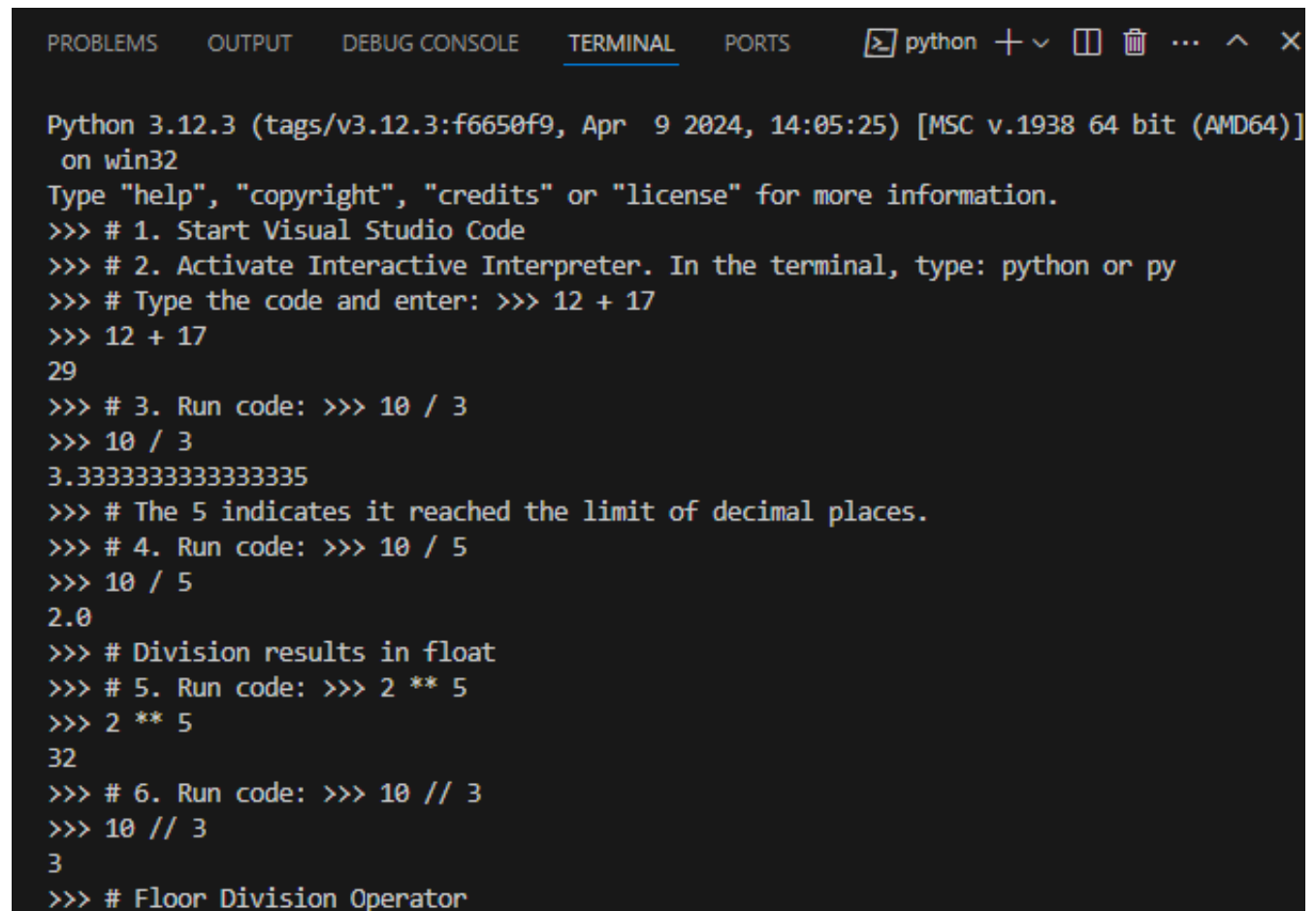


Week 5 Lab 4

Lab materials saved on GitHub in GISC2335_ProgrammingForGIS/WeeklyContent/week5

https://github.com/crystaljhollis/DallasCollege_Portfolio/tree/45cba4dfeee2744b8d6d6902ea73696b0bbf2d8a/GISC2335_ProgrammingForGIS/WeeklyContent/week5

WORK WITH NUMBERS



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + - [ ] [ ] ... ^ X

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> # 1. Start Visual Studio Code
>>> # 2. Activate Interactive Interpreter. In the terminal, type: python or py
>>> # Type the code and enter: >>> 12 + 17
>>> 12 + 17
29
>>> # 3. Run code: >>> 10 / 3
>>> 10 / 3
3.3333333333333335
>>> # The 5 indicates it reached the limit of decimal places.
>>> # 4. Run code: >>> 10 / 5
>>> 10 / 5
2.0
>>> # Division results in float
>>> # 5. Run code: >>> 2 ** 5
>>> 2 ** 5
32
>>> # 6. Run code: >>> 10 // 3
>>> 10 // 3
3
>>> # Floor Division Operator
```

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS python + v [ ] [ ] ... ^ X

>>> # WORK WITH STRINGS
>>> # 1. Run code: >>> print("Hello World")
>>> print("Hello World")
Hello World
>>> # String of characters. Make consistent use of '' and "" marks
>>> # 2. Run code: >>> print('Let's go!')
>>> print('Let's go!')
File "<stdin>", line 1
    print('Let's go!')
    ^^^^^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
>>> # 3. Run code: >>> print("Let's go!")
>>> print("Let's go!")
Let's go!
>>> # 4. Run code: >>> z = "Alphabet Soup"
>>> # >>> print(z[7])
>>> z = "Alphabet Soup"
>>> print(z[7])
t
>>> # 5. Run code: >>> print(z[-1])
>>> print(z[-1])
p
>>> # Letter p from soup. Slicing
>>> # 6. Run code: >>> print(z[0])
>>> print(z[0])
A
>>> # 7. Run code: >>> print([0:8])
>>> print(z[0:8])
Alphabet

>>> # Returns index 0 to index 8
>>> # 8. Run code: >>> name = "Geographic Information Systems"
>>> # >>> name.find("Info")
>>> name = "Geographic Information Systems"
>>> name.find("Info")
11
>>> # 11 is index number of letter I. Spaces counted as characters
>>> [ ]

Ln 73, Col 1 (62 selected) Spaces: 4 UTF-8 CRLF {} Python 3.12.3 [ ]
```

WORK WITH VARIABLES

```
>>> # WORK WITH VARIABLES
>>> # 1. Run code: >>> x = 12
>>> # >>> print(x)
>>> x = 12
>>> print(x)
12
>>> # 2. Run code: >>> x = 12
>>> # >>> y = x / 4
>>> # >>> print(y)
>>> x = 12
>>> y = x / 4
>>> print(y)
3.0
>>> # Variables store different data types like numbers, strings,
>>> # lists, tuples, dictionaries, files, etc.
>>> # 3. Run code: >>> k = 'This is a string'
>>> # >>> print(k)
>>> k = 'This is a string'
>>> print(k)
This is a string
>>> []
```

Ln 93, Col 28 (70 selected) Spaces: 4 UTF-8 CRLF {} Python 3.12.3

WORK WITH LISTS

```
>>> # WORK WITH LISTS
>>> # 1. Run code: >>> w = ["Apple", "Banana", "Cantaloupe", "Durian"]
>>> # >>> print(w)
>>> w = ["Apple", "Banana", "Cantaloupe", "Durian"]
>>> print(w)
['Apple', 'Banana', 'Cantaloupe', 'Durian']
>>> # 2. Run code: >>> print(w[0])
>>> print(w[0])
Apple
>>> # Indexing
>>> # 3. Run code: >>> print(w[-1])
>>> print(w[-1])
Durian
>>> #4. Run code: >>> print(w[1:-1])
>>> print(w[1:-1])
['Banana', 'Cantaloupe']
>>> []
```

Ln 114, Col 1 (14 selected) Spaces: 4 UTF-8 CRLF {} Python 3.12.3

USE FUNCTIONS

```
>>> # USE FUNCTIONS
>>> # Standard functions complete list dir(_builtins_)
>>> # 1. Run code: >>> d = pow (2, 3)
>>> # >>> print(d)
>>> d = pow (2, 3)
>>> print(d)
8
>>> # Power function pow () instead of exponential operator
>>> # 2. Run code: >>> print(dir(_builtins_))
Error', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'Connect
ionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarnin
g', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError', 'Exception',
'ExceptionGroup', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPo
intError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportW
arning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryErr
or', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotF
oundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotIm
plementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'Perm
issionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'Resour
ceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIterat
ion', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError',
'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError',
'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning',
'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError',
'_', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '_
_name__', '__package__', '__spec__', 'abs', 'aiter', 'all', 'anext', 'any', 'as
cii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'cl
assmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'd
ir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format'
```

Ln 131, Col 1 (92 selected) Spaces: 4 UTF-8 CRLF {}

```
, 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'i
nput', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'lo
cals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord
', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 's
et', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tupl
e', 'type', 'vars', 'zip']
>>> []
```

```
>>> # 3. Run code: >>> e = abs(-12.729)
>>> # >>> print(e)
>>> e = abs(-12.729)
>>>
>>> e = abs(-12.729)
>>> print(e)
12.729
>>> # 4. Enter: >>> type(
>>> # Syntax pop-up
```

3/18/2025

```
>>> # 5. Run code: >>> type(123)
>>> type(123)
<class 'int'>
>>> # Input value's type is an integer
>>> # 6. Run code: >>> type(1.23)
>>> type(1.23)
<class 'float'>
>>> # Input value's type is a floating-point number
>>> # 7. Run code: >>> type("GIS")
>>> type("GIS")
<class 'str'>
>>> # Input value's type is a string
>>> # 8. Enter: >>> round(
>>> # Syntax pop-up, containing function's parameters: number and ndigits
>>> # ndigits can be set to None
>>> # (number, ndigits=None)
>>> # Round a number to a given precision in decimal digits
>>> # 9. Run code: >>> round(1.234567, 4)
>>> round(1.234567, 4)
1.2346
>>> # 10. Run code: >>> round(1.234567)
>>> round(1.234567)
1
>>> []
```

Ln 171, Col 14 Spaces: 4 UTF-8 CRLF {} Python 3.12.3

USE METHODS

```
>>> # USE METHODS
>>> # 1. Run code: >>> topic = "Geographic Information Systems"
>>> # >>> topic.count("i")
>>> topic = "Geographic Information Systems"
>>> topic.count("i")
2
>>> # Output: 2 because that's how many lower case i are in the input string
>>> # Case sensitive
>>> # 2. Run code: >>> topic.split(" ")
>>> topic.split(" ")
['Geographic', 'Information', 'Systems']
>>> # splits the string
```

```

>>> # 3. Run code:
>>> # >>> path = "C:/data/part1/final"
>>> # >>> pathlist = path.split("/")
>>> # >>> lastpath = pathlist[-1]
>>> # >>> print(lastpath)
>>> path = "C:/data/part1/final"
>>> pathlist = path.split("/")
>>> lastpath = pathlist[-1]
>>> print(lastpath)
final
>>> # 4. Run code:
>>> # >>> mylist = ["A", "B", "C"]
>>> # >>> mylist.append("D")
>>> # >>> print(mylist)
>>> mylist = ["A", "B", "C"]
>>> mylist.append("D")
>>> print(mylist)
['A', 'B', 'C', 'D']
>>> 

```

USE MODULES

```

>>> # USE MODULES
>>> # 1. Run the following code:
>>> # >>> import math
>>> # >>> h = math.floor (7.89)
>>> # >>> print(h)
>>> import math
>>> h = math.floor (7.89)
>>> print(h)
7
>>> # 2. Run the following code:
>>> # >>> print(dir(math))
>>> print(dir(math))
['_doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh',
'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist',
'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod',
'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']

```

```

>>> # 3. Run the following code:
>>> # >>> print(math.radians.__doc__)
>>> print(math.radians.__doc__)
Convert angle x from degrees to radians.
>>> # 4-5. Locate complete list of modules in Python manuals
>>> # In IDLE, click Help > Python Docs.
>>> # At docs.python.org/3.11/
>>> # i.e. random module > uniform() function
>>> # 6. Return to Interpreter

```

```
>>> # 7. Run the following code:
>>> # >>> import random
>>> # >>> j = random.uniform(0, 100)
>>> # >>> print(j)
>>> import random
>>> j = random.uniform(0, 100)
>>> print(j)
52.44800550444629
```

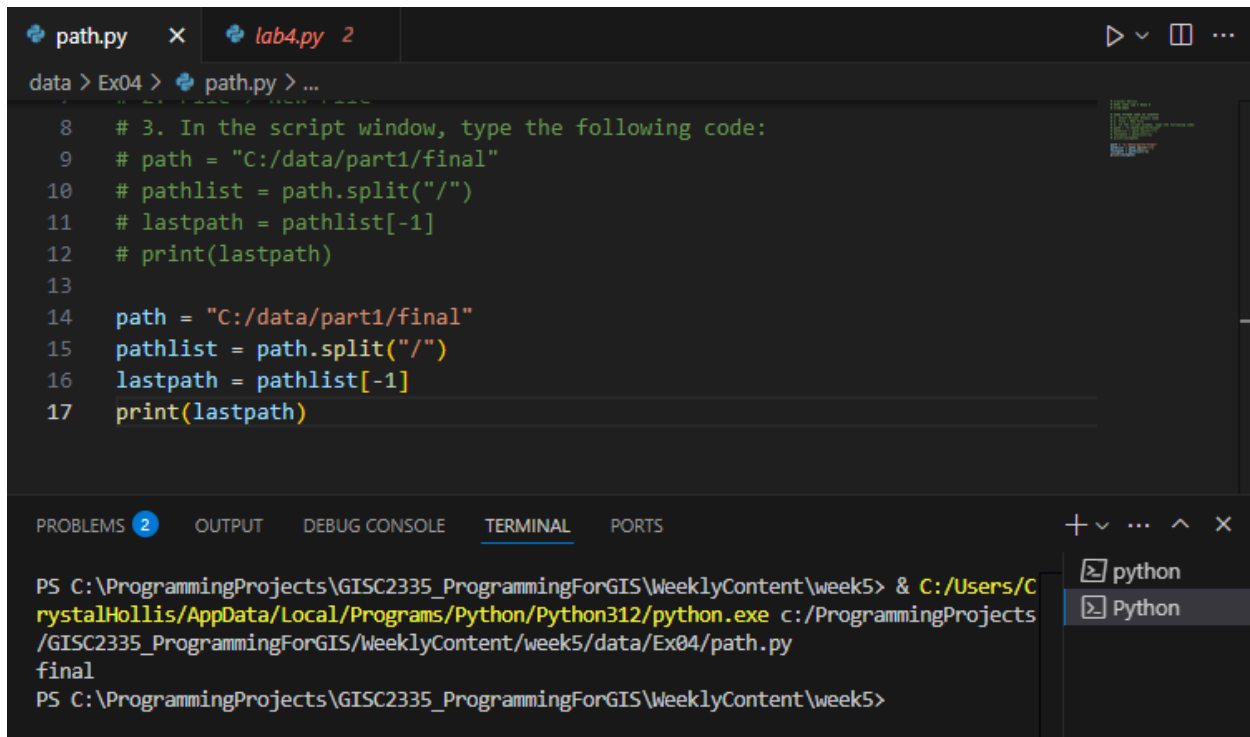
The rest of the lab materials are saved in additional scripts in Data/EX04

Python scripts saved in

GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04

https://github.com/crystaljhollis/DallasCollege_Portfolio/tree/main/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04

SAVE PYTHON CODE AS SCRIPTS



The screenshot shows a Python IDE with two tabs: `path.py` and `lab4.py 2`. The `path.py` tab is active, displaying the following code:

```
8 # 3. In the script window, type the following code:
9 # path = "C:/data/part1/final"
10 # pathlist = path.split("/")
11 # lastpath = pathlist[-1]
12 # print(lastpath)
13
14 path = "C:/data/part1/final"
15 pathlist = path.split("/")
16 lastpath = pathlist[-1]
17 print(lastpath)
```

The bottom panel shows the `TERMINAL` tab with the following command and output:

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/path.py
final
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>
```

On the right side of the terminal, there is a dropdown menu with the following options:

- python
- Python

WRITE CONDITIONAL STATEMENTS

```
1  # Crystal Hollis
2  # GISC 2335 GIS Programming
3  # 3/18/2025
4  # Week 5 Lab 4
5
6  # WRITE CONDITIONAL STATEMENTS
7  # 1. In the Python Shell, click File > New File.
8  # 2. Write the following code to generate a random number between 1 and 6:
9  # import random
10 # p = random.randint(1, 6)
11 # print(p)
12
13 import random
14 p = random.randint(1, 6)
15 print(p)
16
17 # 3. Save the script as branching.py to the C:\PythonPro\Ex04 folder.
18 # 4. Run the script to confirm that it works correctly.
19
20 # 5. Replace line 3 with the following:
21 #if p == 6:
22 # 6. Write the following line of code following the if statement:
23 # print("You win!" )
24 # 8. In the branching.py script, place your pointer at the end of the line
```

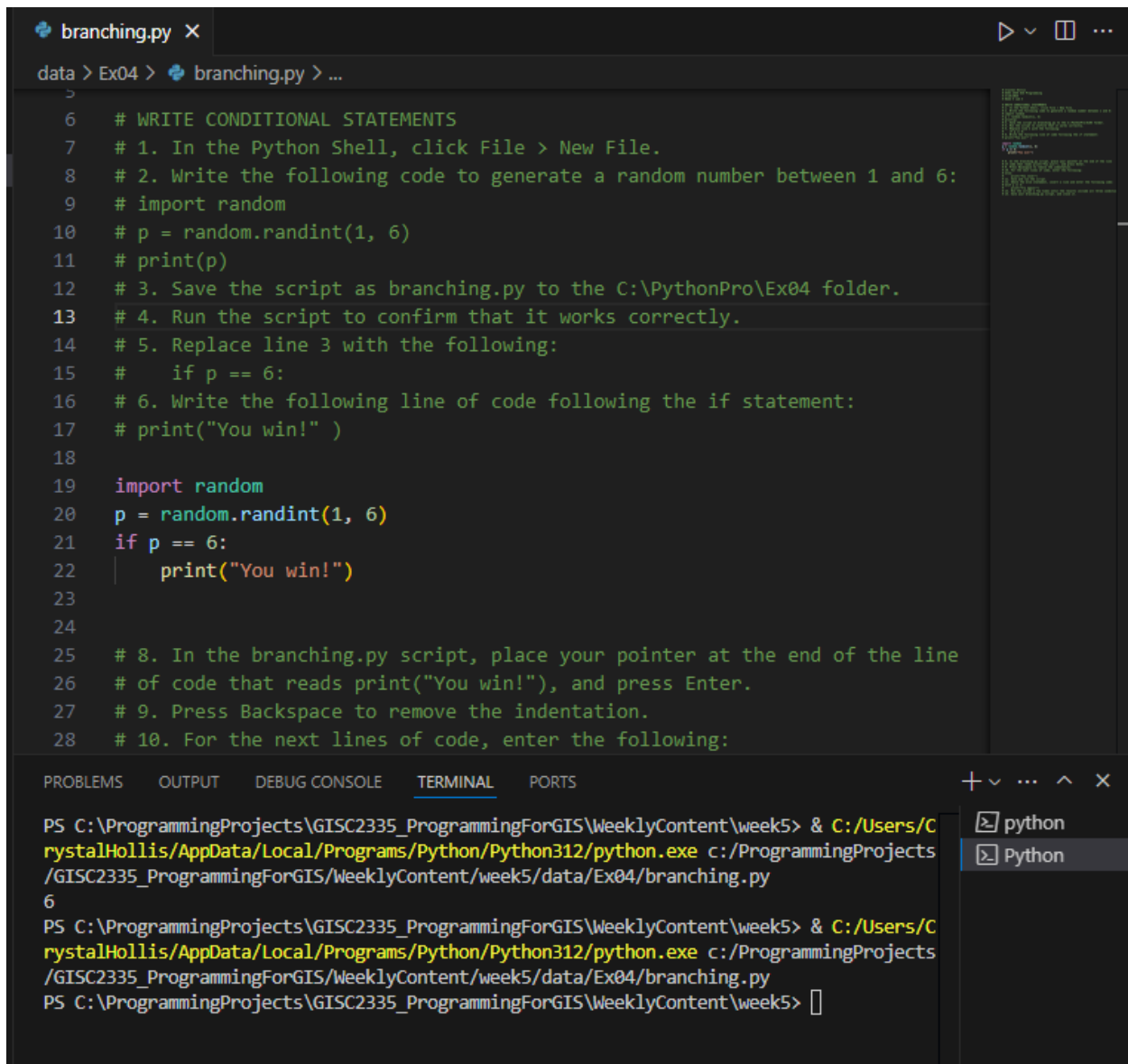
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
6
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>
```

+ v ... ^ x

python

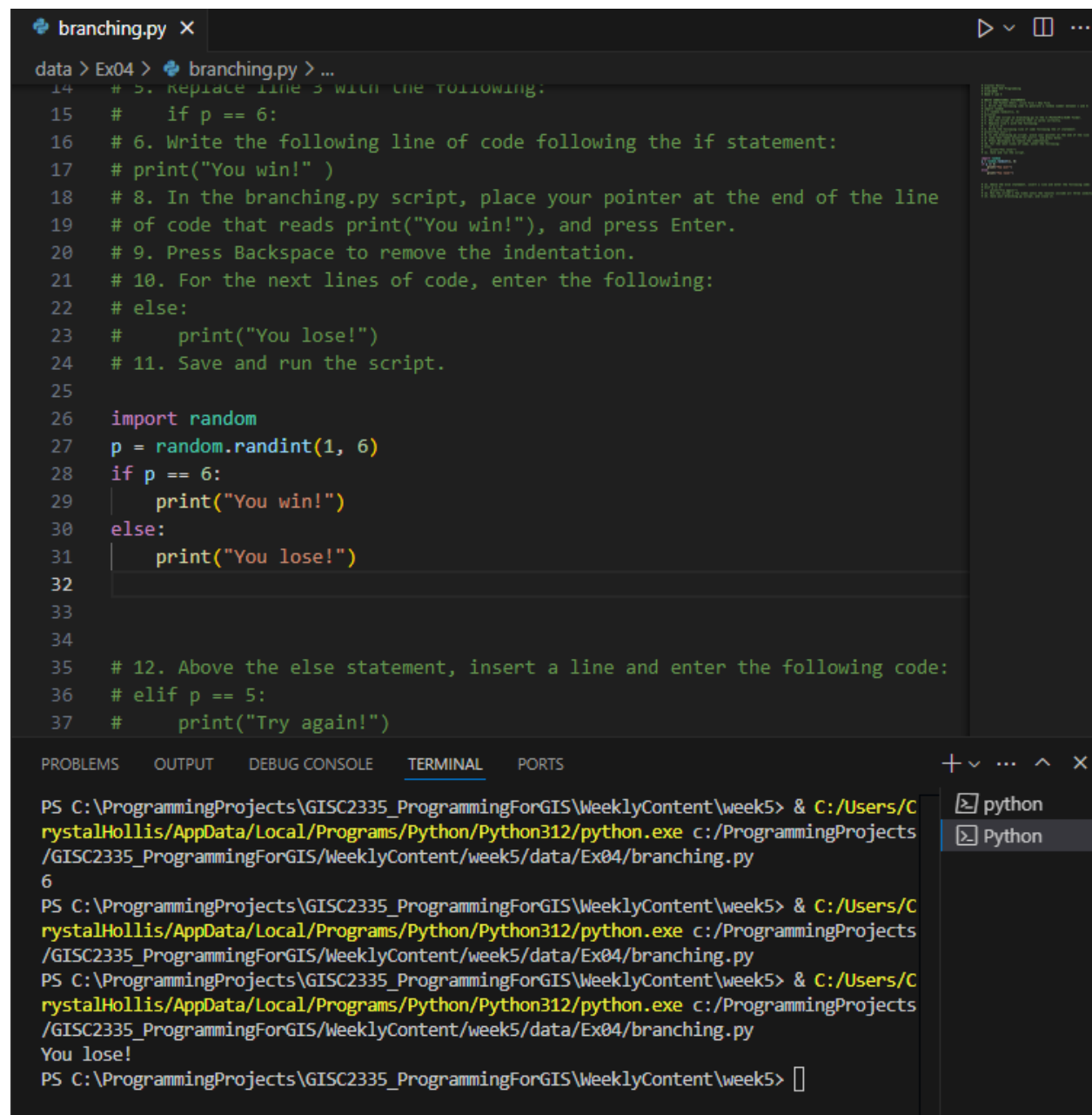
Python



The image shows a Python IDE with a file named `branching.py` open. The script contains instructions for writing conditional statements and a small Python program that generates a random number and checks if it is 6. The terminal window at the bottom shows the command to run the script, which outputs the number 6.

```
branching.py X
data > Ex04 > branching.py > ...
6 # WRITE CONDITIONAL STATEMENTS
7 # 1. In the Python Shell, click File > New File.
8 # 2. Write the following code to generate a random number between 1 and 6:
9 # import random
10 # p = random.randint(1, 6)
11 # print(p)
12 # 3. Save the script as branching.py to the C:\PythonPro\Ex04 folder.
13 # 4. Run the script to confirm that it works correctly.
14 # 5. Replace line 3 with the following:
15 #     if p == 6:
16 # 6. Write the following line of code following the if statement:
17 # print("You win!")
18
19 import random
20 p = random.randint(1, 6)
21 if p == 6:
22     print("You win!")
23
24
25 # 8. In the branching.py script, place your pointer at the end of the line
26 # of code that reads print("You win!"), and press Enter.
27 # 9. Press Backspace to remove the indentation.
28 # 10. For the next lines of code, enter the following:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
6
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> 
```



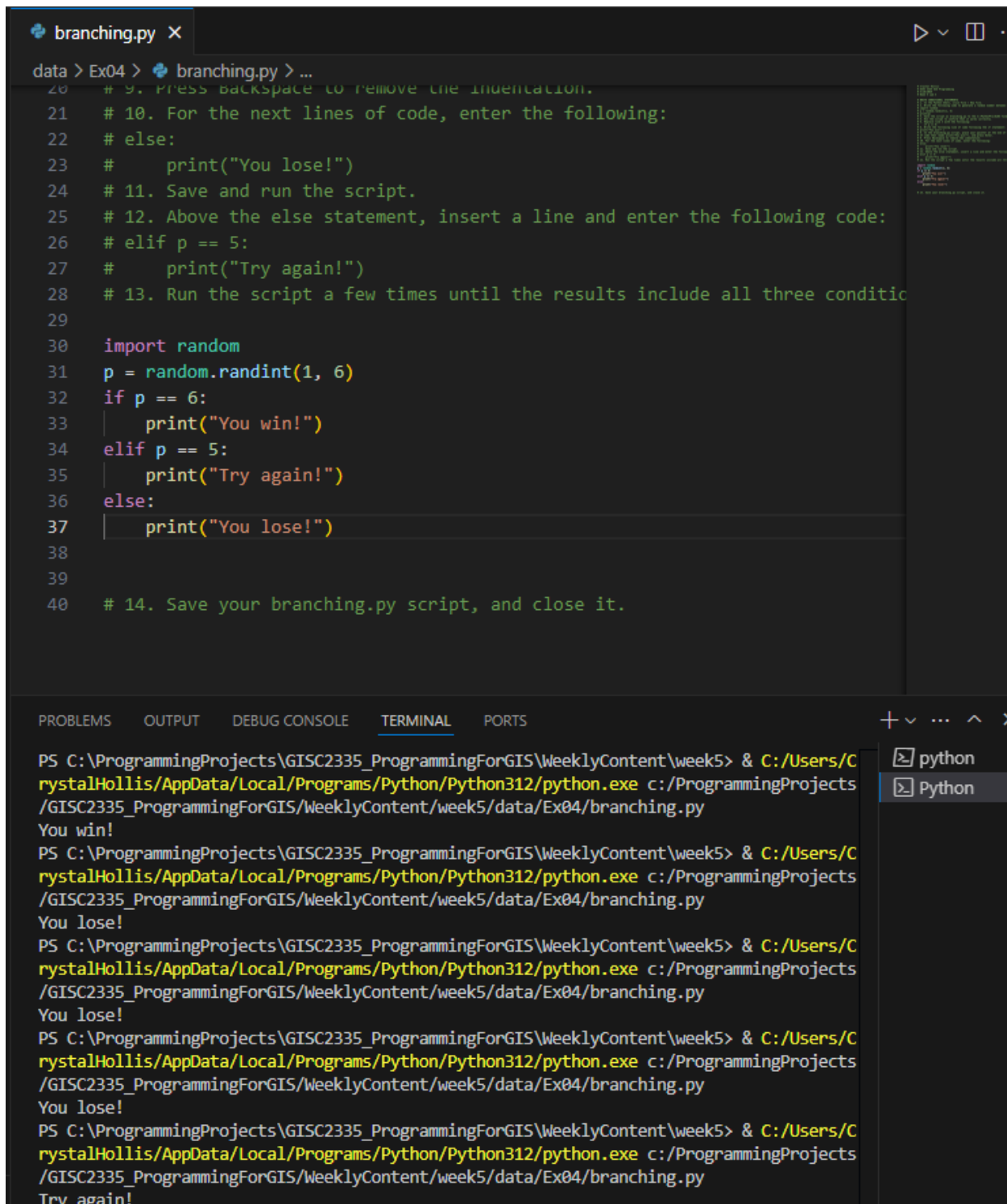
The image shows a Visual Studio Code editor window with a file named `branching.py` open. The editor has a dark theme. The Python script contains comments and code for a simple game. The terminal at the bottom shows the command to run the script and its output.

```
data > Ex04 > branching.py > ...
14 # 5. Replace line 5 with the following:
15 #     if p == 6:
16 # 6. Write the following line of code following the if statement:
17 # print("You win!")
18 # 8. In the branching.py script, place your pointer at the end of the line
19 # of code that reads print("You win!"), and press Enter.
20 # 9. Press Backspace to remove the indentation.
21 # 10. For the next lines of code, enter the following:
22 # else:
23 #     print("You lose!")
24 # 11. Save and run the script.
25
26 import random
27 p = random.randint(1, 6)
28 if p == 6:
29     print("You win!")
30 else:
31     print("You lose!")
32
33
34
35 # 12. Above the else statement, insert a line and enter the following code:
36 # elif p == 5:
37 #     print("Try again!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
6
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
You lose!
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> 
```

python
Python



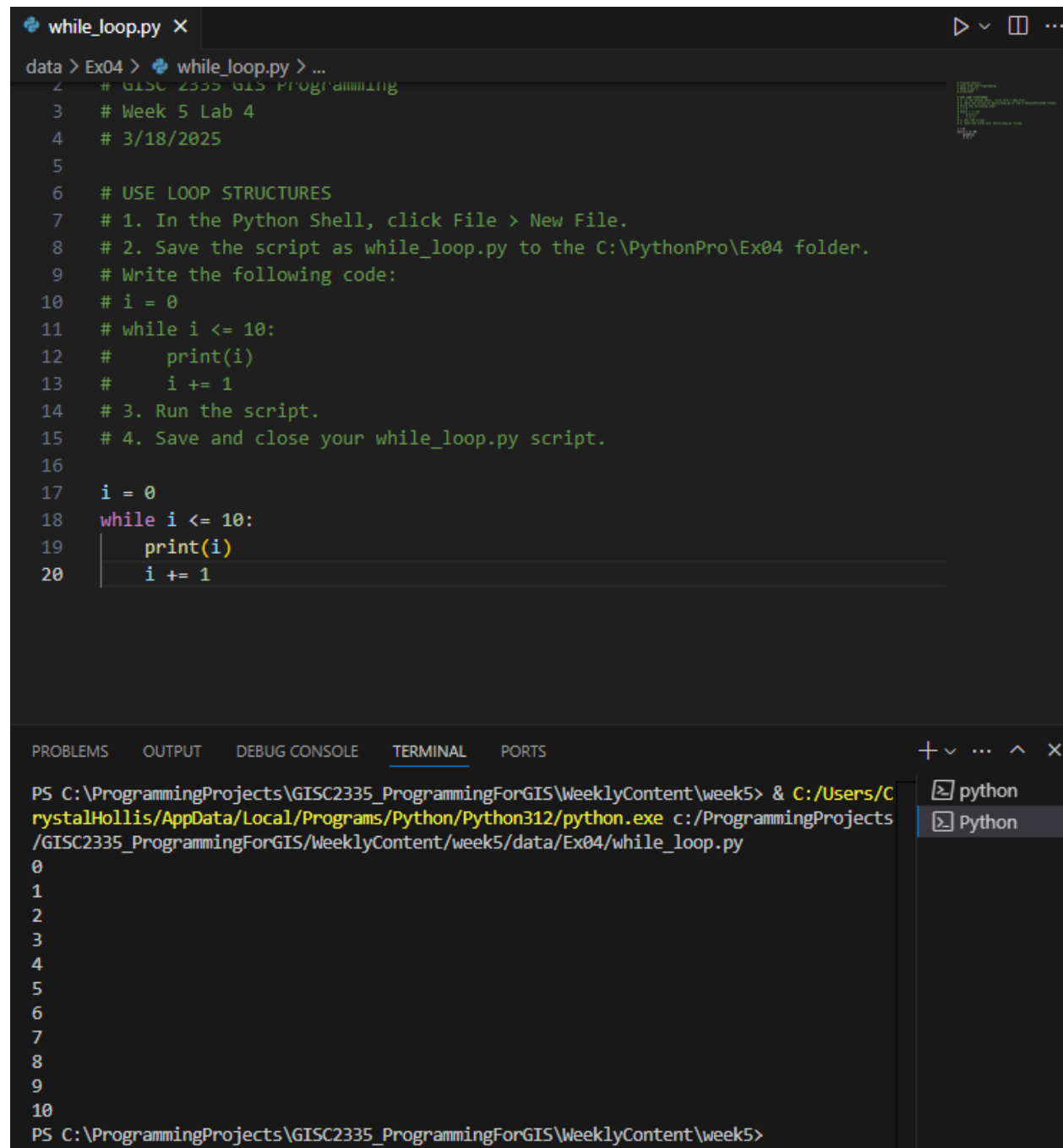
The image shows a Visual Studio Code editor window with a file named `branching.py` open. The script is a Python program that generates a random number between 1 and 6 and prints a message based on its value. The terminal window at the bottom shows the script being executed multiple times, resulting in "You win!", "You lose!", and "Try again!" messages.

```
data > Ex04 > branching.py > ...
20 # 9. Press backspace to remove the indentation.
21 # 10. For the next lines of code, enter the following:
22 # else:
23 #     print("You lose!")
24 # 11. Save and run the script.
25 # 12. Above the else statement, insert a line and enter the following code:
26 # elif p == 5:
27 #     print("Try again!")
28 # 13. Run the script a few times until the results include all three conditions.
29
30 import random
31 p = random.randint(1, 6)
32 if p == 6:
33     print("You win!")
34 elif p == 5:
35     print("Try again!")
36 else:
37     print("You lose!")
38
39
40 # 14. Save your branching.py script, and close it.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
You win!
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
You lose!
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
You lose!
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
You lose!
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/branching.py
Try again!
```

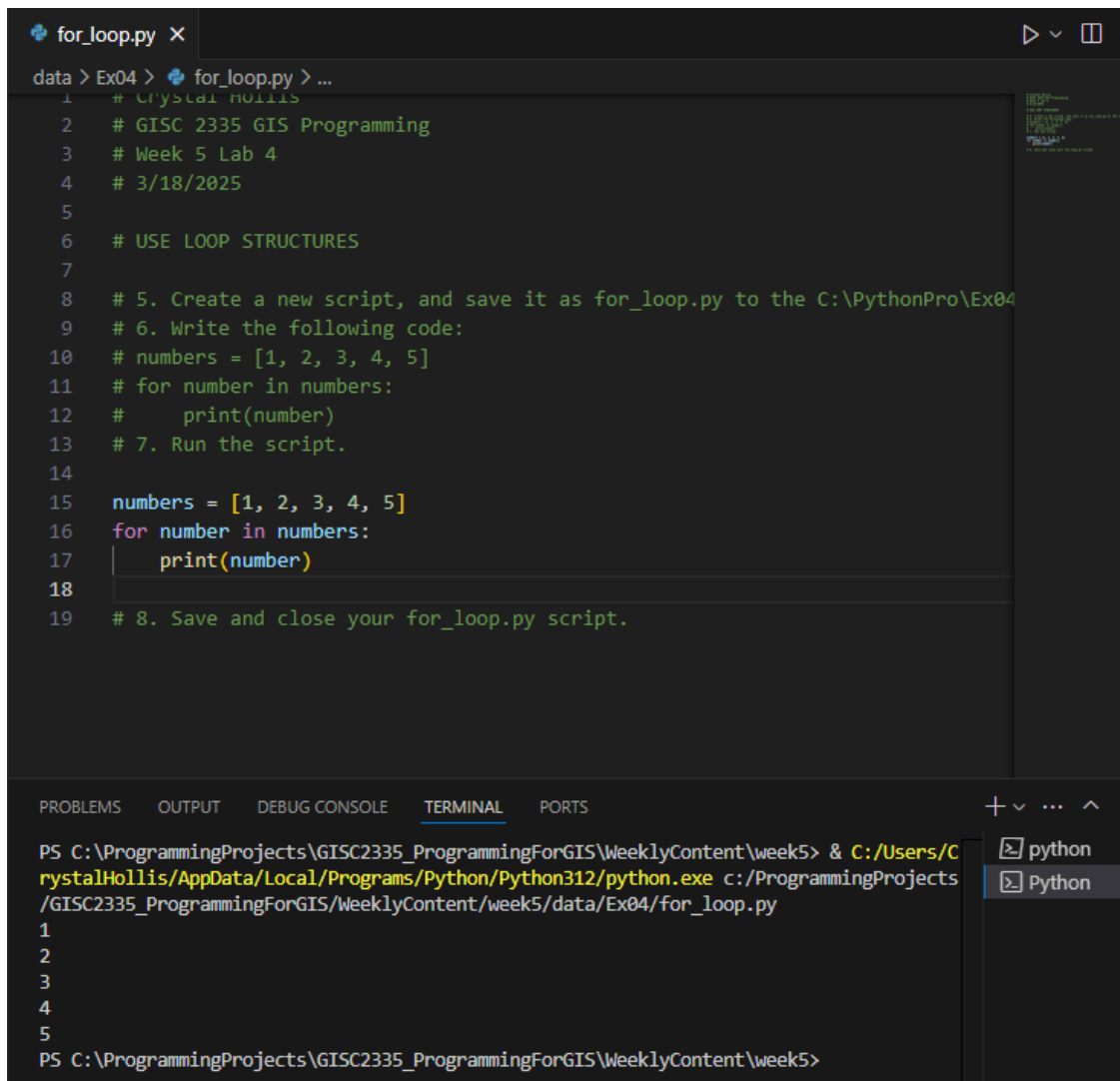
USE LOOP STRUCTURES



The screenshot shows a Python IDE with a file named `while_loop.py` open. The script contains a while loop that prints numbers from 0 to 10. Below the editor, the terminal window shows the command to run the script and the resulting output.

```
while_loop.py X
data > Ex04 > while_loop.py > ...
2  # GISC 2335 GIS Programming
3  # Week 5 Lab 4
4  # 3/18/2025
5
6  # USE LOOP STRUCTURES
7  # 1. In the Python Shell, click File > New File.
8  # 2. Save the script as while_loop.py to the C:\PythonPro\Ex04 folder.
9  # Write the following code:
10 # i = 0
11 # while i <= 10:
12 #     print(i)
13 #     i += 1
14 # 3. Run the script.
15 # 4. Save and close your while_loop.py script.
16
17 i = 0
18 while i <= 10:
19     print(i)
20     i += 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/while_loop.py
0
1
2
3
4
5
6
7
8
9
10
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>
```



The image shows a screenshot of a Python IDE with a dark theme. The editor window displays a file named `for_loop.py` with the following content:

```
1 # Crystal Hollis
2 # GISC 2335 GIS Programming
3 # Week 5 Lab 4
4 # 3/18/2025
5
6 # USE LOOP STRUCTURES
7
8 # 5. Create a new script, and save it as for_loop.py to the C:\PythonPro\Ex04
9 # 6. Write the following code:
10 # numbers = [1, 2, 3, 4, 5]
11 # for number in numbers:
12 #     print(number)
13 # 7. Run the script.
14
15 numbers = [1, 2, 3, 4, 5]
16 for number in numbers:
17     print(number)
18
19 # 8. Save and close your for_loop.py script.
```

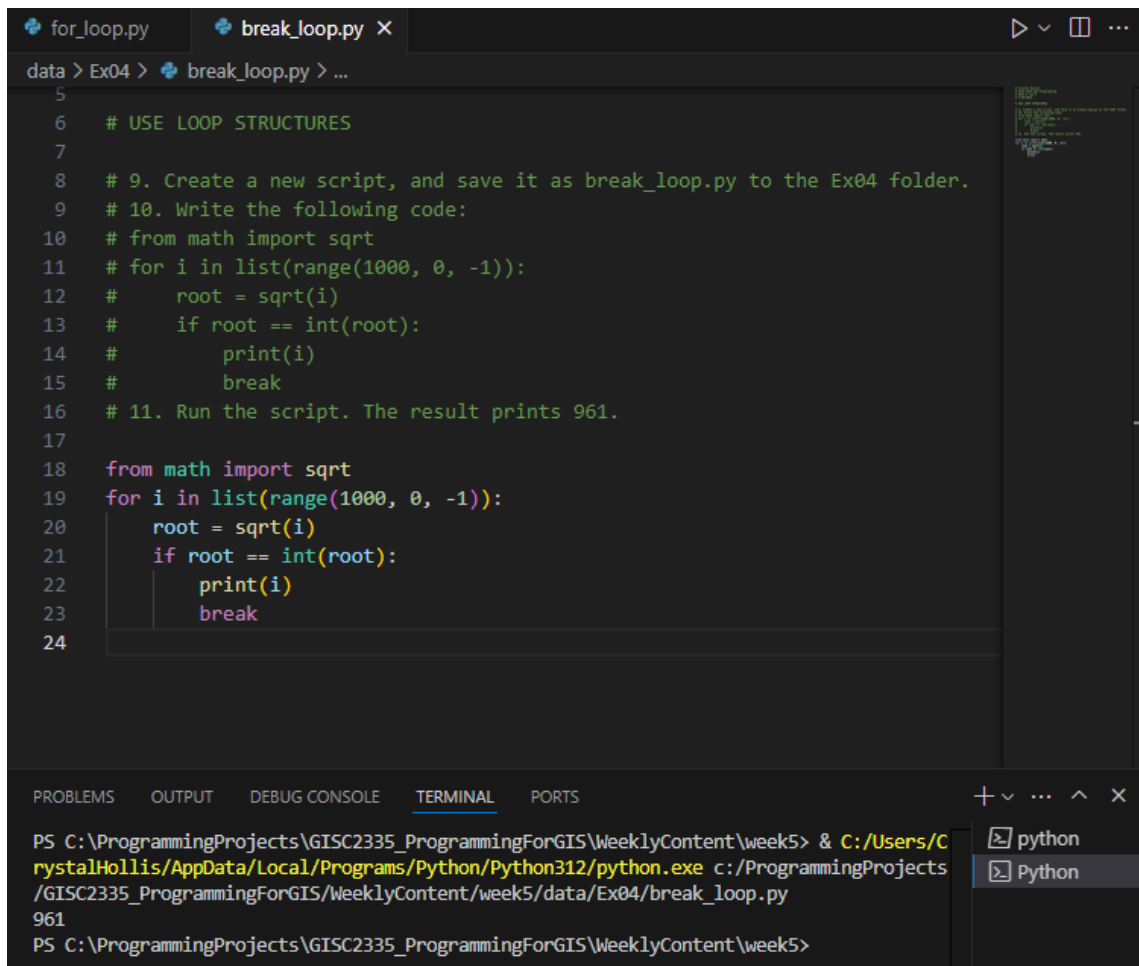
The bottom panel of the IDE shows the **TERMINAL** tab. The command prompt displays the command to run the script:

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/CrystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/for_loop.py
```

The output of the script is shown in the terminal:

```
1
2
3
4
5
```

The terminal prompt is now `PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>`. On the right side of the terminal, there is a dropdown menu with `python` and `Python` options.



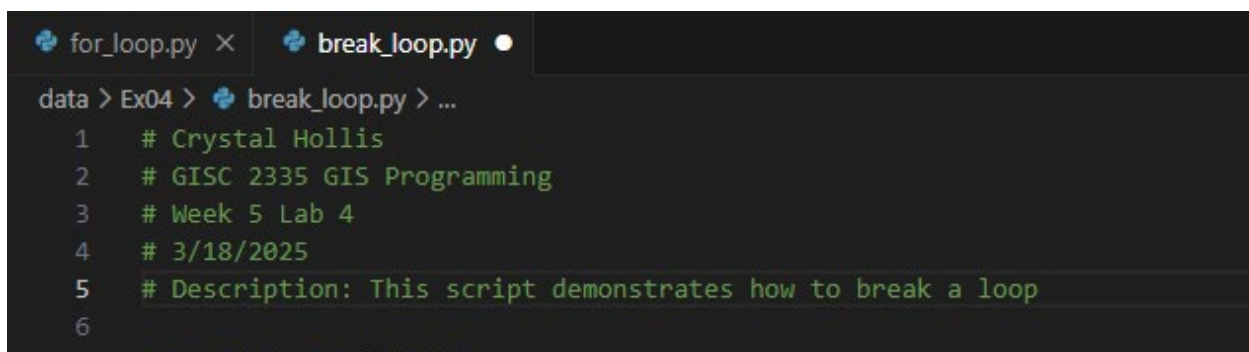
The screenshot shows a code editor with two tabs: `for_loop.py` and `break_loop.py`. The `break_loop.py` tab is active, displaying a Python script. The script includes comments explaining the task: creating a script to find perfect squares between 0 and 1000. The code uses a `for` loop with a `break` statement to exit the loop once a perfect square is found. The terminal output shows the command to run the script, which prints the number 961.

```
5  
6 # USE LOOP STRUCTURES  
7  
8 # 9. Create a new script, and save it as break_loop.py to the Ex04 folder.  
9 # 10. Write the following code:  
10 # from math import sqrt  
11 # for i in list(range(1000, 0, -1)):  
12 #     root = sqrt(i)  
13 #     if root == int(root):  
14 #         print(i)  
15 #         break  
16 # 11. Run the script. The result prints 961.  
17  
18 from math import sqrt  
19 for i in list(range(1000, 0, -1)):  
20     root = sqrt(i)  
21     if root == int(root):  
22         print(i)  
23         break  
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5> & C:/Users/C  
rystalHollis/AppData/Local/Programs/Python/Python312/python.exe c:/ProgrammingProjects  
/GISC2335_ProgrammingForGIS/WeeklyContent/week5/data/Ex04/break_loop.py  
961  
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>
```

COMMENT SCRIPTS



The screenshot shows a code editor with two tabs: `for_loop.py` and `break_loop.py`. The `break_loop.py` tab is active, displaying the first six lines of a Python script. The lines are comments providing metadata about the script, including the author's name, course, lab number, date, and a description of the script's purpose.

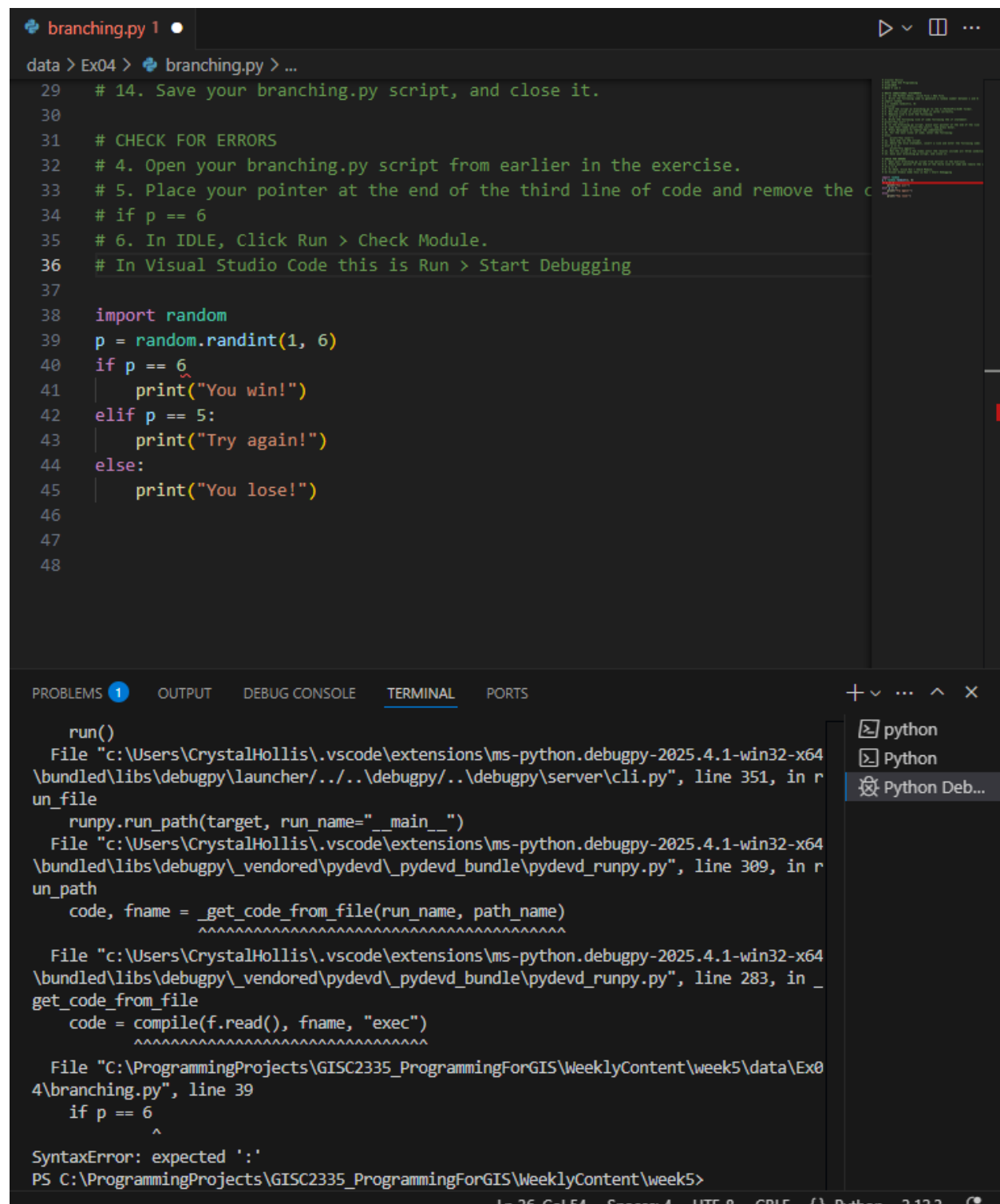
```
1 # Crystal Hollis  
2 # GISC 2335 GIS Programming  
3 # Week 5 Lab 4  
4 # 3/18/2025  
5 # Description: This script demonstrates how to break a loop  
6
```

```
for_loop.py × break_loop.py ●
data > Ex04 > break_loop.py > ...
19
20 # COMMENT SCRIPTS
21 # 1. In the break_loop.py script, place your pointer at the beginning of the
22 # Name: <your name>
23 # Date: <current date>
24 # Description: This script demonstrates how to break a loop
25
26 # 2. At the end of the line if root == int(root), enter a few spaces and the
27 # This evaluates whether the root is an integer.
28 # 3. In the break_loop.py script, select and highlight the last three lines
29 # 4. In IDLE, click Format > Comment Out Region. In Visual Studio Code, comm
30 # 5. In IDLE, To make the code active again, select and highlight the comment
31 # In Visual Studio Code, just go back to Edit > Toggle Line Comment to remov
32
33 from math import sqrt
34 for i in list(range(1000, 0, -1)):
35     root = sqrt(i)
36     # if root == int(root): # This evaluates whether the root is an integer
37     #     print(i)
38     #     break
39
40 # 6. Save your break_loop.py script.
```

```
for_loop.py break_loop.py ●
data > Ex04 > break_loop.py > ...
19
20 # COMMENT SCRIPTS
21 # 1. In the break_loop.py script, place your pointer at the beginning of the
22 # Name: <your name>
23 # Date: <current date>
24 # Description: This script demonstrates how to break a loop
25
26 # 2. At the end of the line if root == int(root), enter a few spaces and then
27 # This evaluates whether the root is an integer.
28 # 3. In the break_loop.py script, select and highlight the last three lines c
29 # 4. In IDLE, click Format > Comment Out Region. In Visual Studio Code, comme
30 # 5. In IDLE, To make the code active again, select and highlight the comment
31 # In Visual Studio Code, just go back to Edit > Toggle Line Comment to remove
32
33 from math import sqrt
34 for i in list(range(1000, 0, -1)):
35     root = sqrt(i)
36     if root == int(root): # This evaluates whether the root is an integer
37         print(i)
38         break
39
40 # 6. Save your break_loop.py script.
```



```
>>> # CHECK FOR ERRORS
>>> # 1. Return to the Python Shell.
>>> # 2. Run the following code, in which print is intentionally misspelled:
>>> # >>> pint("Hello World!")
>>> pint("Hello World!")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pint' is not defined. Did you mean: 'print'?
>>> # 3. Try a small variation of your print statement by running the following code:
>>> # >>> print("Hello World!)
>>> print("Hello World!)
  File "<stdin>", line 1
    print("Hello World!)
    ^
SyntaxError: unterminated string literal (detected at line 1)
>>> █
```

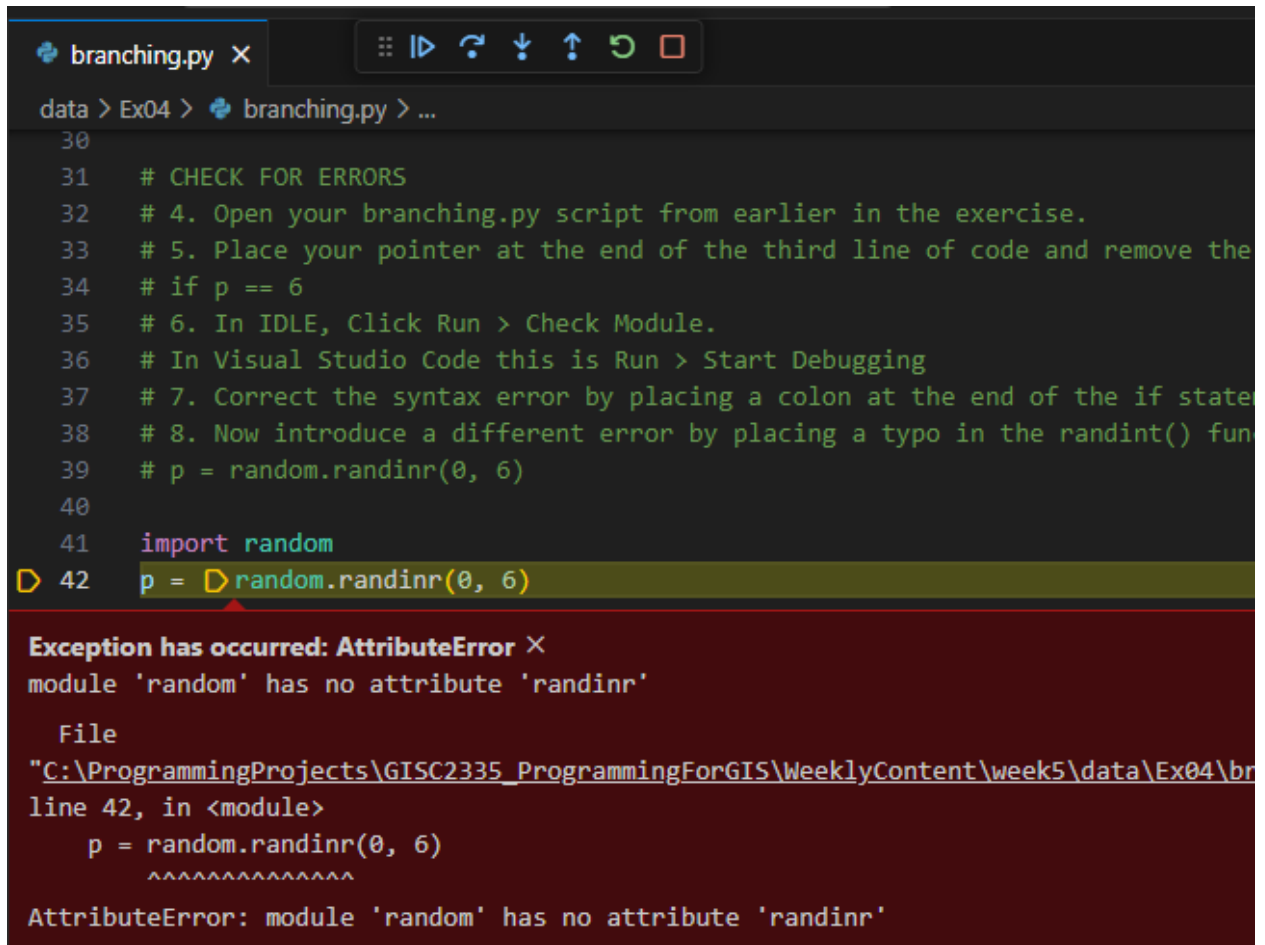
```
data > Ex04 > branching.py > ...
29 # 14. Save your branching.py script, and close it.
30
31 # CHECK FOR ERRORS
32 # 4. Open your branching.py script from earlier in the exercise.
33 # 5. Place your pointer at the end of the third line of code and remove the c
34 # if p == 6
35 # 6. In IDLE, Click Run > Check Module.
36 # In Visual Studio Code this is Run > Start Debugging
37
38 import random
39 p = random.randint(1, 6)
40 if p == 6
41     print("You win!")
42 elif p == 5:
43     print("Try again!")
44 else:
45     print("You lose!")
46
47
48
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
run()
File "c:\Users\CrystalHollis\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64
\bundled\libs\debugpy\launcher\..\..\debugpy\..\debugpy\server\cli.py", line 351, in r
un_file
runpy.run_path(target, run_name="__main__")
File "c:\Users\CrystalHollis\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64
\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 309, in r
un_path
code, fname = _get_code_from_file(run_name, path_name)
~~~~~
File "c:\Users\CrystalHollis\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64
\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 283, in _
get_code_from_file
code = compile(f.read(), fname, "exec")
~~~~~
File "C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5\data\Ex0
4\branching.py", line 39
if p == 6
^
SyntaxError: expected ':'
PS C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5>
```

python
Python
Python Deb...

In 36 Col 54 Spaces: 4 UTF-8 CRLF Python 3.12.3



The screenshot shows a Python IDE window titled 'branching.py'. The editor contains a script with several lines of comments and code. Line 42 is highlighted, showing the code `p = random.randinr(0, 6)`. Below the editor, a red error message is displayed, indicating an `AttributeError` because the `random` module does not have an attribute `randinr`. The error message includes the file path and the specific line of code that caused the error.

```
branching.py × [Run] [Debug] [Break] [Step Into] [Step Over] [Step Out] [Stop]
```

```
data > Ex04 > branching.py > ...  
30  
31 # CHECK FOR ERRORS  
32 # 4. Open your branching.py script from earlier in the exercise.  
33 # 5. Place your pointer at the end of the third line of code and remove the  
34 # if p == 6  
35 # 6. In IDLE, Click Run > Check Module.  
36 # In Visual Studio Code this is Run > Start Debugging  
37 # 7. Correct the syntax error by placing a colon at the end of the if state  
38 # 8. Now introduce a different error by placing a typo in the randint() fun  
39 # p = random.randinr(0, 6)  
40  
41 import random  
42 p = random.randinr(0, 6)
```

Exception has occurred: AttributeError ×
module 'random' has no attribute 'randinr'

File
"C:\ProgrammingProjects\GISC2335_ProgrammingForGIS\WeeklyContent\week5\data\Ex04\br
line 42, in <module>
p = random.randinr(0, 6)
^^^^^^^^^^^^^^^^
AttributeError: module 'random' has no attribute 'randinr'