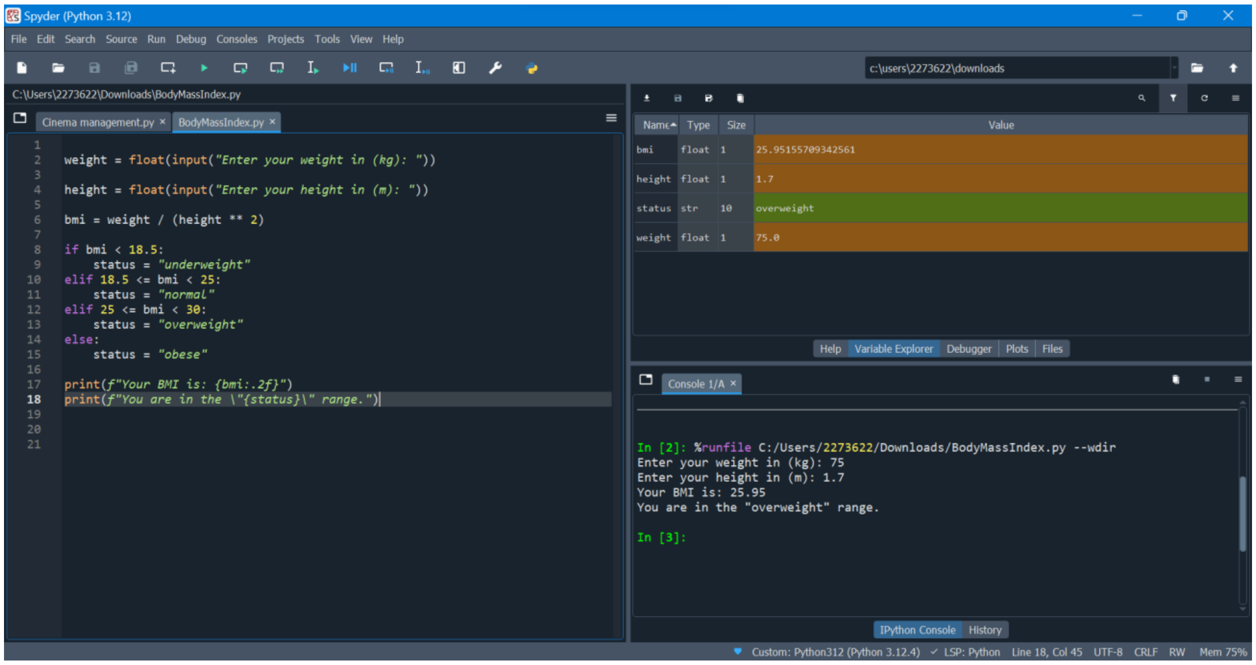
Definition: Body Mass Index (BMI) is a widely used measurement to determine if an individual has a healthy weight for their height. It is calculated using the formula:
BMI=height2(m2)weight (kg)
.
Purpose: The purpose of this program is to help individuals quickly and accurately calculate their BMI and understand their weight status based on standard categories. This can be useful for personal health assessments and for educational purposes in learning basic programming concepts.
Use Cases:

1. **Health and Fitness:** Individuals can use this program to monitor their weight status and make informed decisions about their health and fitness routines.
2. **Healthcare Applications:** It can be integrated into healthcare applications to provide quick BMI calculations for patients.

Code 3:

```
weight = float(input("Enter your weight in (kg): "))

height = float(input("Enter your height in (m): "))

bmi = weight / (height ** 2)

if bmi < 18.5:
    status = "underweight"
```

```
elif 18.5 <= bmi < 25:
    status = "normal"
elif 25 <= bmi < 30:
    status = "overweight"
else:
    status = "obese"

print(f"Your BMI is: {bmi:.2f}")
print(f"You are in the \"{status}\" range.")
```


Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\downloads
C:\Users\2273622\Downloads\BodyMassIndex.py
Cinema management.py
BodyMassIndex.py
Help Variable Explorer Debugger Plots Files
Console 1/A
IPython Console History
Custom: Python312 (Python 3.12.4) LSP: Python Line 18, Col 45 UTF-8 CRLF RW Mem 75%

```

weight = float(input("Enter your weight in (kg): "))

height = float(input("Enter your height in (m): "))

bmi = weight / (height ** 2)

if bmi < 18.5:
    status = "underweight"
elif 18.5 <= bmi < 25:
    status = "normal"
elif 25 <= bmi < 30:
    status = "overweight"
else:
    status = "obese"

print(f"Your BMI is: {bmi:.2f}")
print(f"You are in the \"{status}\" range.")
```

Name	Type	Size	Value
bmi	float	1	25.95155709342561
height	float	1	1.7
status	str	10	overweight
weight	float	1	75.0

```
In [2]: %runfile C:/Users/2273622/Downloads/BodyMassIndex.py --wdir
Enter your weight in (kg): 75
Enter your height in (m): 1.7
Your BMI is: 25.95
You are in the "overweight" range.

In [3]:
```

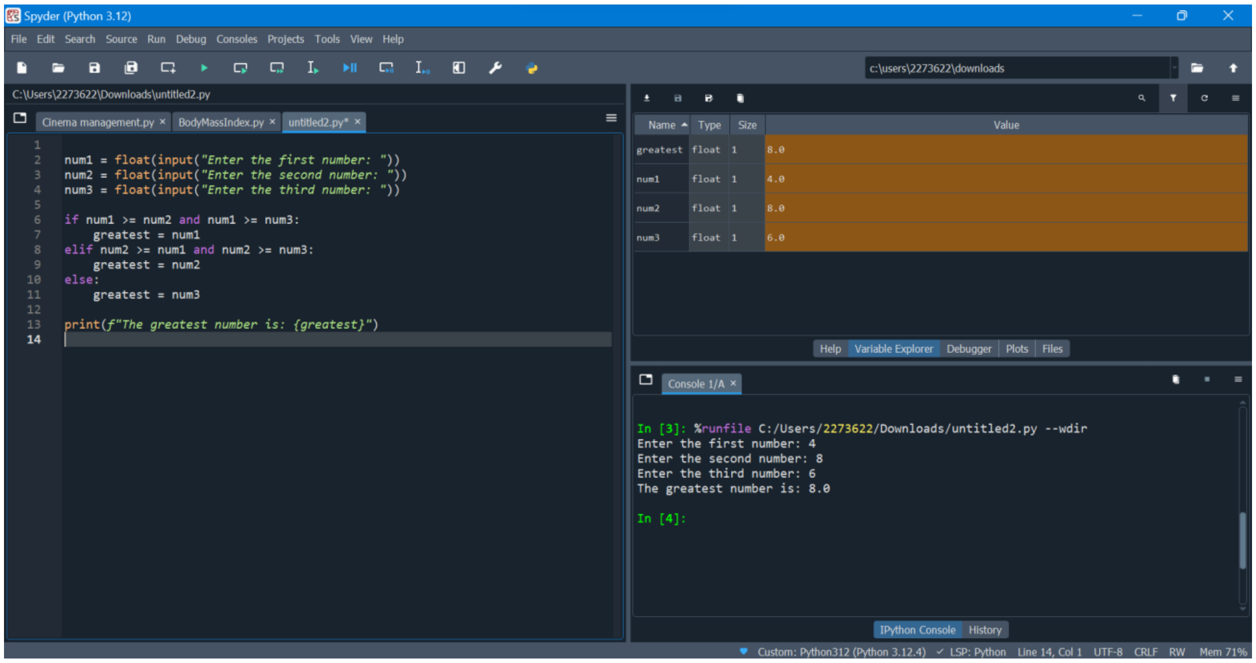
Exercise 4
Write a Python program to receive 3 numbers from the user and print the greatest among them.

Definition: This program takes three numbers as input from the user and determines which one is the greatest. It uses basic comparison operations to achieve this.
Purpose: The purpose of this program is to demonstrate how to handle multiple inputs and use conditional statements to compare values. This is a fundamental concept in programming that is widely applicable in various scenarios.
Use Cases:

1. **Data Analysis:** It can be used in data analysis to find the maximum value in a set of numbers.
2. **Decision Making:** Helps in scenarios where decisions need to be made based on the largest value among several options.

Code4:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

if num1 >= num2 and num1 >= num3:
    greatest = num1
elif num2 >= num1 and num2 >= num3:
    greatest = num2
else:
    greatest = num3

print(f"The greatest number is: {greatest}")
```


Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\downloads
C:\Users\2273622\Downloads\untitled2.py
Cinema management.py
BodyMassIndex.py
untitled2.py*
Help
Variable Explorer
Debugger
Plots
Files
Console 1/A
IPython Console
History
Custom: Python312 (Python 3.12.4) LSP: Python Line 14, Col 1 UTF-8 CRLF RW Mem 71%

```

num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

if num1 >= num2 and num1 >= num3:
    greatest = num1
elif num2 >= num1 and num2 >= num3:
    greatest = num2
else:
    greatest = num3

print(f"The greatest number is: {greatest}")

```

Name	Type	Size	Value
greatest	float	1	8.0
num1	float	1	4.0
num2	float	1	8.0
num3	float	1	6.0

```
In [3]: %runfile C:/Users/2273622/Downloads/untitled2.py --wdir
Enter the first number: 4
Enter the second number: 8
Enter the third number: 6
The greatest number is: 8.0

In [4]:
```

Exercise 5
Find the factorial of a given number using loops(note the number is received from the user)

Definition: The factorial of a non-negative integer (n) is the product of all positive integers less than or equal to (n). It is denoted by (n!). For example, the factorial of 5 (denoted as (5!)) is (5 \times 4 \times 3 \times 2 \times 1 = 120).

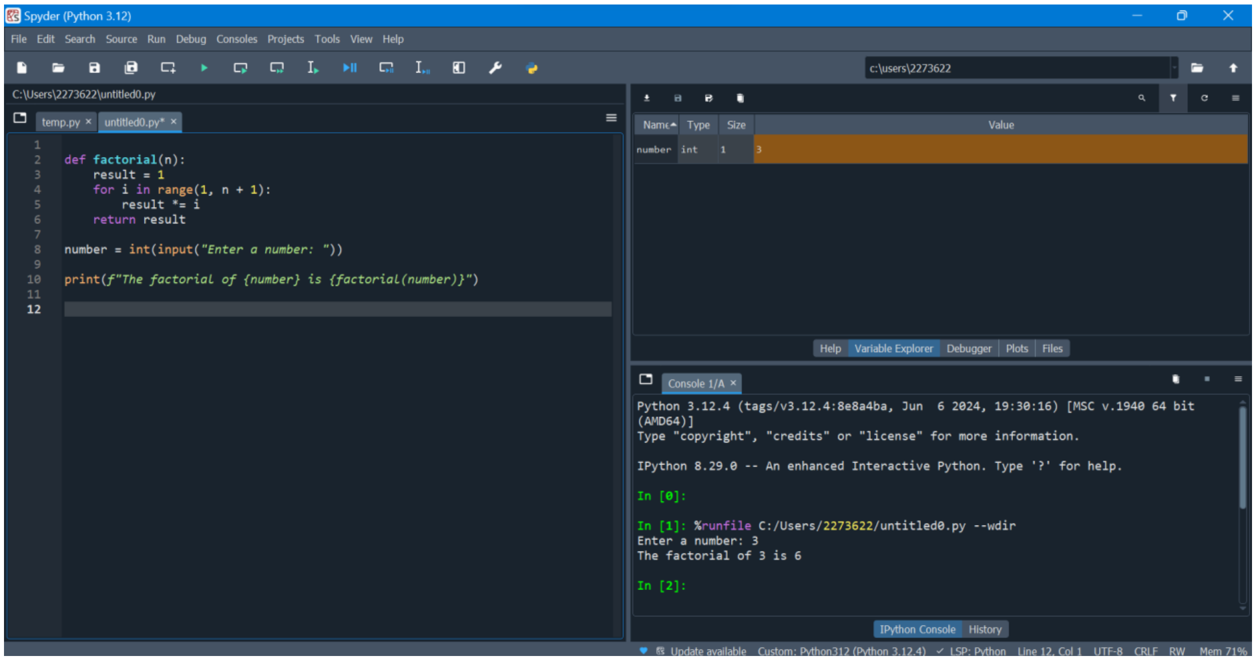
Purpose: Factorials are used in various fields of mathematics and computer science, including combinatorics, algebra, and calculus. They are essential in calculating permutations and combinations, which are fundamental in probability and statistics.

Use Cases:

1. Calculating the number of ways to arrange or select items.
2. Determining probabilities in complex scenarios.

Code5:

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

number = int(input("Enter a number: "))

print(f"The factorial of {number} is {factorial(number)}")
```



C:\Users\2273622\untitled0.py

```

def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

number = int(input("Enter a number: "))

print(f"The factorial of {number} is {factorial(number)}")

```

Name	Type	Size	Value
number	int	1	3

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [0]:

In [1]: %runfile C:/Users/2273622/untitled0.py --wdir
Enter a number: 3
The factorial of 3 is 6

In [2]:
```

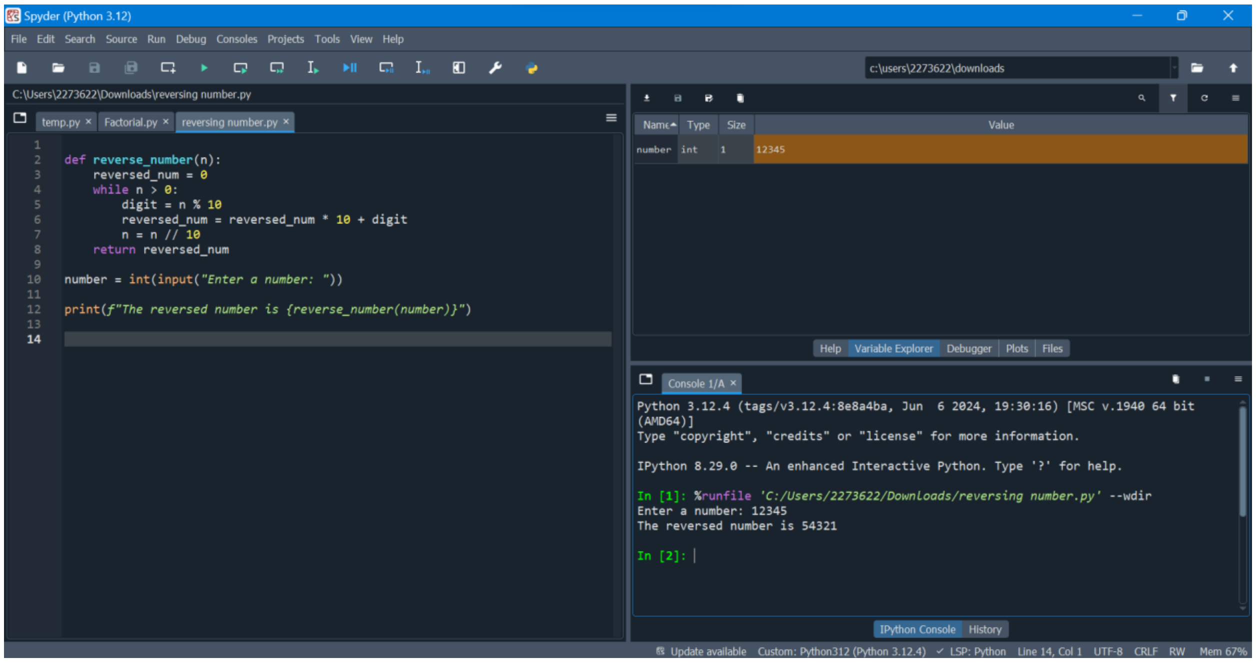
Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622
temp.py
untitled0.py*
Help Variable Explorer Debugger Plots Files
Console 1/A
IPython Console History
Update available Custom: Python312 (Python 3.12.4) LSP: Python Line 12, Col 1 UTF-8 CRLF RW Mem 71%

Exercise 6
Reverse a number using while loop

Definition: Reversing a number means rearranging its digits in the opposite order. For example, reversing 12345 results in 54321.

Purpose: Reversing numbers is a common task in programming that helps in understanding loops and control structures. It is also useful in various applications such as palindromes, data encryption, and digital signal processing.

Code 6:
```
def reverse_number(n):
    reversed_num = 0
    while n > 0:
        digit = n % 10
        reversed_num = reversed_num * 10 + digit
        n = n // 10
    return reversed_num

number = int(input("Enter a number: "))

print(f"The reversed number is {reverse_number(number)}")
```



```
def reverse_number(n):
    reversed_num = 0
    while n > 0:
        digit = n % 10
        reversed_num = reversed_num * 10 + digit
        n = n // 10
    return reversed_num

number = int(input("Enter a number: "))

print(f"The reversed number is {reverse_number(number)}")
```

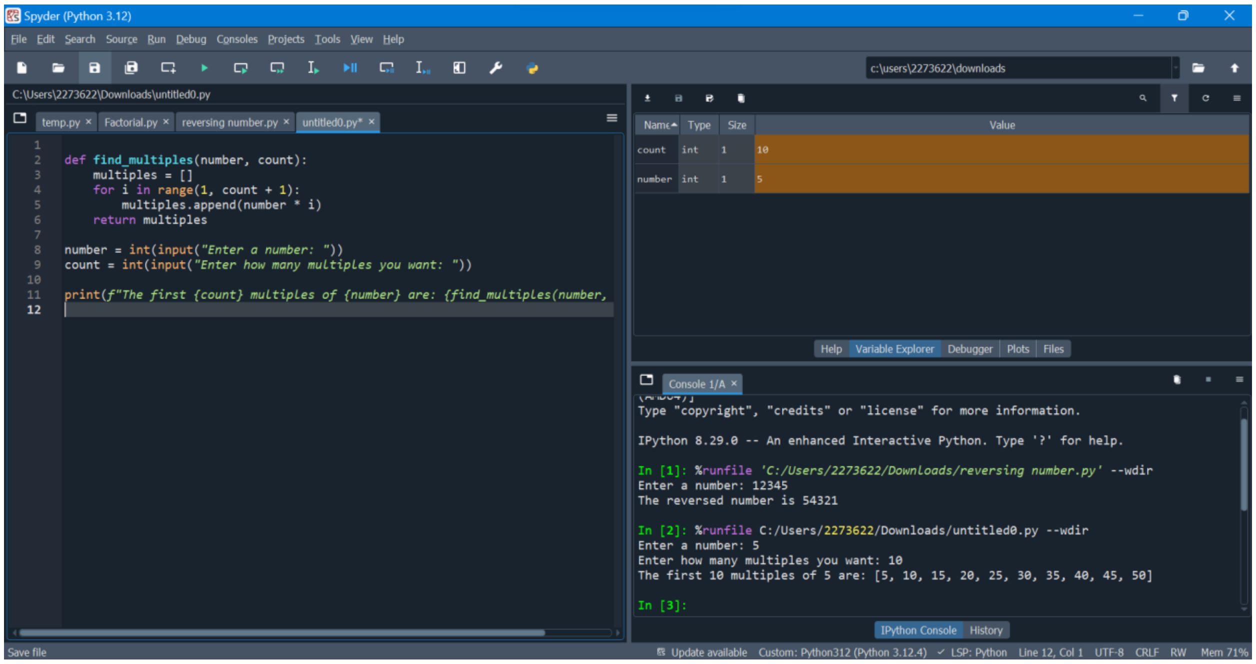
Name	Type	Size	Value
number	int	1	12345

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %runfile 'C:/Users/2273622/Downloads/reversing number.py' --wdir
Enter a number: 12345
The reversed number is 54321

In [2]:
```

Exercise 7
Finding the multiples of a number using loop

Definition: A multiple of a number is the product of that number and an integer. For example, the multiples of 3 are 3, 6, 9, 12, etc.
Purpose: Finding multiples is a fundamental task in mathematics and programming. It helps in understanding loops and iteration, and is useful in various applications such as generating sequences, solving problems in number theory, and creating patterns.

Code:
```
def find_multiples(number, count):
    multiples = []
    for i in range(1, count + 1):
        multiples.append(number * i)
    return multiples

number = int(input("Enter a number: "))
count = int(input("Enter how many multiples you want: "))

print(f"The first {count} multiples of {number} are: {find_multiples(number, count)}")
```



```
def find_multiples(number, count):
    multiples = []
    for i in range(1, count + 1):
        multiples.append(number * i)
    return multiples

number = int(input("Enter a number: "))
count = int(input("Enter how many multiples you want: "))

print(f"The first {count} multiples of {number} are: {find_multiples(number,
```

Name	Type	Size	Value
count	int	1	10
number	int	1	5

```
Type "copyright", "credits" or "license" for more information.

IPython 8.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %runfile 'C:/Users/2273622/Downloads/reversing number.py' --wdir
Enter a number: 12345
The reversed number is 54321

In [2]: %runfile C:/Users/2273622/Downloads/untitled0.py --wdir
Enter a number: 5
Enter how many multiples you want: 10
The first 10 multiples of 5 are: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

In [3]:
```

Exercise 8

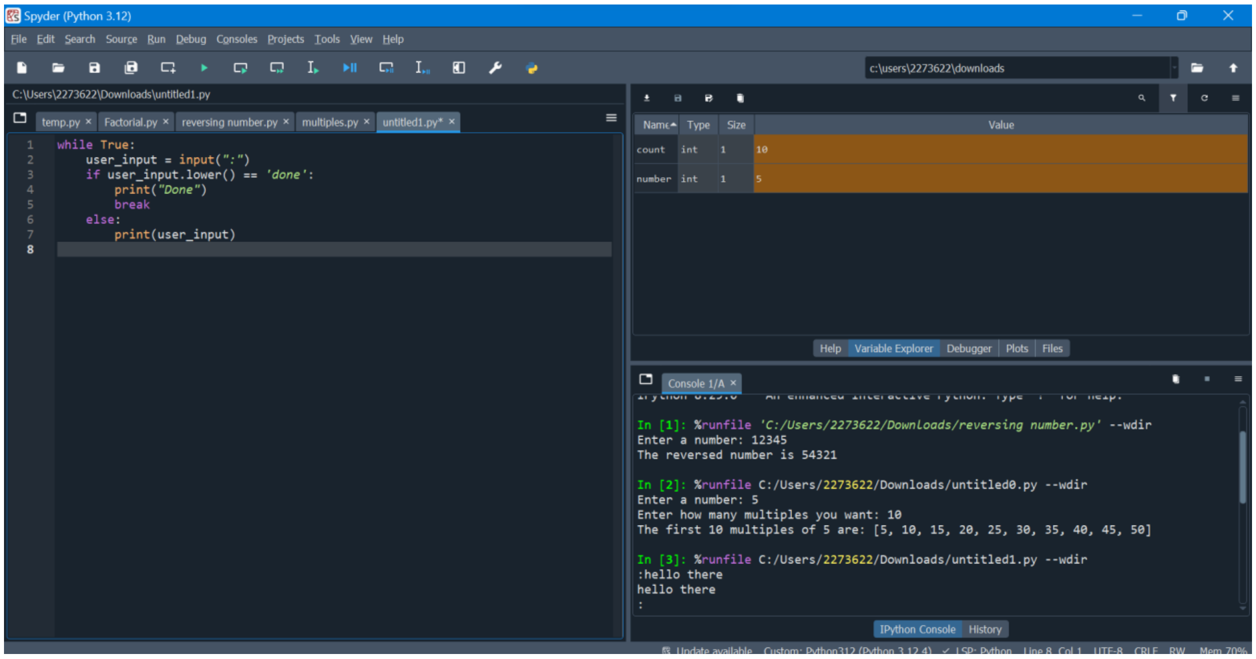
Write a program to print the inputted value as it is and break the loop if the value is 'done'.
Example run of the program

```
:hello there
hello there
:finished
finished
:done
Done
```

Definition: Loop control statements change the execution from its normal sequence. When the break statement is encountered inside a loop, the loop is immediately terminated, and the program control resumes at the next statement following the loop.
Purpose: Using loop control statements like break helps in managing the flow of loops, especially when certain conditions are met. This is useful in scenarios where you need to exit a loop based on user input or specific conditions.

Code:

```python
while True:
    user_input = input(":")
    if user_input.lower() == 'done':
        print("Done")
        break
    else:
        print(user_input)
```



Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\downloads
C:\Users\2273622\Downloads\untitled1.py
temp.py × Factorial.py × reversing number.py × multiples.py × untitled1.py* ×
Help Variable Explorer Debugger Plots Files
Console 1/A ×
IPython Console History

```python
while True:
    user_input = input(":")
    if user_input.lower() == 'done':
        print("Done")
        break
    else:
        print(user_input)
```

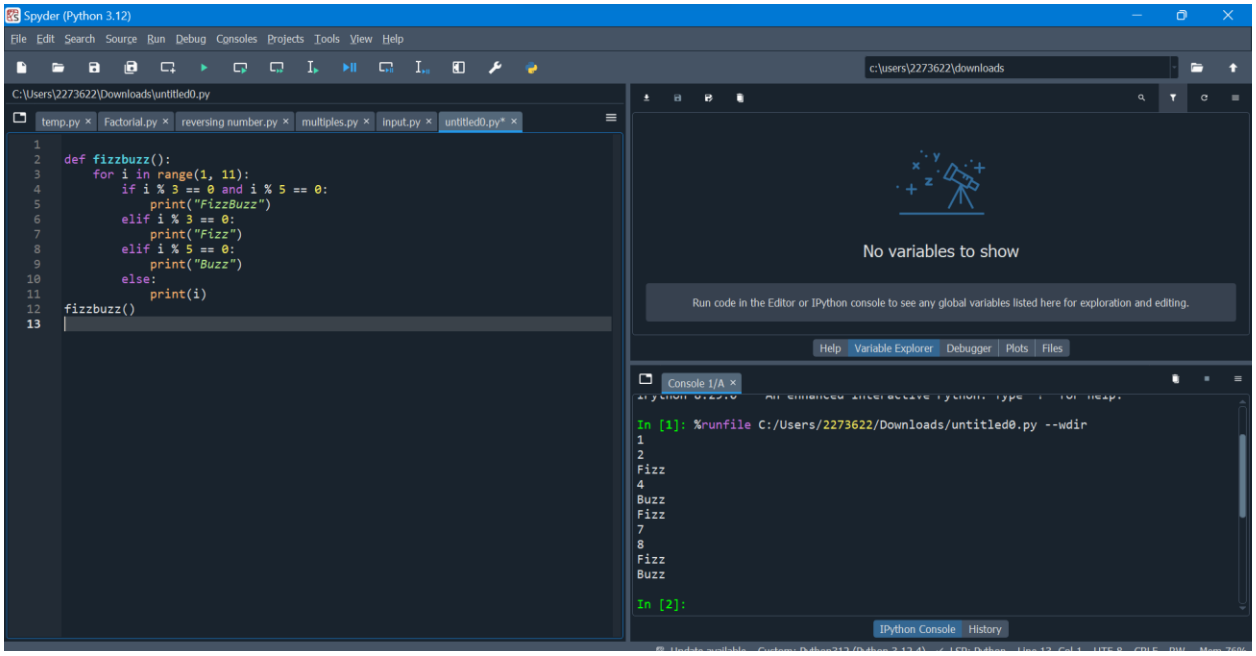
Name	Type	Size	Value
count	int	1	10
number	int	1	5

```
In [1]: %runfile 'C:/Users/2273622/Downloads/reversing number.py' --wdir
Enter a number: 12345
The reversed number is 54321

In [2]: %runfile C:/Users/2273622/Downloads/untitled0.py --wdir
Enter a number: 5
Enter how many multiples you want: 10
The first 10 multiples of 5 are: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

In [3]: %runfile C:/Users/2273622/Downloads/untitled1.py --wdir
:hello there
hello there
:
```

Exercise 9

Write a program that prints the numbers from 1 to 10. But for multiples of three print "Fizz" instead of the number and for the multiple of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"

Definition: FizzBuzz is a common programming exercise that involves printing numbers from 1 to a specified limit with substitutions for multiples of certain numbers. Specifically, for multiples of three, “Fizz” is printed instead of the number, for multiples of five, “Buzz” is printed, and for multiples of both three and five, “FizzBuzz” is printed.
Purpose: FizzBuzz is often used to teach basic programming concepts such as loops, conditionals, and modulo operations. It is also a popular interview question to assess a candidate’s problem-solving skills and understanding of control flow.

Code:
```
def fizzbuzz():
    for i in range(1, 11):
        if i % 3 == 0 and i % 5 == 0:
            print("FizzBuzz")
        elif i % 3 == 0:
            print("Fizz")
        elif i % 5 == 0:
            print("Buzz")
        else:
            print(i)
fizzbuzz()
```

Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
c:\users\2273622\downloads
C:\Users\2273622\Downloads\untitled0.py
temp.py × Factorial.py × reversing number.py × multiples.py × input.py × untitled0.py* ×
Help Variable Explorer Debugger Plots Files
Console 1/A ×
IPython Console History

```

def fizzbuzz():
    for i in range(1, 11):
        if i % 3 == 0 and i % 5 == 0:
            print("FizzBuzz")
        elif i % 3 == 0:
            print("Fizz")
        elif i % 5 == 0:
            print("Buzz")
        else:
            print(i)
fizzbuzz()

```

No variables to show

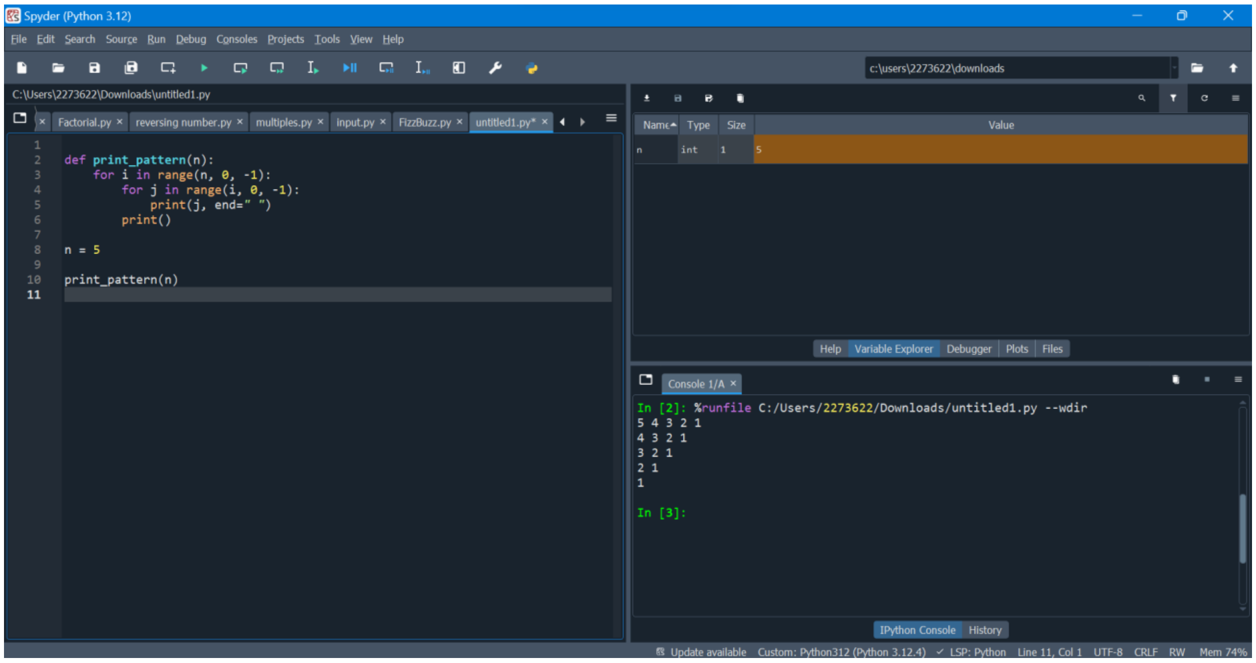
Run code in the Editor or IPython console to see any global variables listed here for exploration and editing.

```
In [1]: %runfile C:/Users/2273622/Downloads/untitled0.py --wdir
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz

In [2]:
```

Exercise 10
Write a program to print the following pattern:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Definition: Pattern printing involves displaying numbers, characters, or symbols in a specific format or sequence. It is a common exercise in programming to practice loops and nested loops.
Purpose: Pattern printing helps in understanding the use of loops and nested loops, which are fundamental concepts in programming. It also enhances logical thinking and problem-solving skills.

Code:
```
def print_pattern(n):
    for i in range(n, 0, -1):
        for j in range(i, 0, -1):
            print(j, end=" ")
        print()

n = 5

print_pattern(n)
```


```

def print_pattern(n):
    for i in range(n, 0, -1):
        for j in range(i, 0, -1):
            print(j, end=" ")
        print()

n = 5

print_pattern(n)

```

Name	Type	Size	Value
n	int	1	5

```
In [2]: %runfile C:/Users/2273622/Downloads/untitled1.py --wdir
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1

In [3]:
```