Pytorial

By Parker Lowney

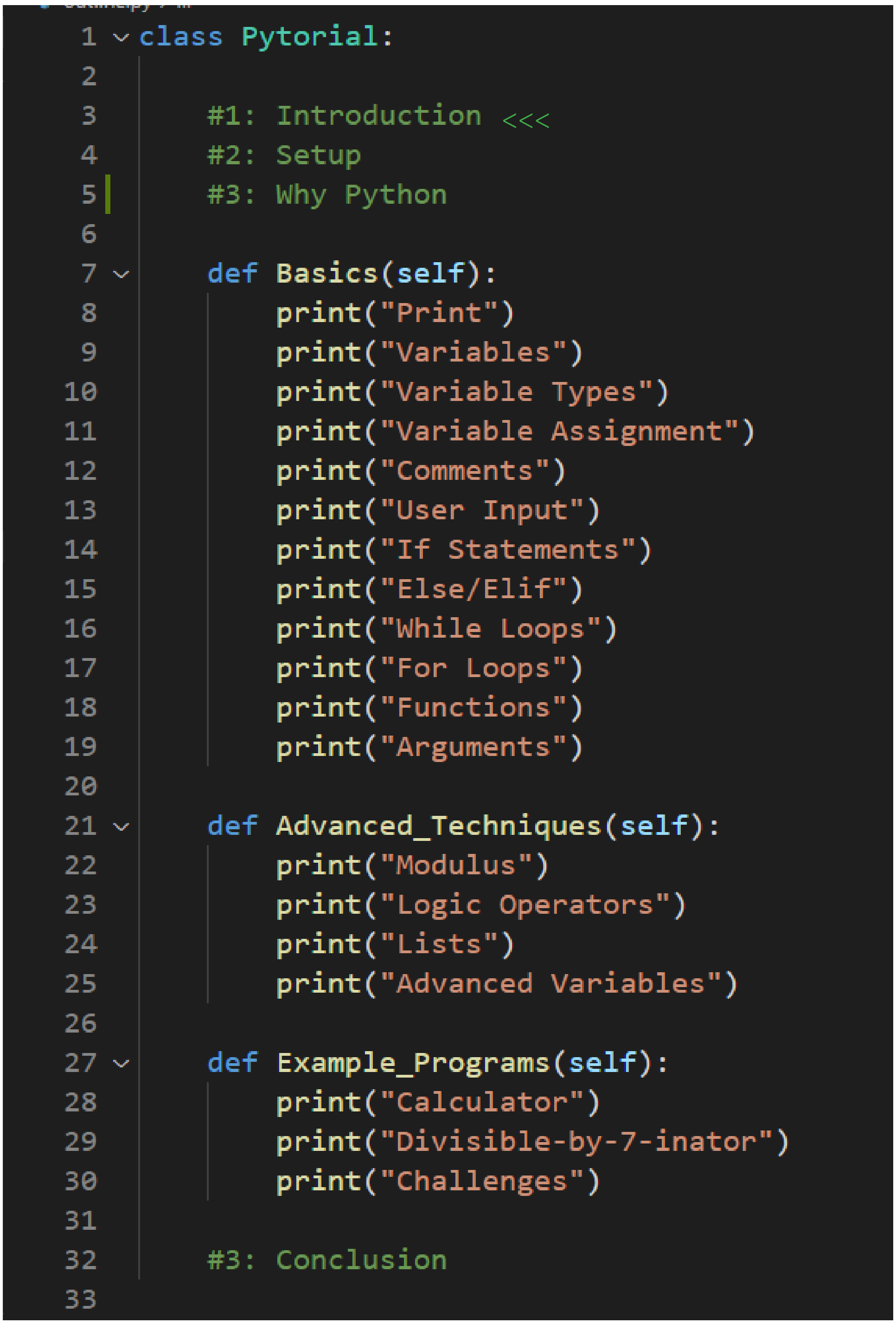
Acknowledgements

Linus Torvalds

Matton Masri

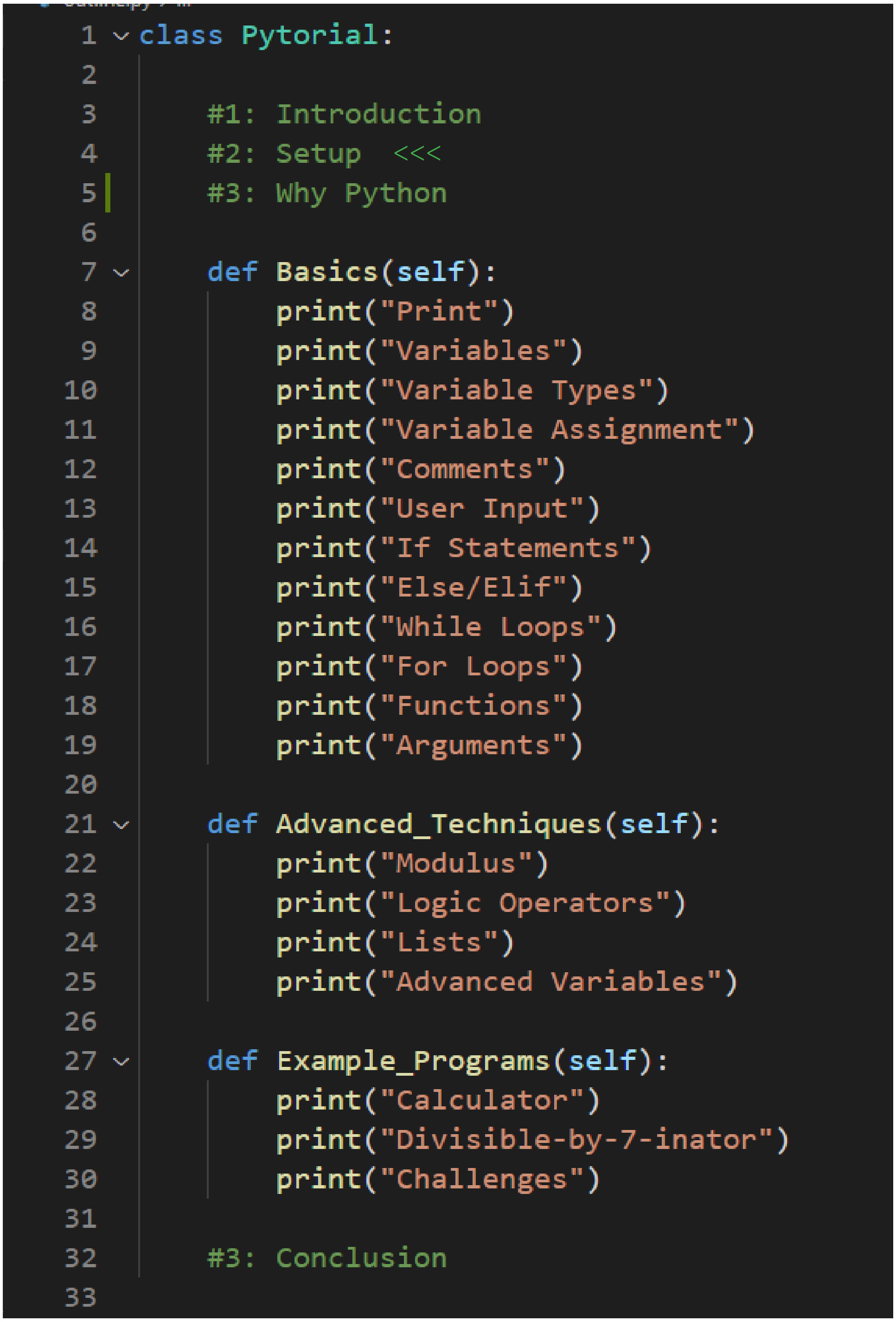
Mr. Connolly

Todd



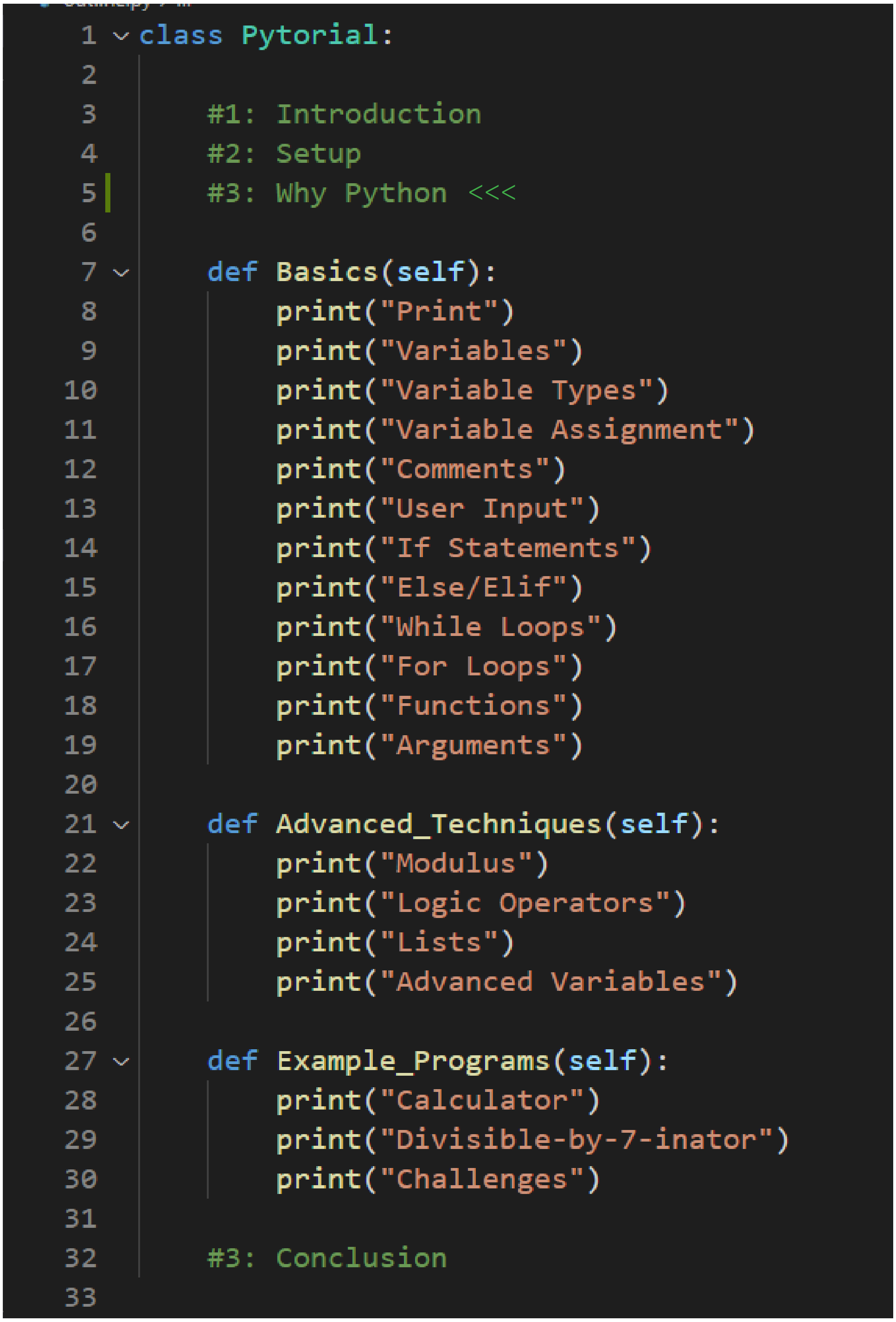
```
class Pytorial:

    #1: Introduction <<<
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

So you want to learn to program? Well you've come to the right book! In this book you'll learn the basics of programming in the Python language with descriptions, code examples, and some challenges at the end to test your skills. I encourage you to try playing around with code as you go through the book, and all of the files used in this book are publicly available on GitHub at this URL:
https://github.com/JuridicalBear14/Pytorial

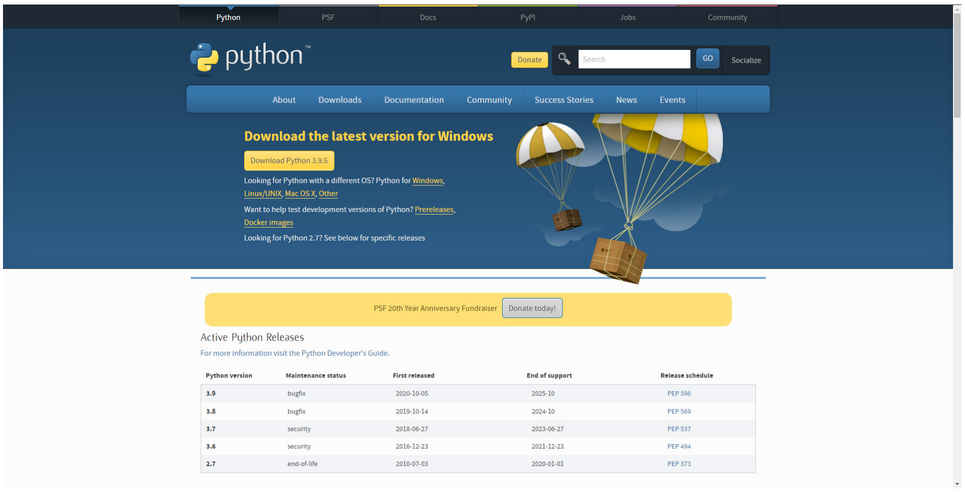


```
class Pytorial:

    #1: Introduction
    #2: Setup  <<<
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

You're probably wondering why this book is in Python and not another language, well there's a pretty good answer for that. Python is a lot simpler in syntax and structure than most other languages, so it's easier to learn with no experience. Python also has much looser rules than other languages, especially when it comes to assigning and adjusting variables. This can cause problems in large complex programs, but in small early programs it makes understanding what's happening much easier.



```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python <<<

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

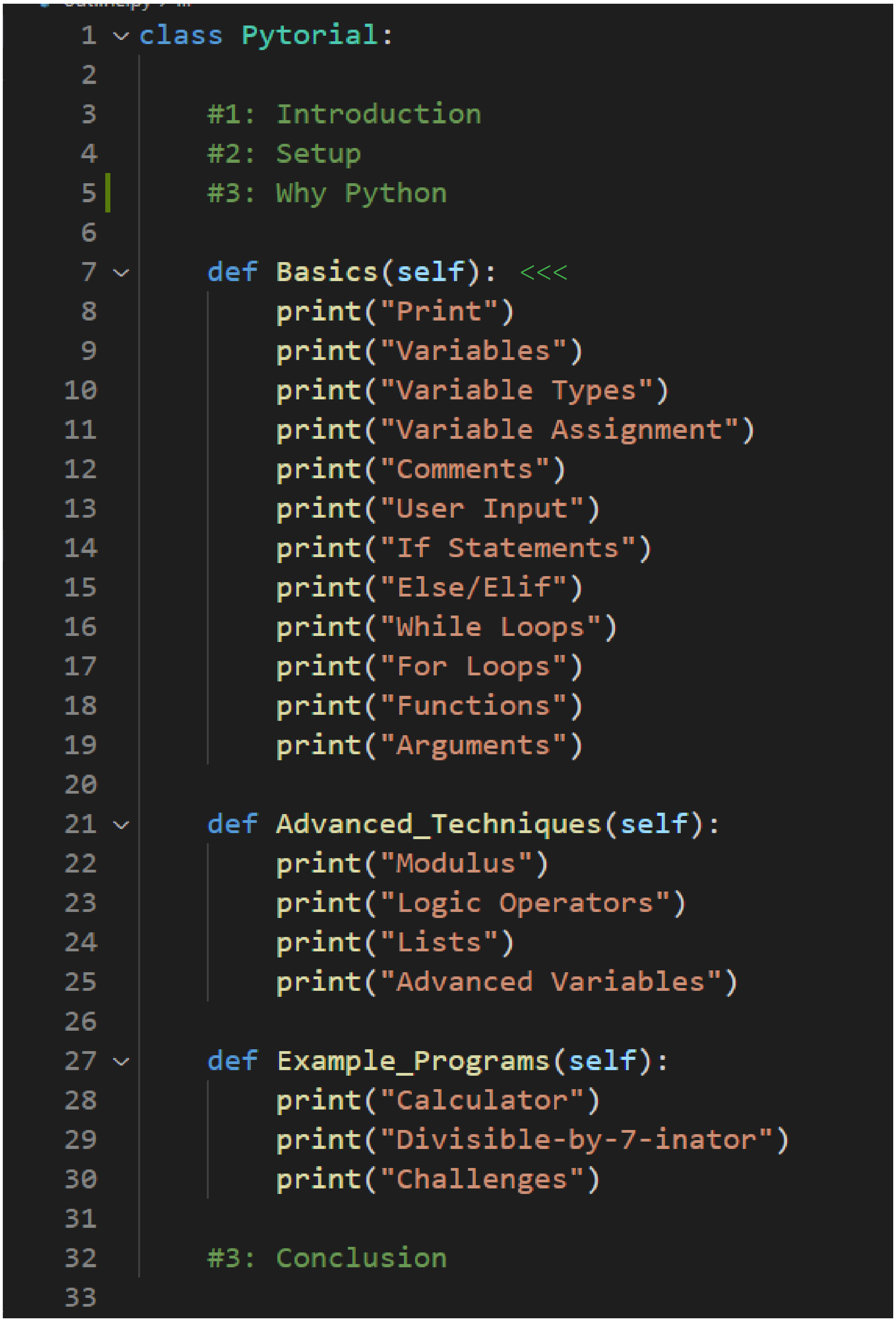
PSF 20th Year Anniversary Fundraiser

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.9	bugfix	2020-10-05	2025-10	PEP 596
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

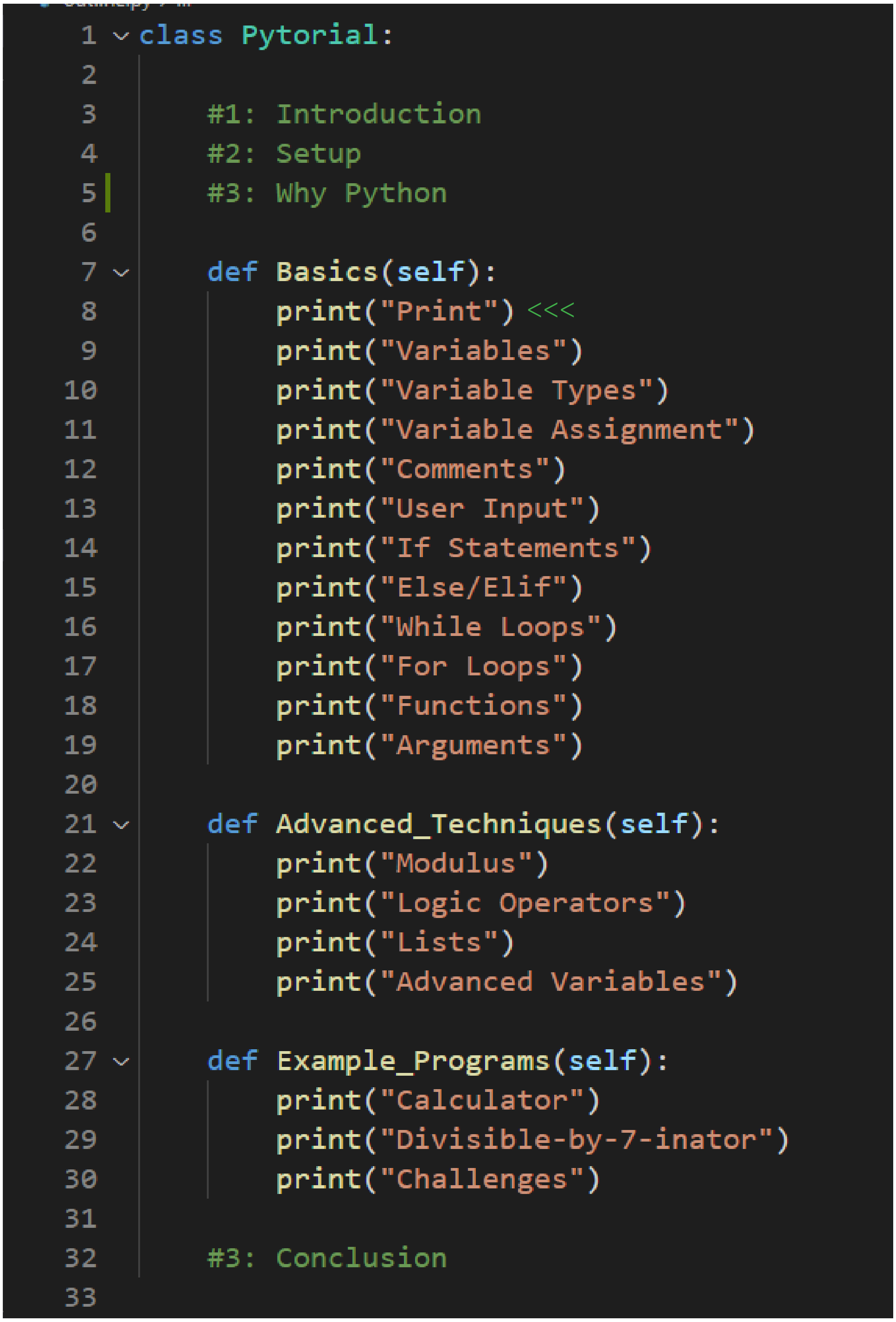
The first step to learning how to program in Python is to get it yourself. To run Python files you need a Python interpreter, or the code that translates your commands into the instructions for the computer to execute. You can download the Python language for free from https://python.org/ and get started right away running things in the command prompt. You can also optionally get an IDE, basically a text editor for programming, which helps with error checking and makes running programs easier. I personally like Visual Studio Code, which you can download for free at https://code.visualstudio.com/, but there are tons of options you can choose from. You also can find online Python interpreters if you don't want to download anything, I'd suggest https://onlinegdb.com/online_python_interpreter.



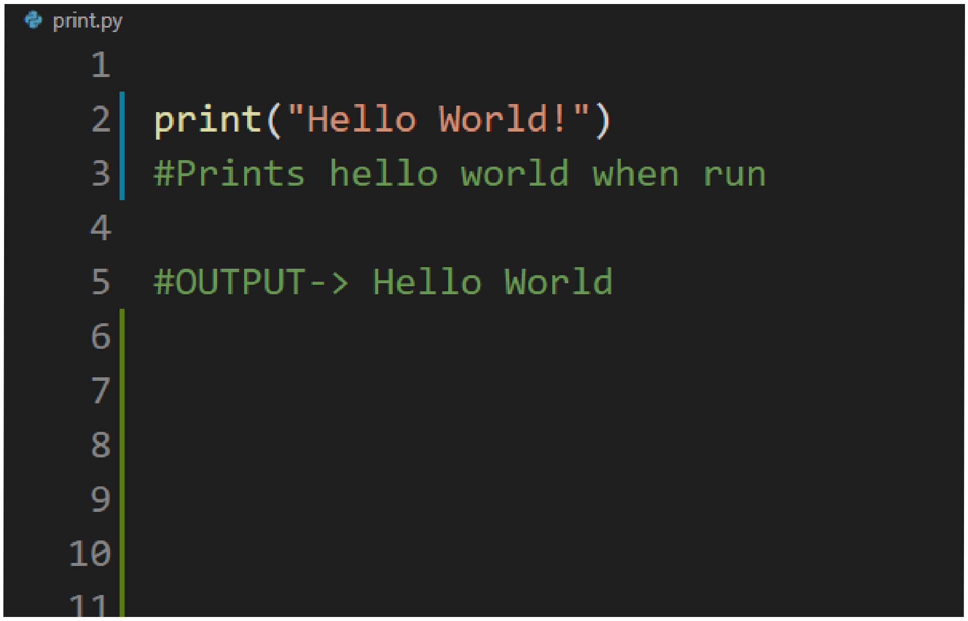
```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self): <<<
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

```
BBBBBBBBBBBBBBBBB               AAA                 SSSSSSSSSSSSSSS IIIIIIIIII      CCCCCCCCCCCCC   SSSSSSSSSSSSSSS
B::::::::::::::::B             A:::A              SS:::::::::::::::SI::::::::I   CCC::::::::::::C SS:::::::::::::::S
B::::::BBBBBB:::::B           A:::::A            S:::::SSSSSS::::::SI::::::::I CC:::::::::::::::CS:::::SSSSSS::::::S
BB:::::B     B:::::B         A:::::::A           S:::::S     SSSSSSSII::::::IIC:::::CCCCCCCC::::CS:::::S     SSSSSSS
  B::::B     B:::::B        A:::::::::A          S:::::S              I::::I C:::::C       CCCCCCS:::::S
  B::::B     B:::::B       A:::::A:::::A         S:::::S              I::::IC:::::C              S:::::S
  B::::BBBBBB:::::B       A:::::A A:::::A         S::::SSSS           I::::IC:::::C               S::::SSSS
  B:::::::::::::BB       A:::::A   A:::::A         SS::::::SSSSS      I::::IC:::::C                SS::::::SSSSS
  B::::BBBBBB:::::B     A:::::A     A:::::A          SSS::::::::SS    I::::IC:::::C                  SSS::::::::SS
  B::::B     B:::::B   A:::::AAAAAAAAA:::::A            SSSSSS::::S   I::::IC:::::C                     SSSSSS::::S
  B::::B     B:::::B  A:::::::::::::::::::::A                S:::::S  I::::IC:::::C                          S:::::S
  B::::B     B:::::B A:::::AAAAAAAAAAAAA:::::A               S:::::S  I::::I C:::::C       CCCCCC            S:::::S
BB:::::BBBBBB::::::BA:::::A             A:::::A  SSSSSSS     S:::::SII::::::IIC:::::CCCCCCCC::::CSSSSSSS     S:::::S
B:::::::::::::::::BA:::::A               A:::::A S::::::SSSSSS:::::SI::::::::I CC:::::::::::::::CS::::::SSSSSS:::::S
B::::::::::::::::BA:::::A                 A:::::AS:::::::::::::::SS I::::::::I   CCC::::::::::::CS:::::::::::::::SS
BBBBBBBBBBBBBBBBBAAAAAAA                   AAAAAAASSSSSSSSSSSSSSS   IIIIIIIIII      CCCCCCCCCCCCC SSSSSSSSSSSSSSS
```

The first step to learning to program is to learn the fundamentals. This section will cover the core concepts and basic syntax for the Python language, and hopefully prepare you to begin more complex topics.

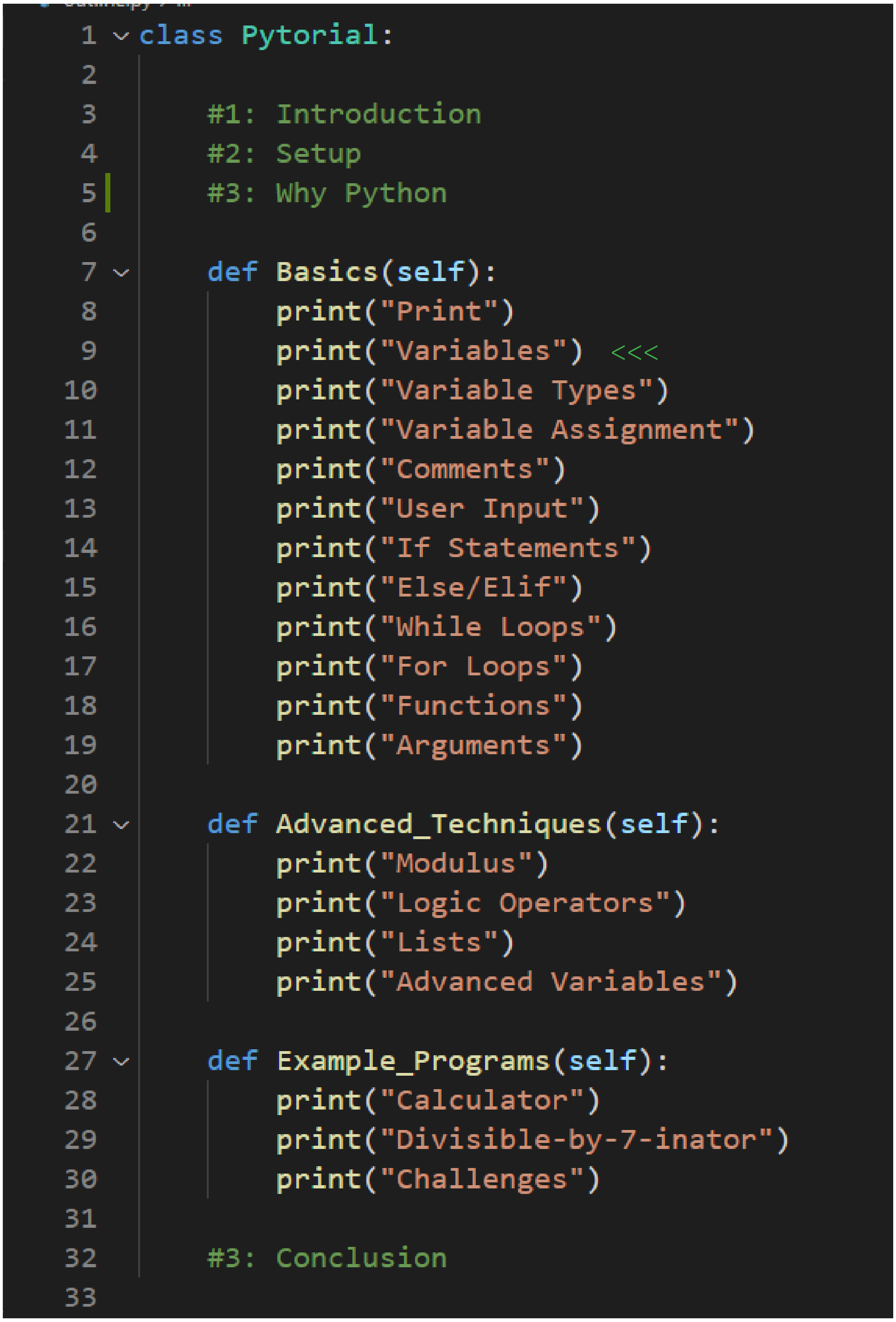


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print") <<<
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

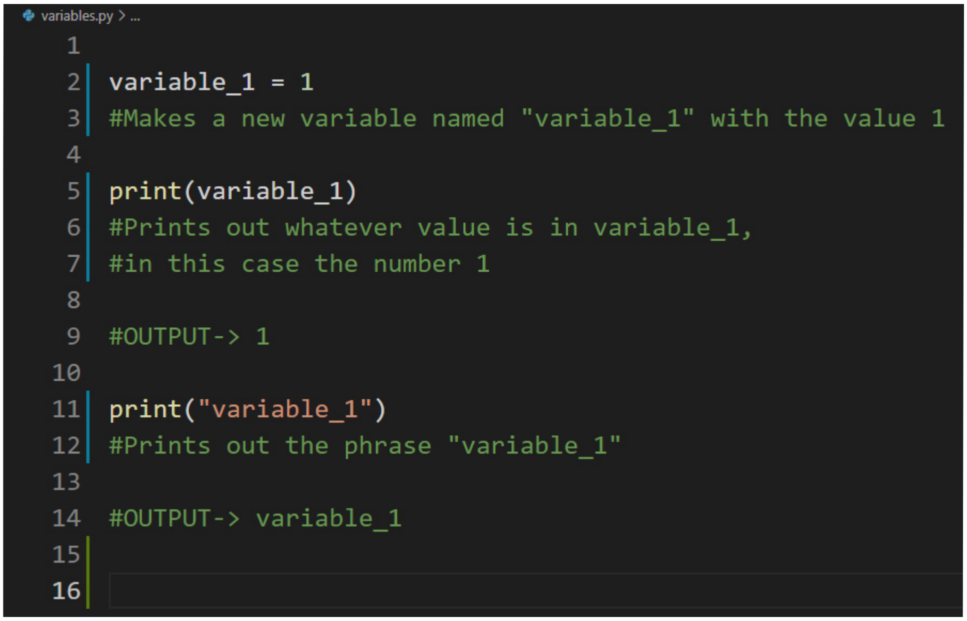

print.py

```
print("Hello World!")
#Prints hello world when run

#OUTPUT-> Hello World
```

Print is the simplest command in any programming language, it simply prints out whatever you put in the parentheses. The message is printed out to the console exactly as given to the print function. In this example our message is put into quotation marks, this tells the computer to read it as just text and not the name of something (we'll cover that in detail later).

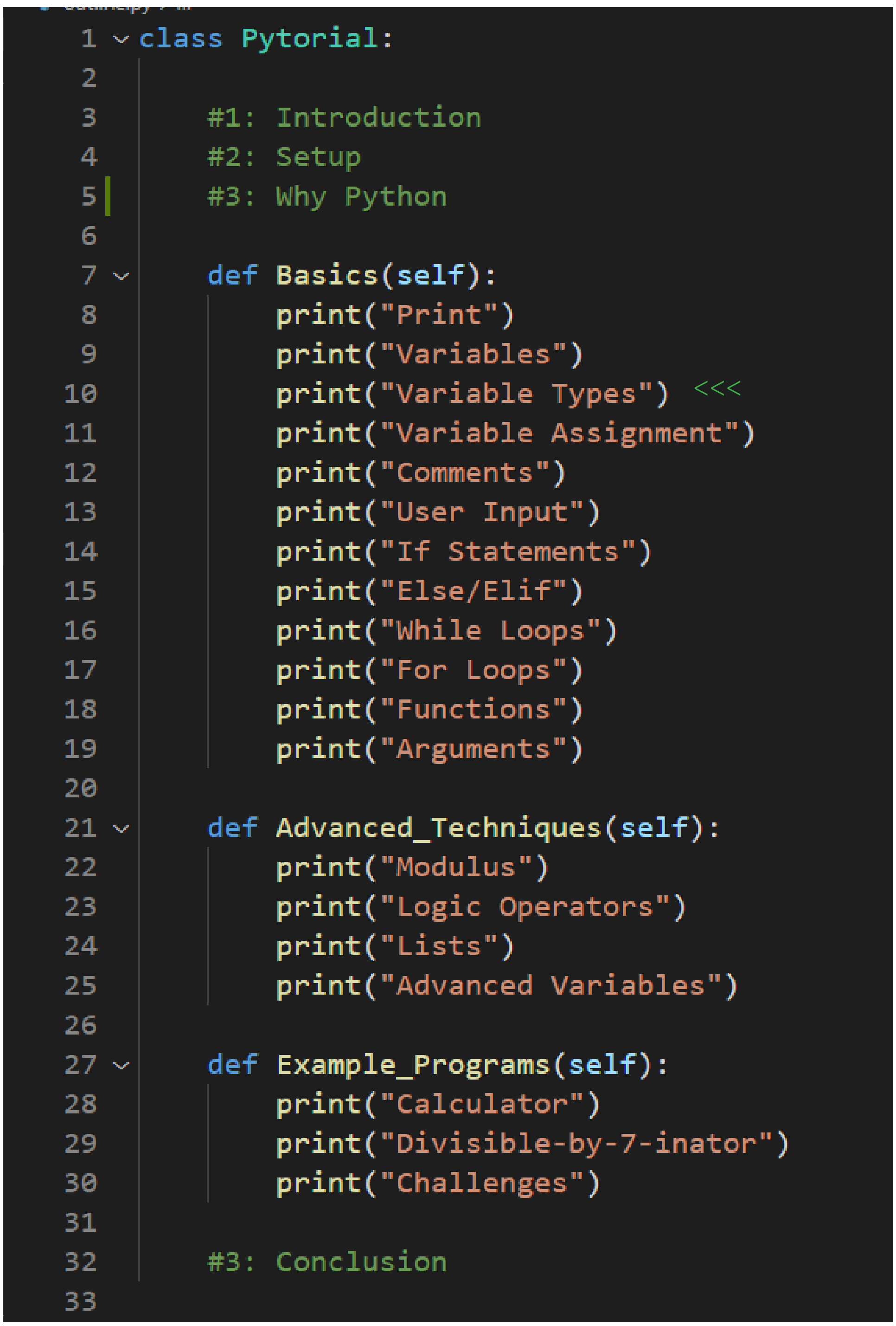


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")  <<<
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion
```

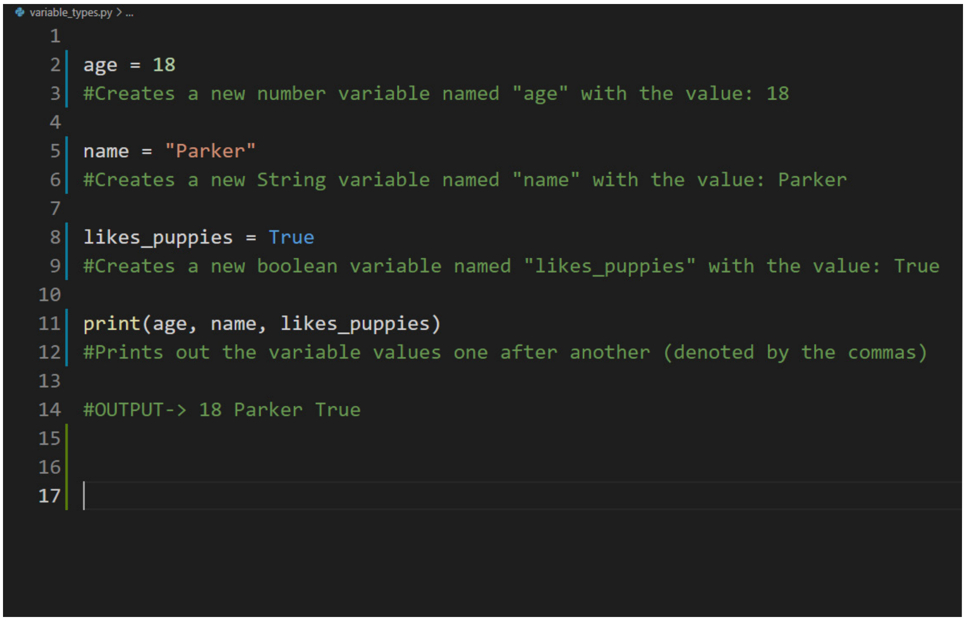

variables.py > ...

```python

variable_1 = 1
#Makes a new variable named "variable_1" with the value 1

print(variable_1)
#Prints out whatever value is in variable_1,
#in this case the number 1

#OUTPUT-> 1

print("variable_1")
#Prints out the phrase "variable_1"

#OUTPUT-> variable_1


```

Now that we know how to output things, we can move on to variables. A variable is a piece of data that we can manipulate and read. We create a variable by typing what we want it to be named, followed by an equals sign, followed by what we want it to be equal to. It can store lots of different kinds of data, but for now we'll stick to just numbers. In this example we make a new variable named "variable_1" and make it equal to 1, now any time we use the word "variable_1" the computer will go find the value of our variable and use that. So when we put "variable_1" into our print statement, it actually prints the number 1 since that's what variable_1 is. We can also still print out the phrase "variable_1" by putting it in quotation marks, which tells the computer to read it literally.



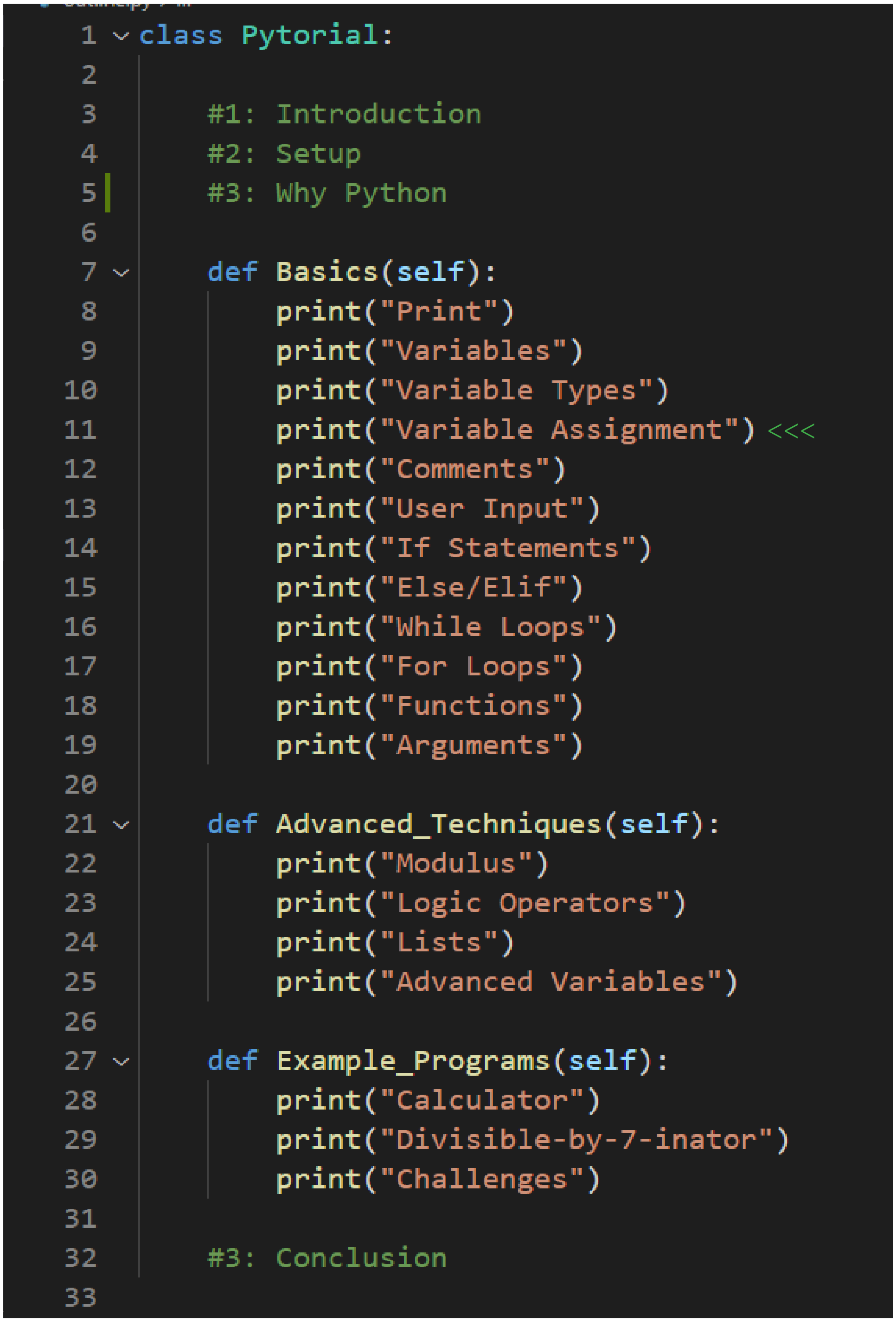
```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types") <<<
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```



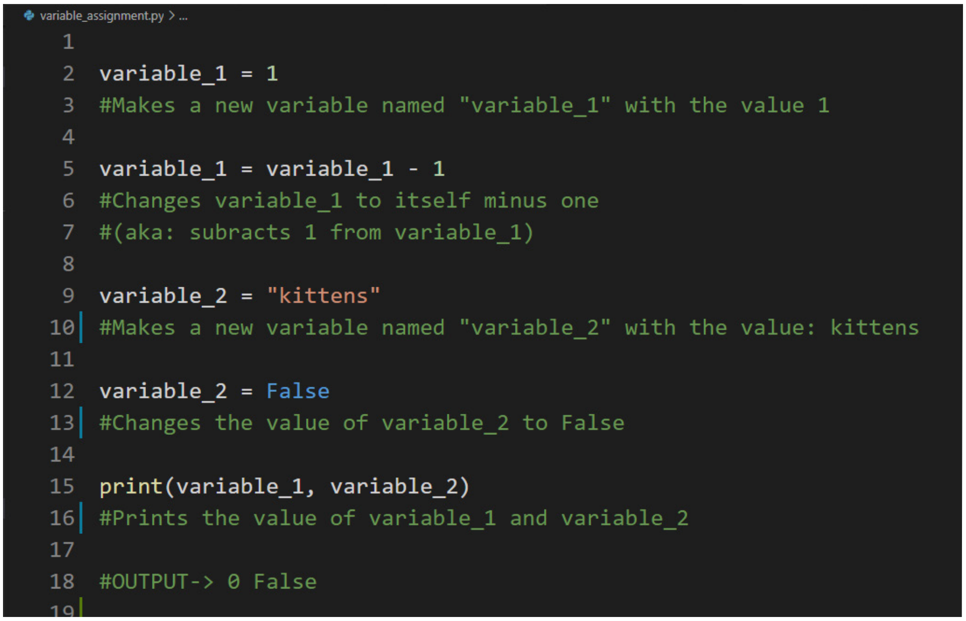
variable_types.py > ...

```python

age = 18
#Creates a new number variable named "age" with the value: 18

name = "Parker"
#Creates a new String variable named "name" with the value: Parker

likes_puppies = True
#Creates a new boolean variable named "likes_puppies" with the value: True

print(age, name, likes_puppies)
#Prints out the variable values one after another (denoted by the commas)

#OUTPUT-> 18 Parker True



```

Now that we know the basics of a variable, we can move on to the specific types of data they can store. There are lots of different variable types, but we are only going to look at the three most basic types: numbers, string literals (or "strings" for short), and booleans. The first type is a number, which is pretty simple to understand, in this example I use a number variable to store my age and print it out. A string is a phrase (or string of characters) denoted by the quotation marks, in this example we make a string variable to store and print out my name. The last basic variable type is the boolean, which is a true/false denoted by the word "True" or "False" (capitalized), in this example we use a boolean to store whether I like puppies or not, and print it out.

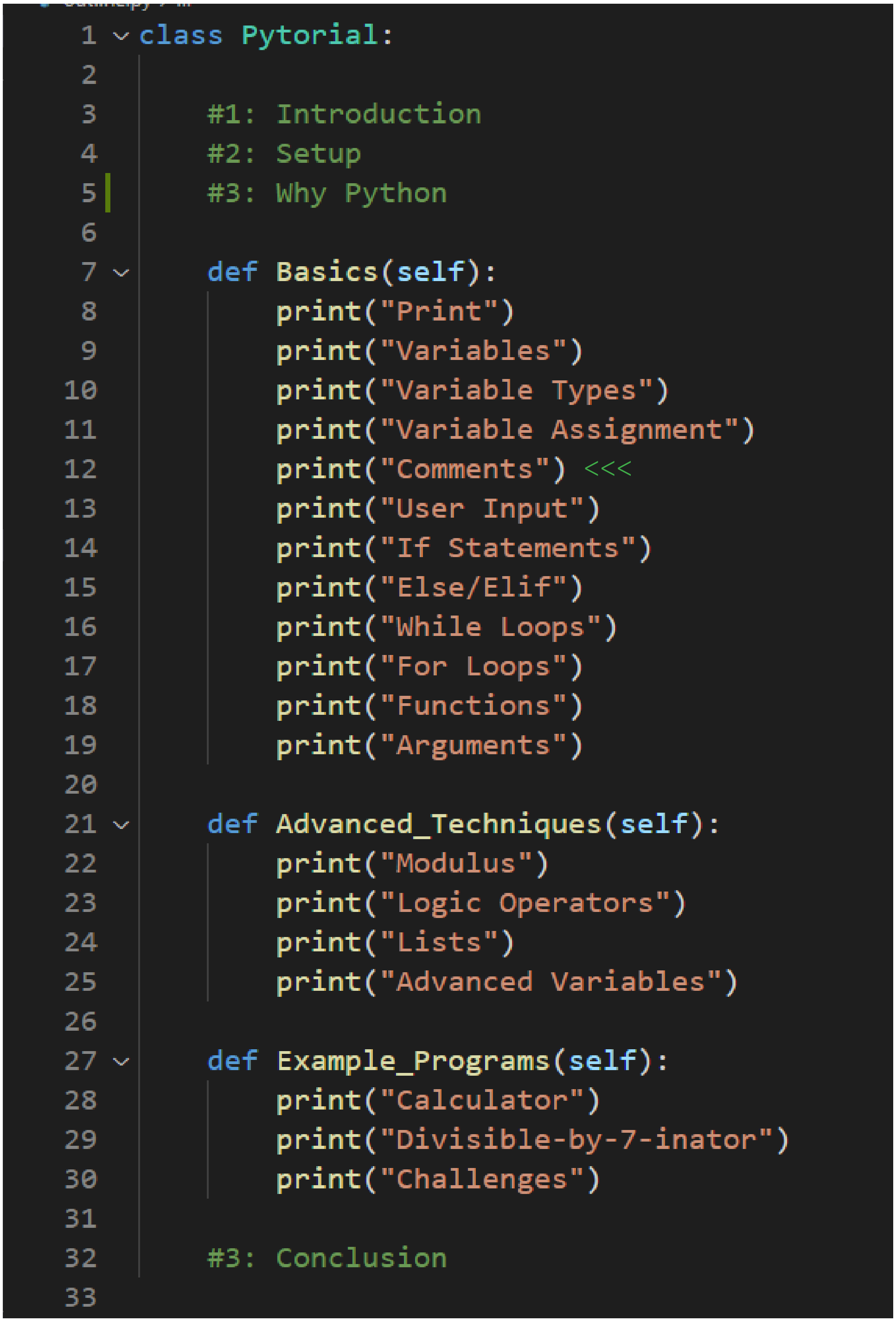


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment") <<<
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

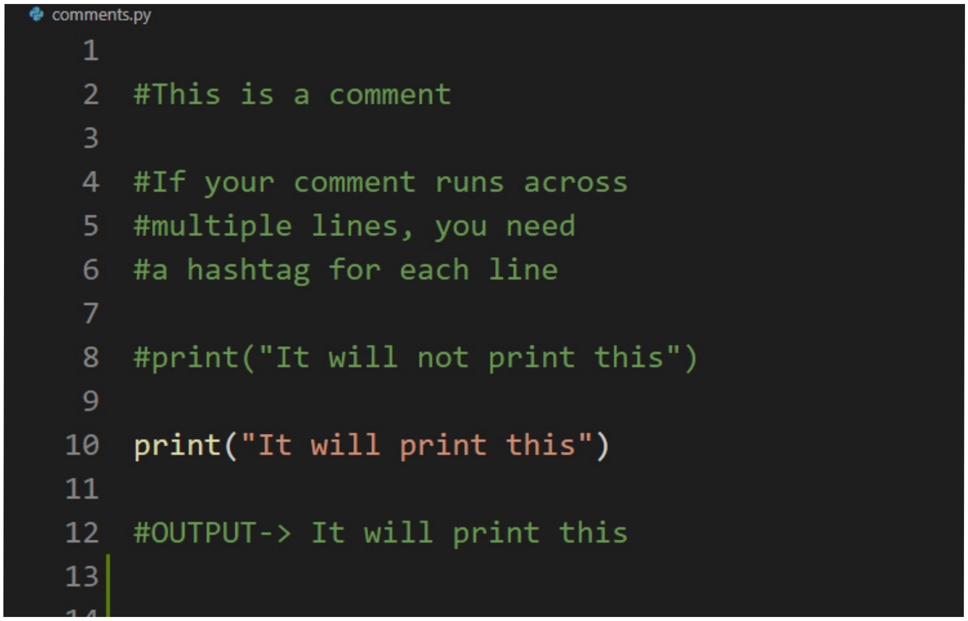

variable_assignment.py > ...

```python

variable_1 = 1
#Makes a new variable named "variable_1" with the value 1

variable_1 = variable_1 - 1
#Changes variable_1 to itself minus one
#(aka: subracts 1 from variable_1)

variable_2 = "kittens"
#Makes a new variable named "variable_2" with the value: kittens

variable_2 = False
#Changes the value of variable_2 to False

print(variable_1, variable_2)
#Prints the value of variable_1 and variable_2

#OUTPUT-> 0 False
```

Variables are not static (or else they would be pretty useless), and can be changed throughout our program. The way to reassign a variable is simply to write the variable name = the new value, just like making a new variable. In the first example we use a common trick, by making variable_1 equal to itself minus 1, we effectively just subtract 1 from it,, this works with any operation (addition, subtraction, etc.). Another property of variables is that their type can be changed at any time (this is almost exclusive to the Python language). In the third example we make a new string named "variable_2" and set it to "kittens", we then set it to the new value of False, making it into a boolean. While this is a great feature, we need to remember to be careful not to accidentally overwrite important variables in programs.

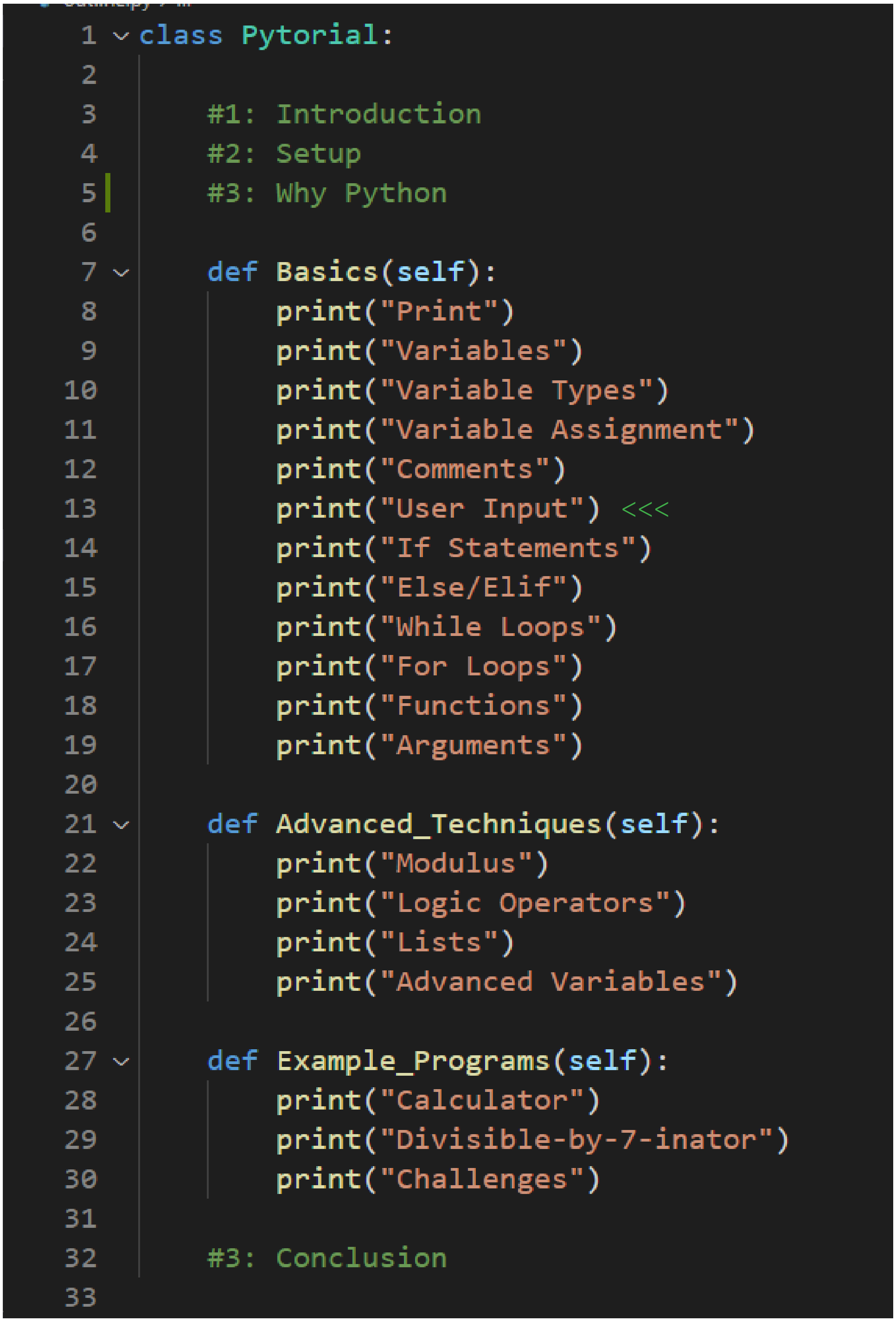


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments") <<<
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

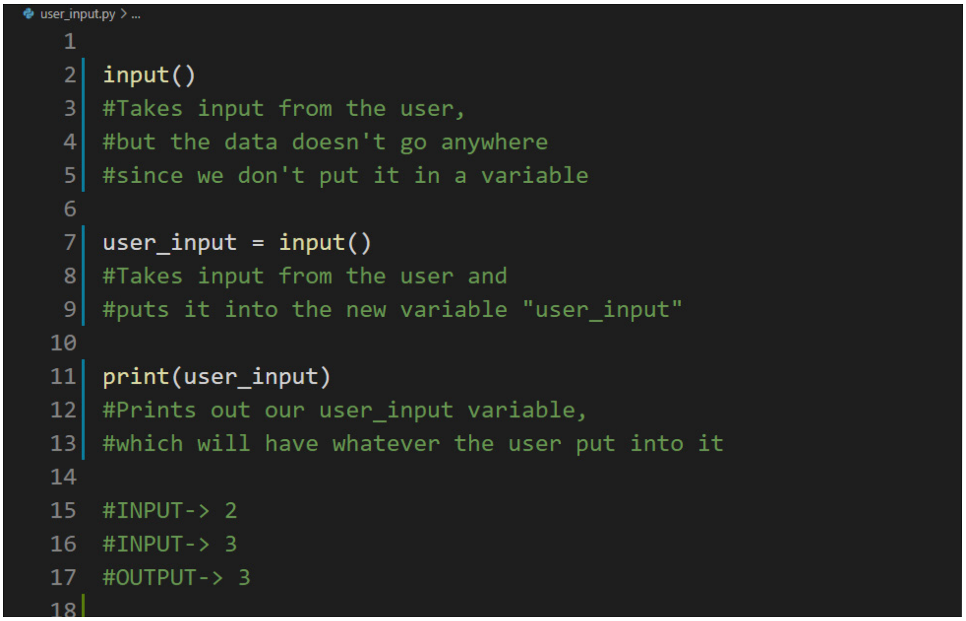


```python

#This is a comment

#If your comment runs across
#multiple lines, you need
#a hashtag for each line

#print("It will not print this")

print("It will print this")

#OUTPUT-> It will print this

```

By now you're probably wondering what that green text all over the place is, those are comments. A comment is a piece of text the computer is told to ignore when running a program, which means anything in a comment will not affect our program. This doesn't do anything for the program itself, but makes it much easier to understand how something works when reading through it, think of these as little sticky notes all over the program explaining what parts of it do. A comment is denoted by a "#", and anything following one on a line will be ignored by the computer. In this example we see that even though that first print statement is completely valid, since a hashtag comes before it on the line the computer completely ignores it. While comments might now seem very important when starting out, they are very useful for remembering how your code works down the line. SERIOUSLY, WRITE COMMENTS.



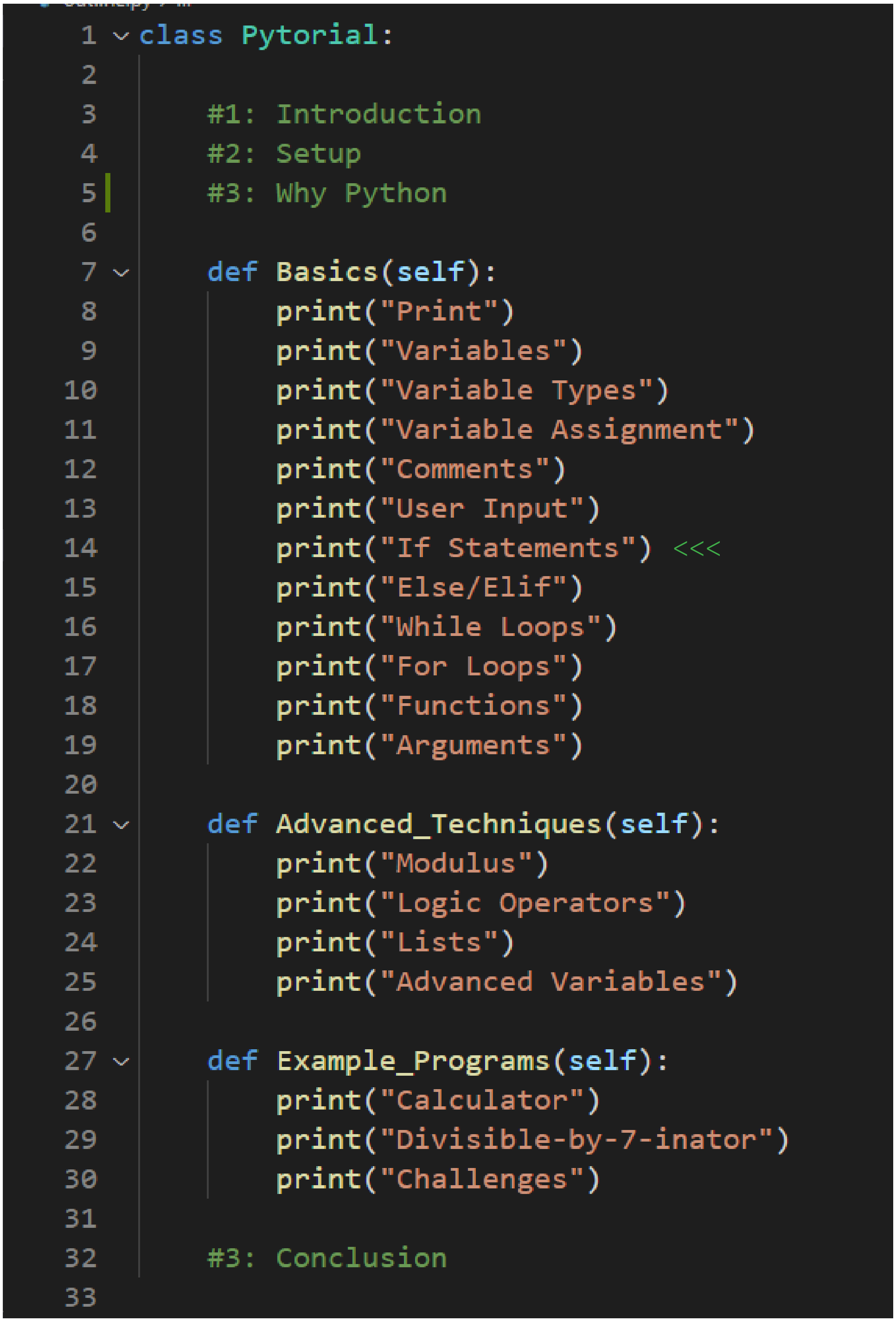
```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input") <<<
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```



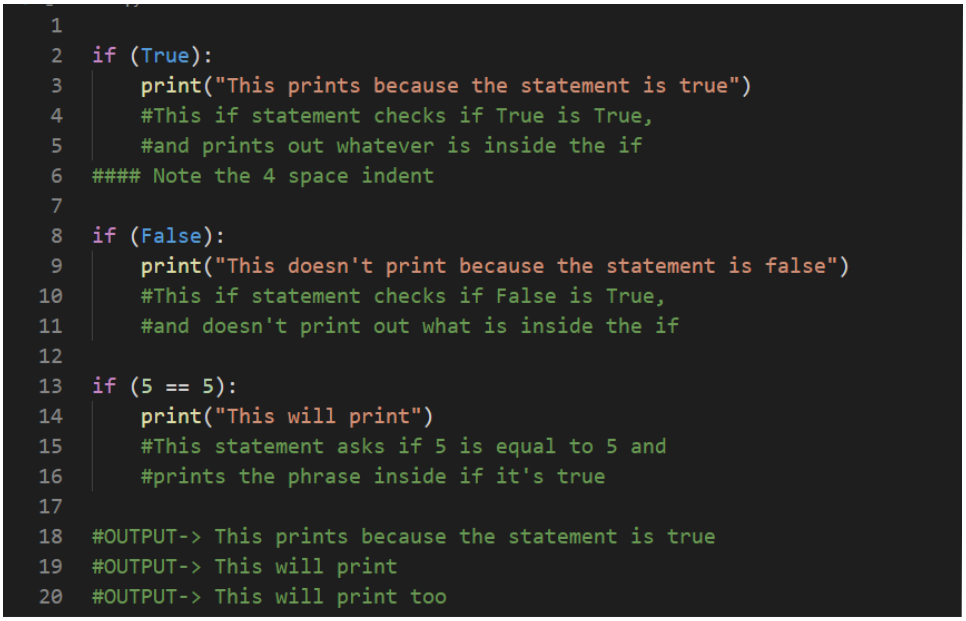
user_input.py > ...

```python

input()
#Takes input from the user,
#but the data doesn't go anywhere
#since we don't put it in a variable

user_input = input()
#Takes input from the user and
#puts it into the new variable "user_input"

print(user_input)
#Prints out our user_input variable,
#which will have whatever the user put into it

#INPUT-> 2
#INPUT-> 3
#OUTPUT-> 3

```

Now that we have the basics of storing and changing data, we can start getting it from the user. The built-in way to get user input in Python is with the input function, which waits for the user to put something in the console. The function acts like a variable with the data the user put into it for the line it's on, before discarding it. In the first example we take the input from the user, but then don't actually store the data in a variable, so it just gets discarded and nothing happens. Instead we need to do what we do in the second example, where we store the user input into a variable to keep it. We then print that data out.

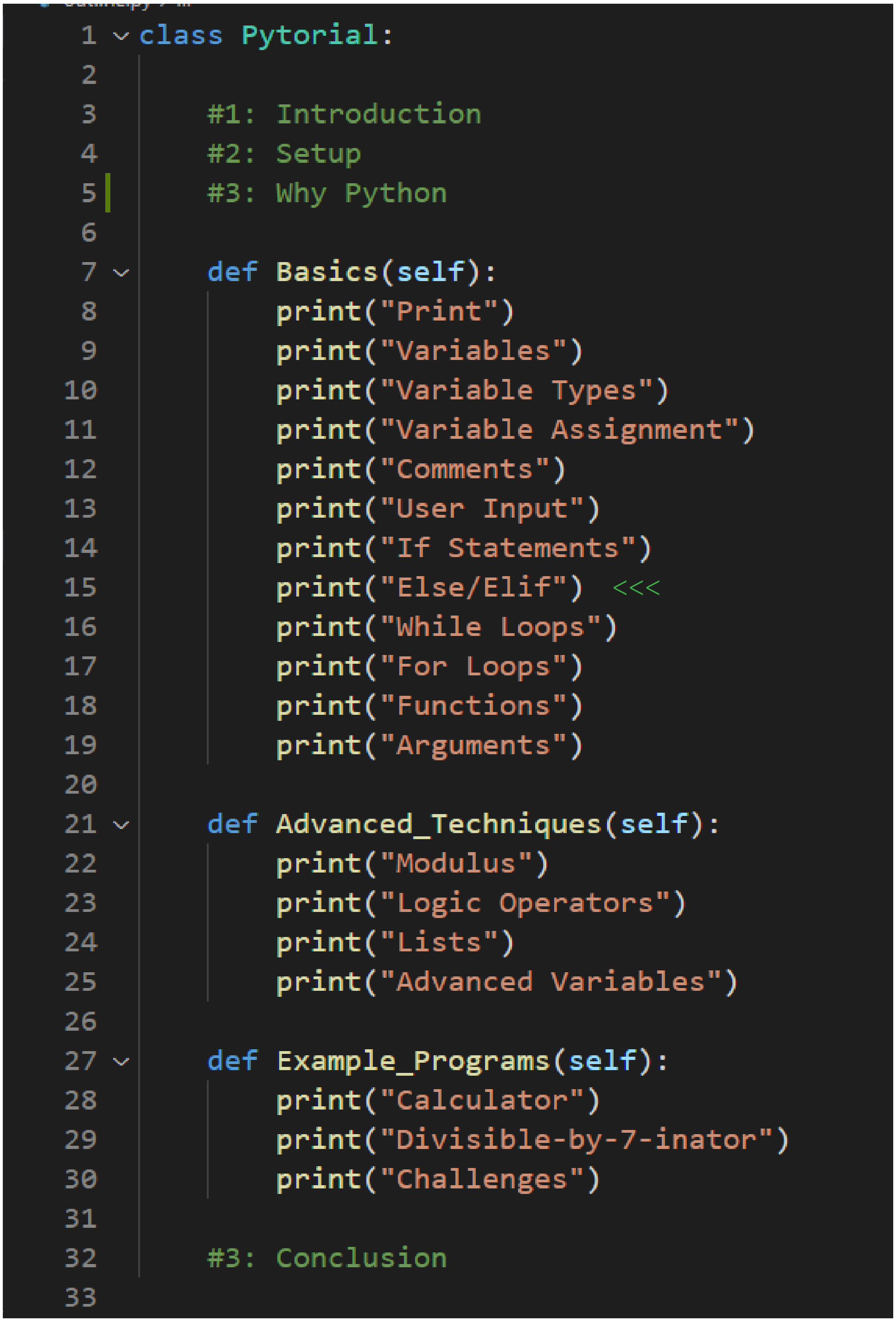


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements") <<<
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

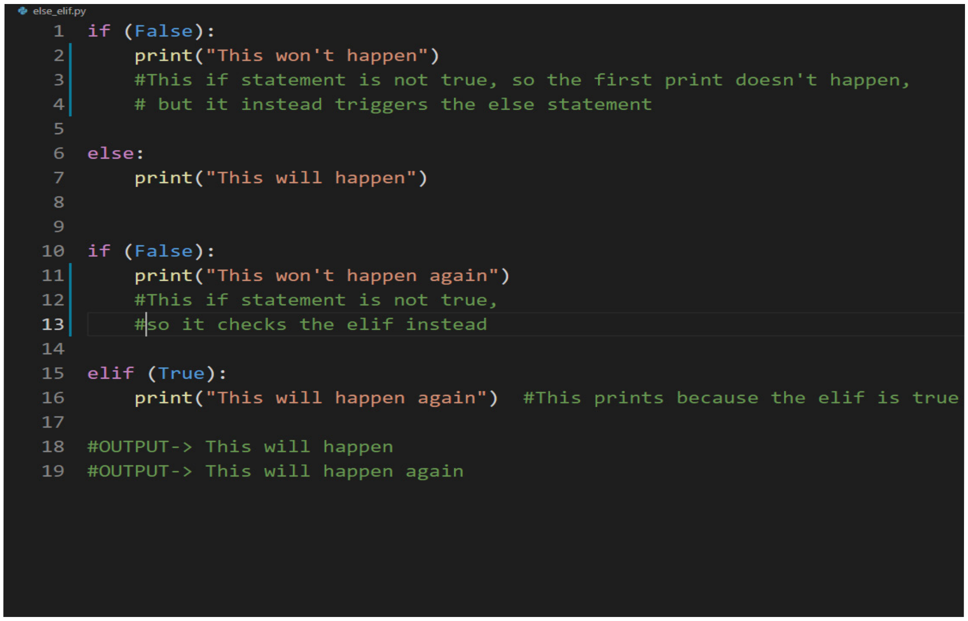


```python

if (True):
    print("This prints because the statement is true")
    #This if statement checks if True is True,
    #and prints out whatever is inside the if
#### Note the 4 space indent

if (False):
    print("This doesn't print because the statement is false")
    #This if statement checks if False is True,
    #and doesn't print out what is inside the if

if (5 == 5):
    print("This will print")
    #This statement asks if 5 is equal to 5 and
    #prints the phrase inside if it's true

#OUTPUT-> This prints because the statement is true
#OUTPUT-> This will print
#OUTPUT-> This will print too
```

While playing with numbers is fun, we need a way to make decisions, which is where if statements come in. An if statement is a conditional, if what is inside the parentheses is true, it will do whatever is inside the statement, if it is not true the computer will just skip over it. An if statement is denoted by the word if, followed by the condition in parentheses, followed by a colon, and then anything within the if statement goes below it and indented 4 spaces (or one press of the tab key). Inside the parentheses we use two equals ("==") to compare two things, remember that one equals is telling the computer two things are equal, while two equals is asking the computer if two things are equal. We can also use an exclamation point and equals ("!=") to represent not equal to.

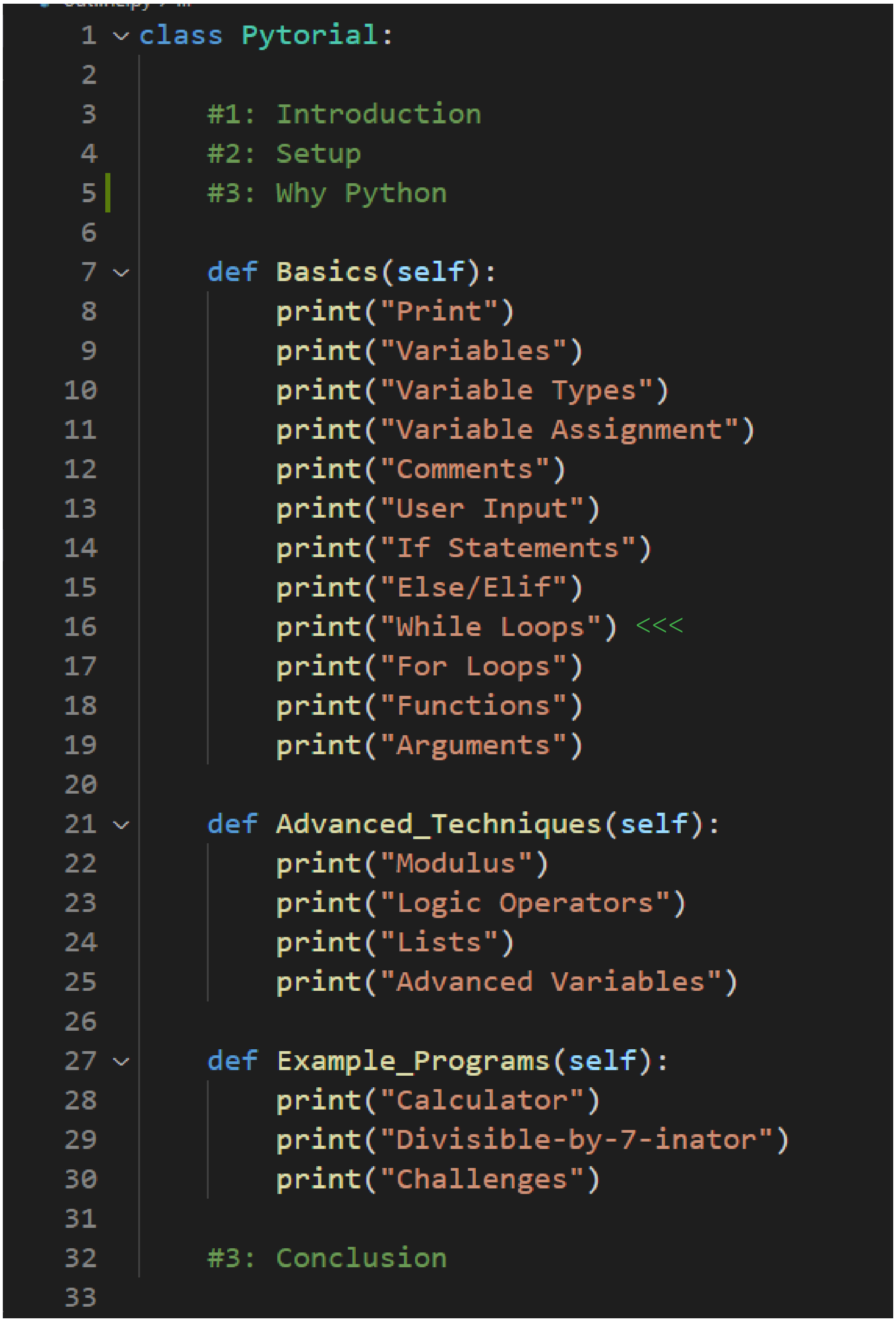


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")  <<<
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

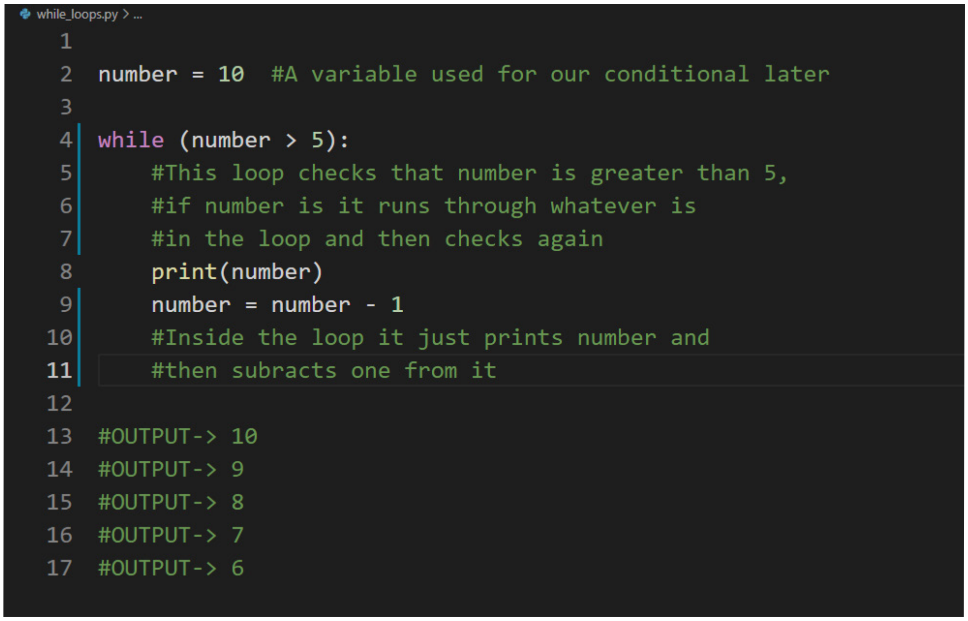

else_elif.py

```python
if (False):
    print("This won't happen")
    #This if statement is not true, so the first print doesn't happen,
    # but it instead triggers the else statement

else:
    print("This will happen")


if (False):
    print("This won't happen again")
    #This if statement is not true,
    #so it checks the elif instead

elif (True):
    print("This will happen again")  #This prints because the elif is true

#OUTPUT-> This will happen
#OUTPUT-> This will happen again
```

Another key to decision making in programming is the else and else if (or "elif" for short) statements. These are an extension of the if statement and work exactly as their names suggest. You can put an else statement after an if statement and it will only happen if the if statement isn't true. The second addition to if statements is the elif, which is a hybrid between if and else. It will check if its condition is true and do whatever is inside it if so (like an if), but only if the if statement before it wasn't true (like an else). You can chain any number of elif statements after one another as long as it starts with a standard if statement, and you can only have one else statement at the end. The formatting for else and elif statements is just like that of if statements, except else statements don't have parentheses.



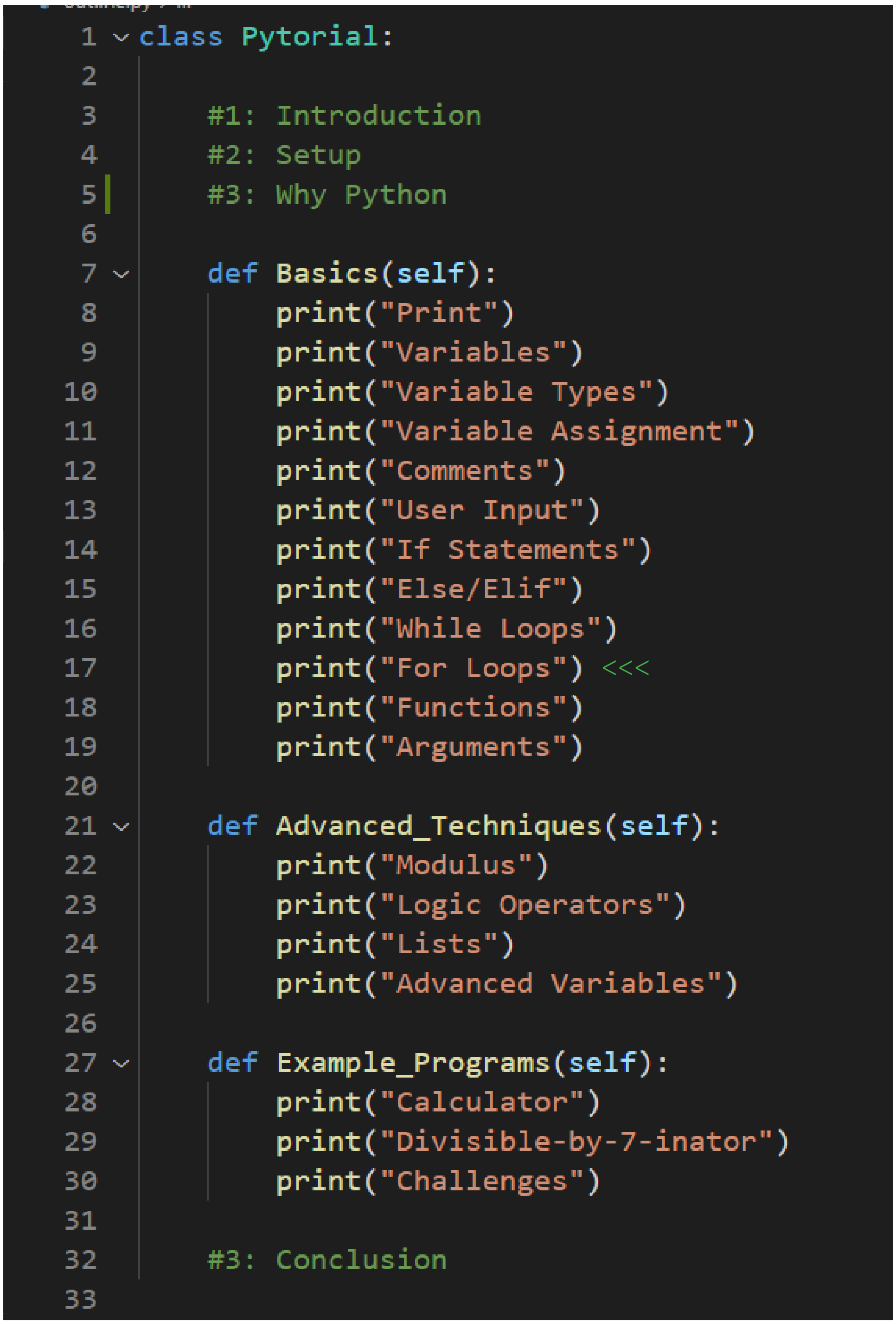
```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops") <<<
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```



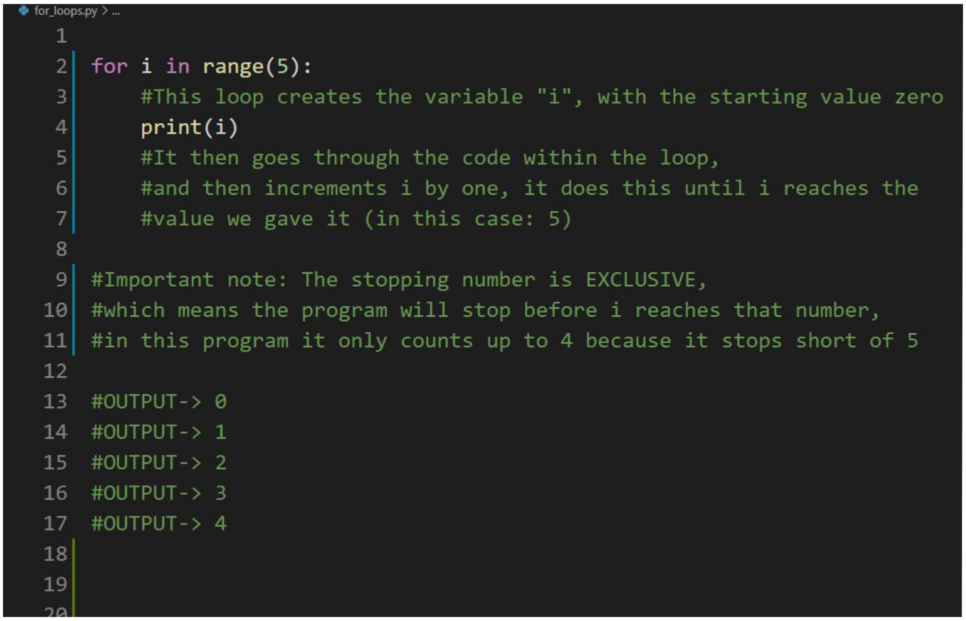
while_loops.py > ...

```python

number = 10  #A variable used for our conditional later

while (number > 5):
    #This loop checks that number is greater than 5,
    #if number is it runs through whatever is
    #in the loop and then checks again
    print(number)
    number = number - 1
    #Inside the loop it just prints number and
    #then subracts one from it

#OUTPUT-> 10
#OUTPUT-> 9
#OUTPUT-> 8
#OUTPUT-> 7
#OUTPUT-> 6
```

Another fundamental part of any programming language is the loop, a piece of code that repeats itself until a condition is met. The first and simplest kind of loop is the while loop, which acts sort of like a repeating if statement. If the condition in parentheses is true, the computer will do whatever is inside the while loop and then check the conditional again. This process will repeat forever until the condition is no longer true, at which point the computer will move on. It can be very easy to accidentally create infinite while loops, which can crash your program, so be careful when using them.

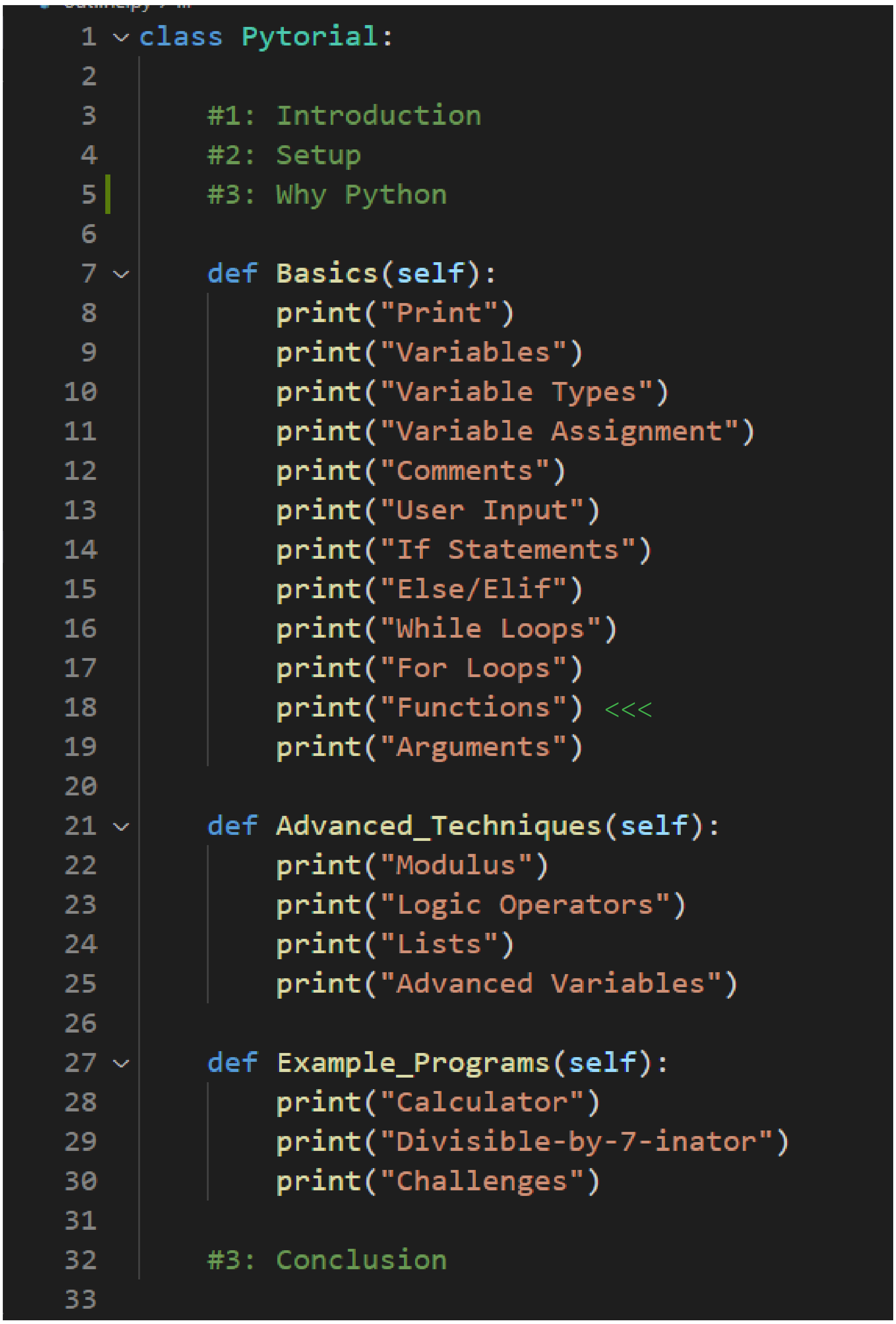


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops") <<<
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

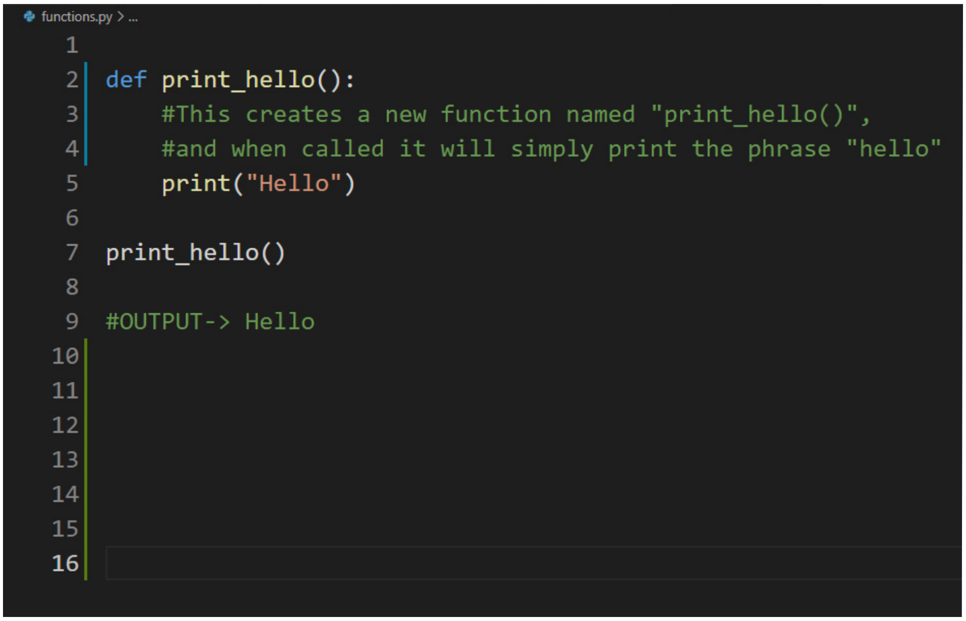


```python

for i in range(5):
    #This loop creates the variable "i", with the starting value zero
    print(i)
    #It then goes through the code within the loop,
    #and then increments i by one, it does this until i reaches the
    #value we gave it (in this case: 5)

#Important note: The stopping number is EXCLUSIVE,
#which means the program will stop before i reaches that number,
#in this program it only counts up to 4 because it stops short of 5

#OUTPUT-> 0
#OUTPUT-> 1
#OUTPUT-> 2
#OUTPUT-> 3
#OUTPUT-> 4
```

The second kind of loop is the for loop, which repeats itself a given number of times before ending. The format for a for loop is: for [variable] in range([number]):, the loop will create a variable with whatever name you give it and the value of zero, and then go through whatever is in the loop, and count the variable up one after each pass. This pattern will continue until the variable you gave it reaches the number you gave it. In this example we create a loop that prints out i each pass, so it prints out a set of numbers counting up.

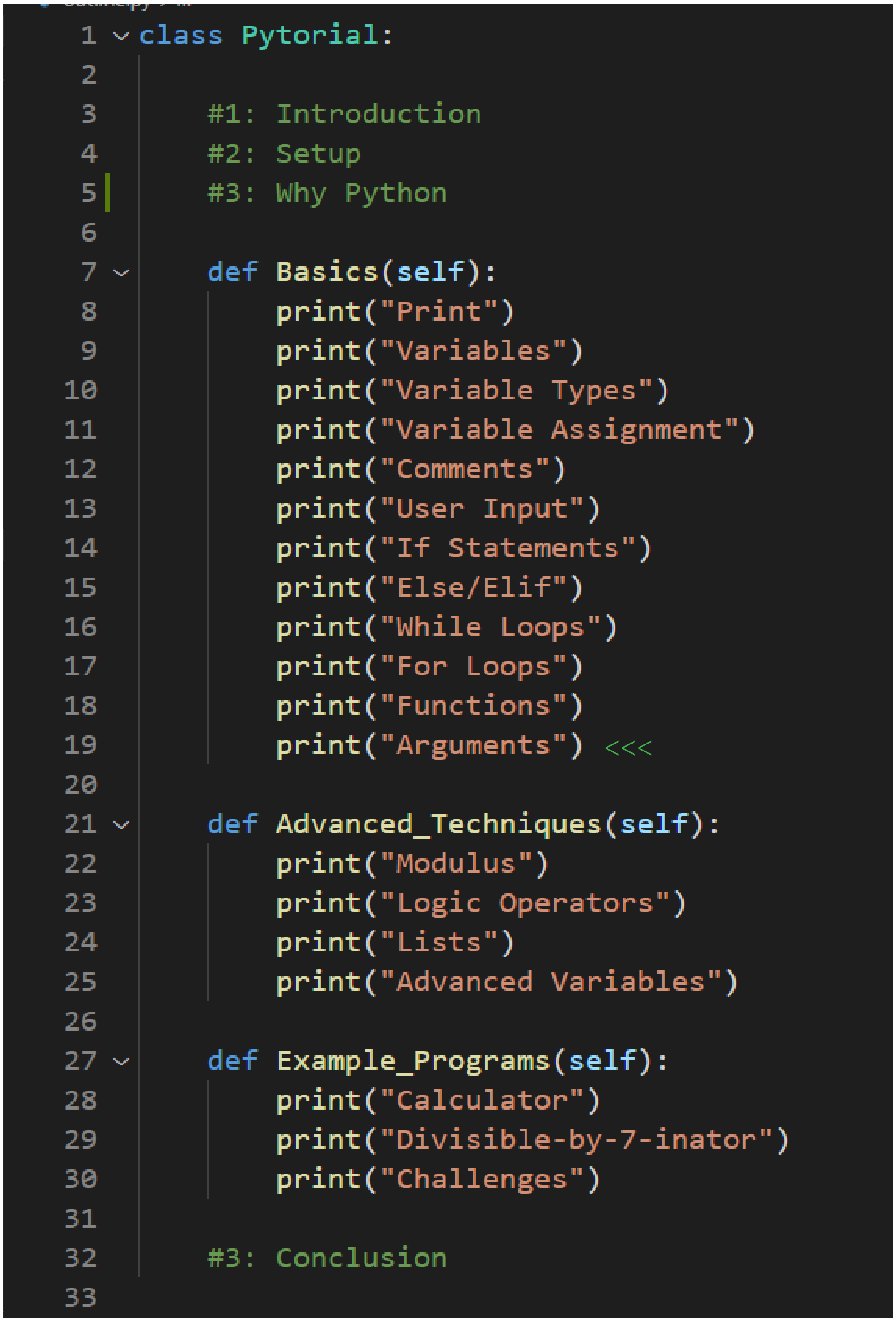


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions") <<<
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

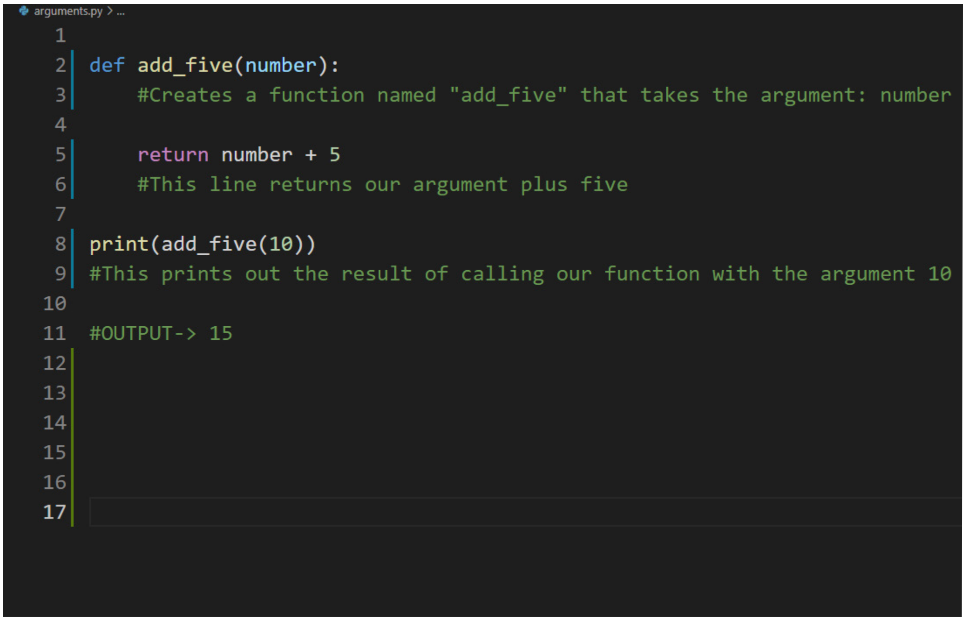


```python

def print_hello():
    #This creates a new function named "print_hello()",
    #and when called it will simply print the phrase "hello"
    print("Hello")

print_hello()

#OUTPUT-> Hello
```

Now that we have our logic operations out of the way, we can get to the really cool stuff, functions (or methods). A function is a chunk of code that can be called at any point in your program, this is helpful for two things. The first major use is for easy organization, since everything is bundled together in neat packages, and the other use is for repeating code. In this example we create a function named "print_hello" and then call it immediately after. Functions are denoted by the parentheses after their name, which tells the computer it's a function and not a variable. To create a function you type: def [function_name]():, and then put your content inside it like any other statement.

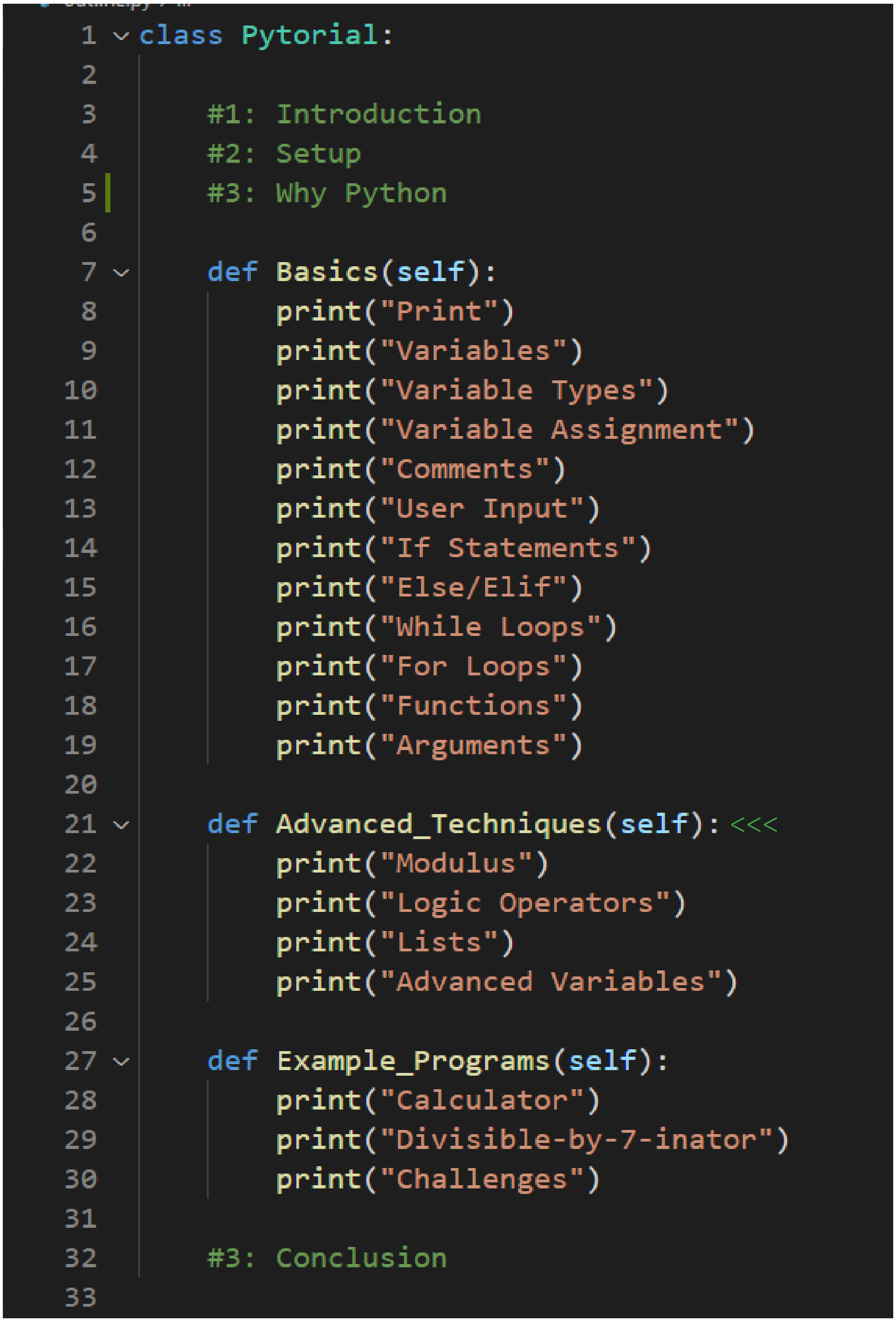


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments") <<<

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

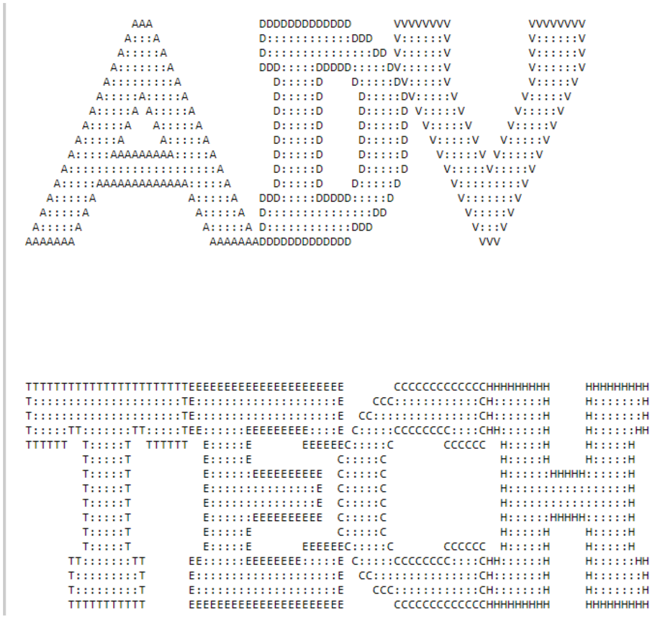

arguments.py > ...

```python

def add_five(number):
    #Creates a function named "add_five" that takes the argument: number

    return number + 5
    #This line returns our argument plus five

print(add_five(10))
#This prints out the result of calling our function with the argument 10

#OUTPUT-> 15
```

While the parentheses are a nice way of telling functions from variables, they're also an important tool for making functions, they contain arguments. Think of an argument as a piece of data passed down to a given function for it to do something with. To add an argument to a function simply put a variable name in the parentheses when defining the function and it will create a new variable with the value of the argument when the function is called. In this example we make a new function that takes the argument number, it then adds 5 to it and then returns the new value. We've actually seen how return works before, since the built-in input() function returns the user input. Then when we call our function we give it the number 10 in the parentheses, so it takes 10 as the argument and returns that plus 5, or 15, then that number is printed. A function can take as many arguments as you want by separating them with commas, but keep in mind that any argument you add will need to be given every time the function is called.

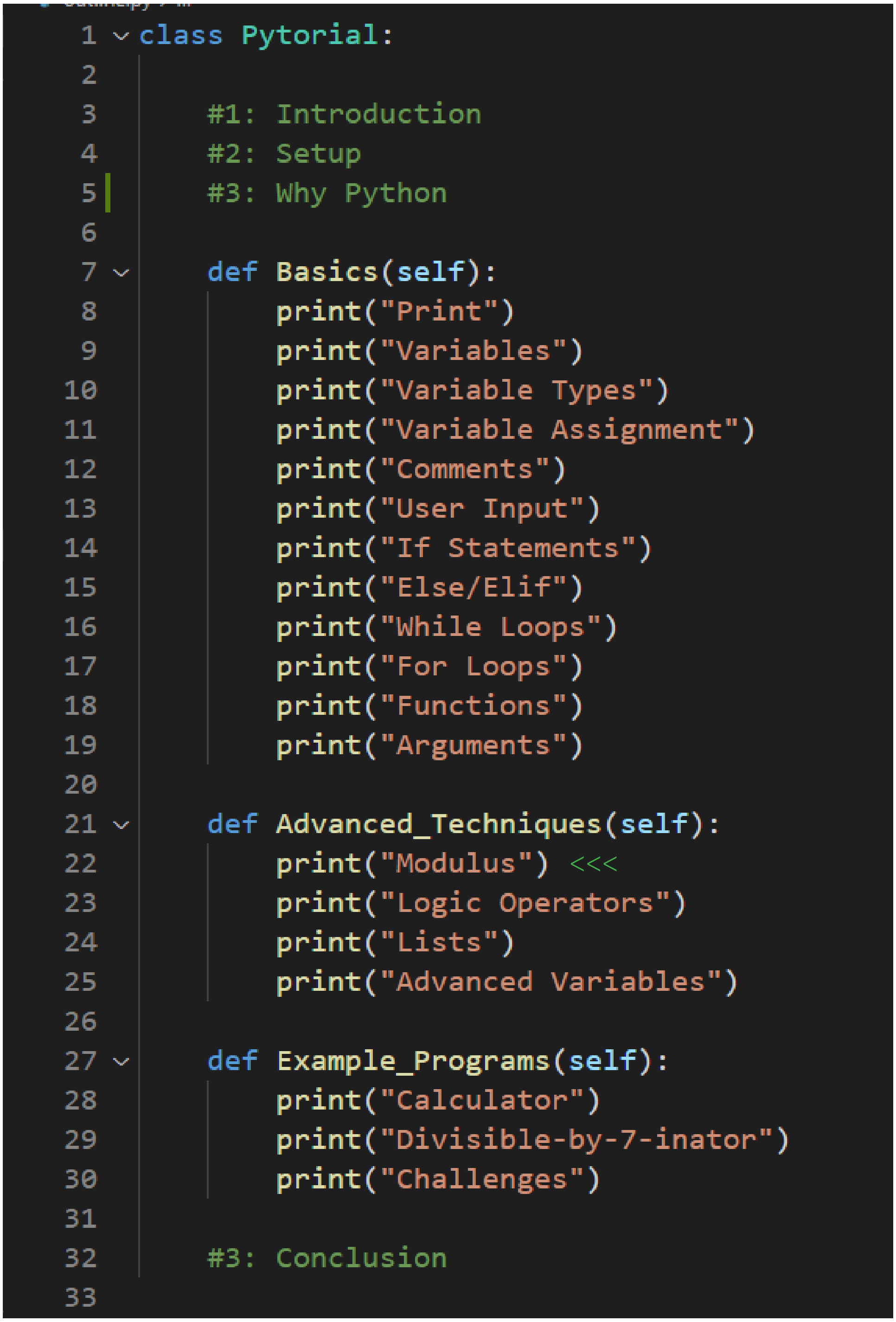


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self): <<<
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

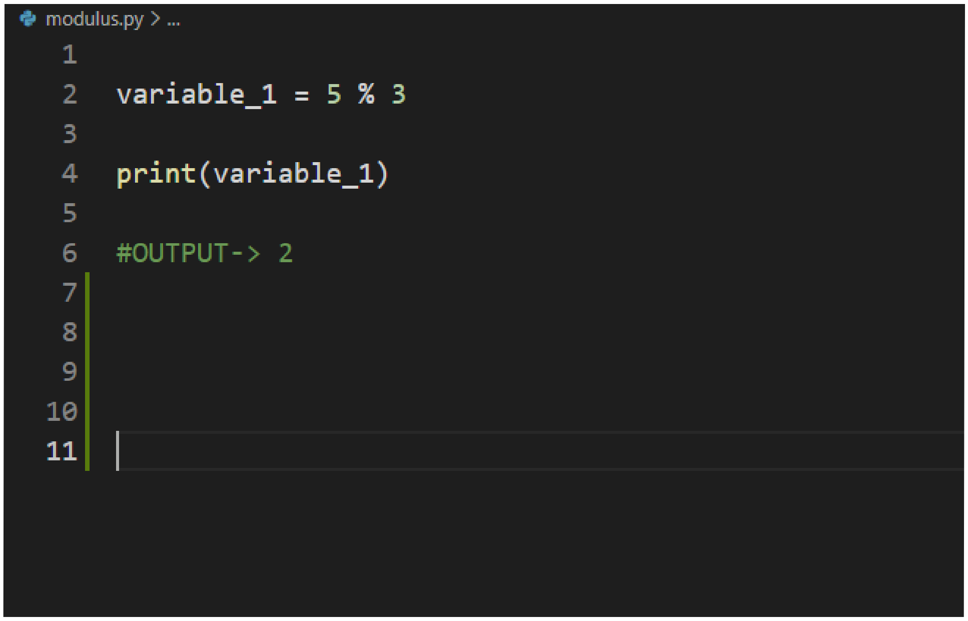


```
               AAA               DDDDDDDDDDDDD      VVVVVVVV           VVVVVVVV
              A:::A              D::::::::::::DDD   V::::::V           V::::::V
             A:::::A             D:::::::::::::::DD V::::::V           V::::::V
            A:::::::A            DDD:::::DDDDD:::::DV::::::V           V::::::V
           A:::::::::A             D:::::D    D:::::DV:::::V           V:::::V
          A:::::A:::::A            D:::::D     D:::::DV:::::V         V:::::V
         A:::::A A:::::A           D:::::D     D:::::D V:::::V       V:::::V
        A:::::A   A:::::A          D:::::D     D:::::D  V:::::V     V:::::V
       A:::::A     A:::::A         D:::::D     D:::::D   V:::::V   V:::::V
      A:::::AAAAAAAAA:::::A        D:::::D     D:::::D    V:::::V V:::::V
     A:::::::::::::::::::::A       D:::::D     D:::::D     V:::::V:::::V
    A:::::AAAAAAAAAAAAA:::::A      D:::::D    D:::::D       V:::::::::V
   A:::::A             A:::::A   DDD:::::DDDDD:::::D         V:::::::V
  A:::::A               A:::::A  D:::::::::::::::DD           V:::::V
 A:::::A                 A:::::A D::::::::::::DDD              V:::V
AAAAAAA                   AAAAAAADDDDDDDDDDDDD                  VVV




TTTTTTTTTTTTTTTTTTTTTTTEEEEEEEEEEEEEEEEEEEEEE       CCCCCCCCCCCCCHHHHHHHHH     HHHHHHHHH
T:::::::::::::::::::::TE::::::::::::::::::::E    CCC::::::::::::CH:::::::H     H:::::::H
T:::::::::::::::::::::TE::::::::::::::::::::E  CC:::::::::::::::CH:::::::H     H:::::::H
T:::::TT:::::::TT:::::TEE::::::EEEEEEEEE::::E C:::::CCCCCCCC::::CHH::::::H     H::::::HH
TTTTTT  T:::::T  TTTTTT  E:::::E       EEEEEEC:::::C       CCCCCC  H:::::H     H:::::H
        T:::::T          E:::::E             C:::::C                H:::::H     H:::::H
        T:::::T          E::::::EEEEEEEEEE   C:::::C                H::::::HHHHH::::::H
        T:::::T          E:::::::::::::::E   C:::::C                H:::::::::::::::::H
        T:::::T          E:::::::::::::::E   C:::::C                H:::::::::::::::::H
        T:::::T          E::::::EEEEEEEEEE   C:::::C                H::::::HHHHH::::::H
        T:::::T          E:::::E             C:::::C                H:::::H     H:::::H
        T:::::T          E:::::E       EEEEEEC:::::C       CCCCCC  H:::::H     H:::::H
      TT:::::::TT      EE::::::EEEEEEEE:::::E C:::::CCCCCCCC::::CHH::::::H     H::::::HH
      T:::::::::T      E::::::::::::::::::::E  CC:::::::::::::::CH:::::::H     H:::::::H
      T:::::::::T      E::::::::::::::::::::E    CCC::::::::::::CH:::::::H     H:::::::H
      TTTTTTTTTTT      EEEEEEEEEEEEEEEEEEEEEE       CCCCCCCCCCCCCHHHHHHHHH     HHHHHHHHH
```

With a solid understanding of the basics of Python, we can now move on to more niche and specific things. Some of these tips and techniques don’t require a full page, so some of them will be grouped together.

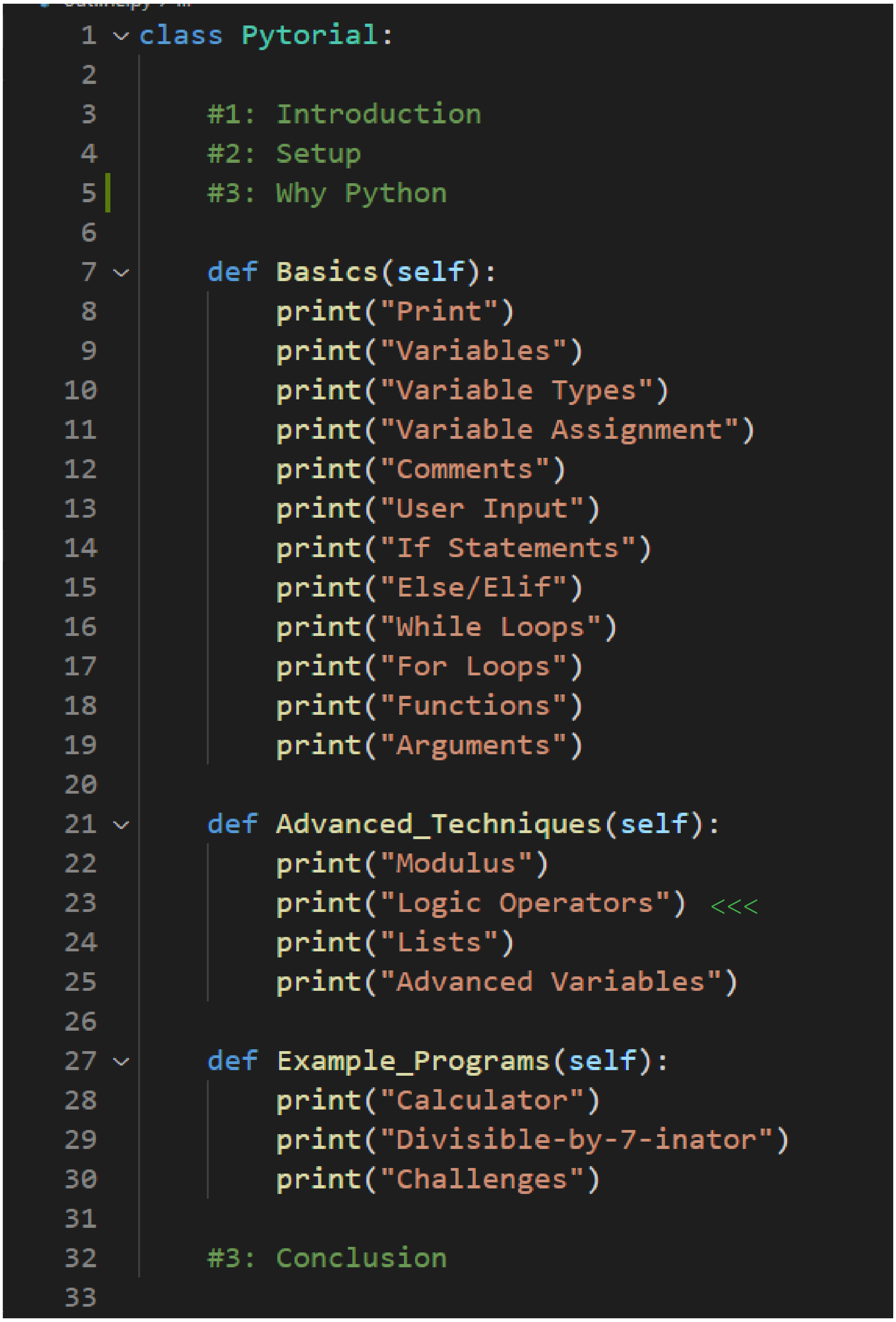


```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus") <<<
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

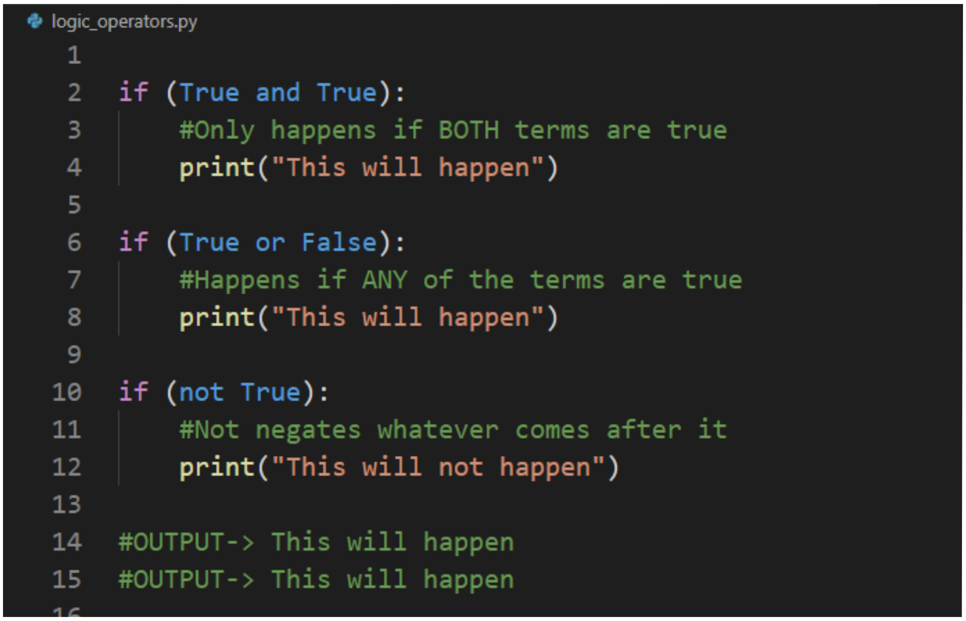


```python

variable_1 = 5 % 3

print(variable_1)

#OUTPUT-> 2
```

The first niche tip we'll look at is the modulus operation, represented in Python by the % sign. It can be used just like the other operations (+,-,*,/) and it will return the remainder of dividing the two numbers. For example: 7 % 2 = 1, 10 % 5 = 0, 12 % 8 = 4. This operation might seem to be useless, but it's actually very helpful in various uses, on common use to see if a number is even or odd. If our number mod 2 returns 0, it's even, if not then it's odd.

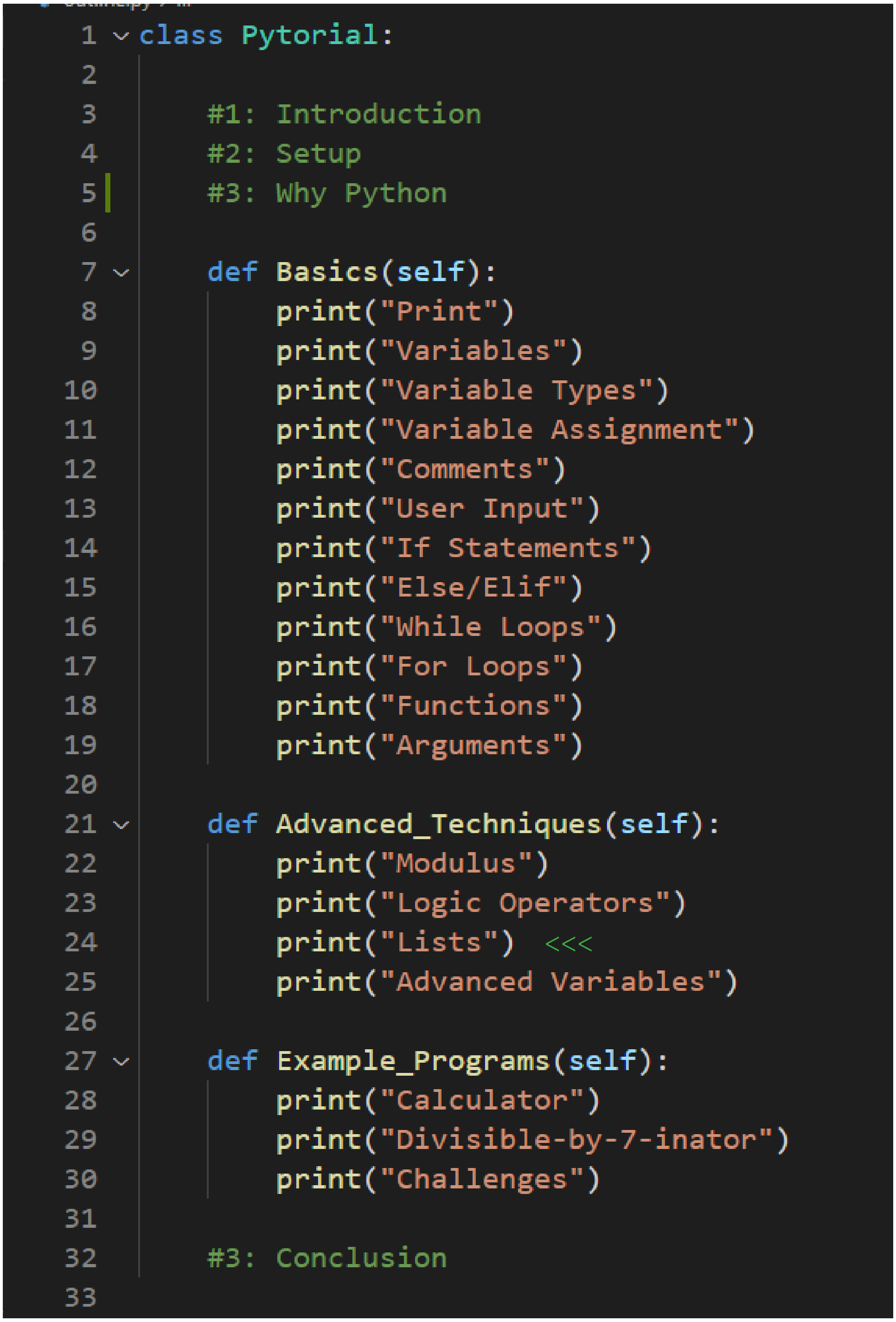


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators") <<<
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion
```

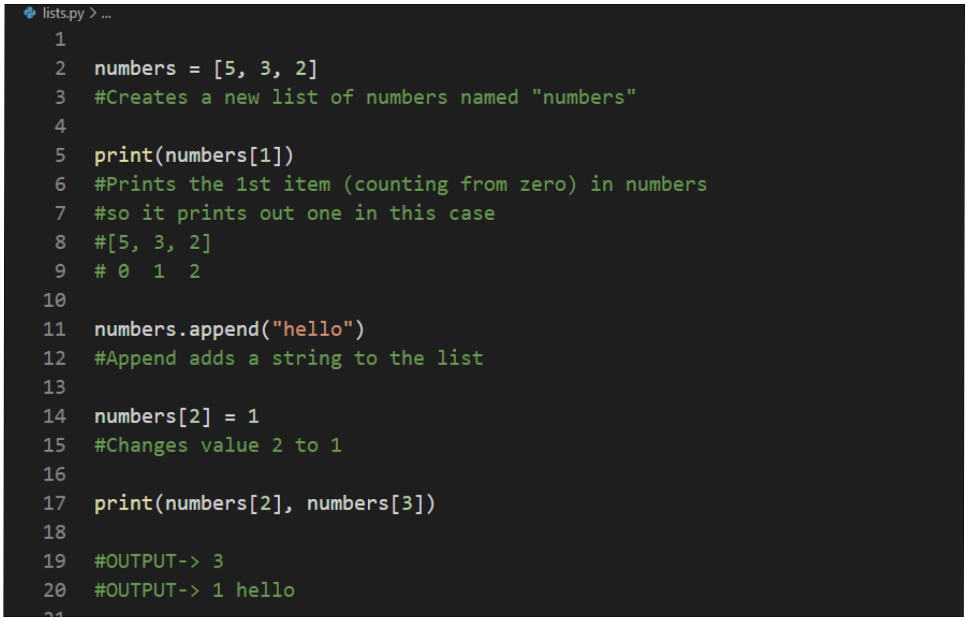

logic_operators.py

```python

if (True and True):
    #Only happens if BOTH terms are true
    print("This will happen")

if (True or False):
    #Happens if ANY of the terms are true
    print("This will happen")

if (not True):
    #Not negates whatever comes after it
    print("This will not happen")

#OUTPUT-> This will happen
#OUTPUT-> This will happen
```

The second advanced topic we're going to cover is the logic operators, which help make more complex logic statements. The first logic operator is and, which will be true only if both the terms on either side of it are true. The second is or, which will be true if either or both of it's terms are true. The final is not, which negates whatever follows it, so typing "not True" will evaluate to False. These statements can be chained together to create complex logic operations, and you use parentheses just like in math to split equations up into separate parts.

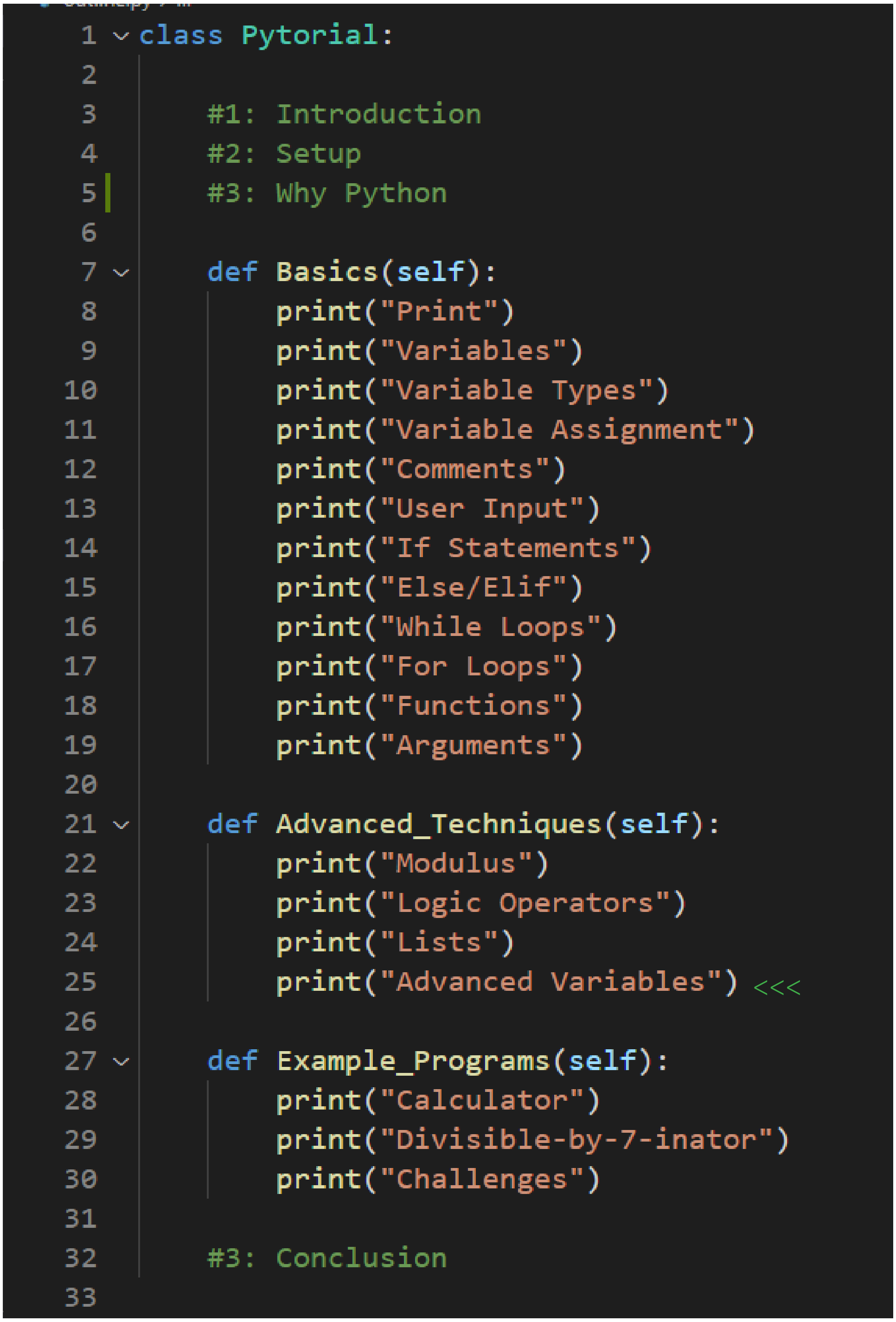


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")  <<<
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion
```

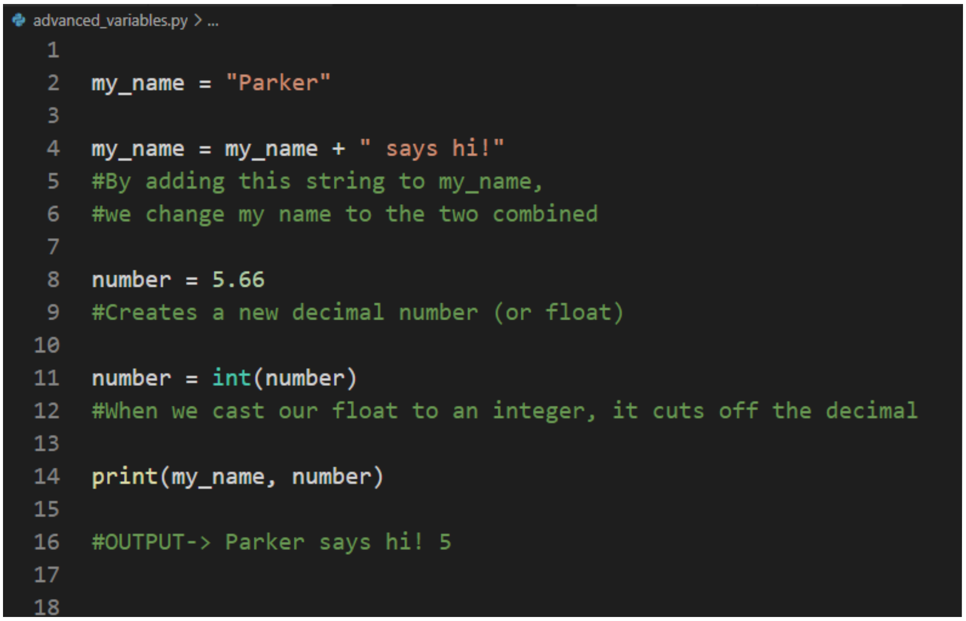


```python

numbers = [5, 3, 2]
#Creates a new list of numbers named "numbers"

print(numbers[1])
#Prints the 1st item (counting from zero) in numbers
#so it prints out one in this case
#[5, 3, 2]
# 0  1  2

numbers.append("hello")
#Append adds a string to the list

numbers[2] = 1
#Changes value 2 to 1

print(numbers[2], numbers[3])

#OUTPUT-> 3
#OUTPUT-> 1 hello
```

A list (or array) is a bunch of variables grouped together into a list. This list has its own name, and each item in the list can be accessed by its index (or place) in the list. An important note is that in programming we always start counting from 0 not 1, so in our example list the number 5 is item 0, and the number 3 is item 1. So when we print out item one it prints the second item in the list, since item 0 would be the first. We can also change the data in lists by changing the items like individual variables, or with some built in functions. One such function is append, which will add whatever value you give it to the end of your list, but there are many more you can look up. We can also mix data types in lists, so it can contain strings and numbers at the same time, like in our second example. Finally, you can use the brackets to access individual characters in strings as if they're a list, since they effectively are.



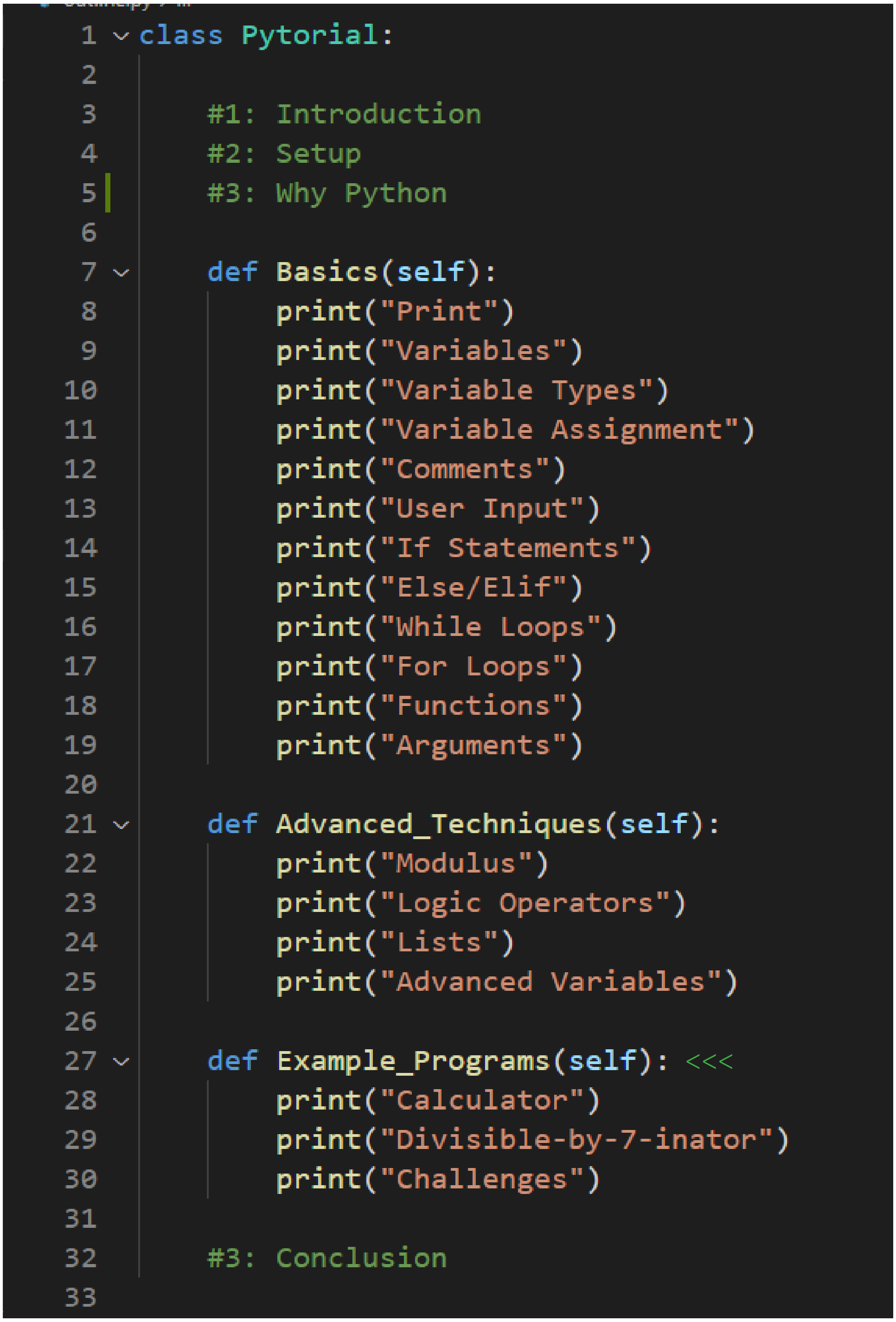
```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables") <<<

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```



advanced_variables.py > ...

```python

my_name = "Parker"

my_name = my_name + " says hi!"
#By adding this string to my_name,
#we change my name to the two combined

number = 5.66
#Creates a new decimal number (or float)

number = int(number)
#When we cast our float to an integer, it cuts off the decimal

print(my_name, number)

#OUTPUT-> Parker says hi! 5


```

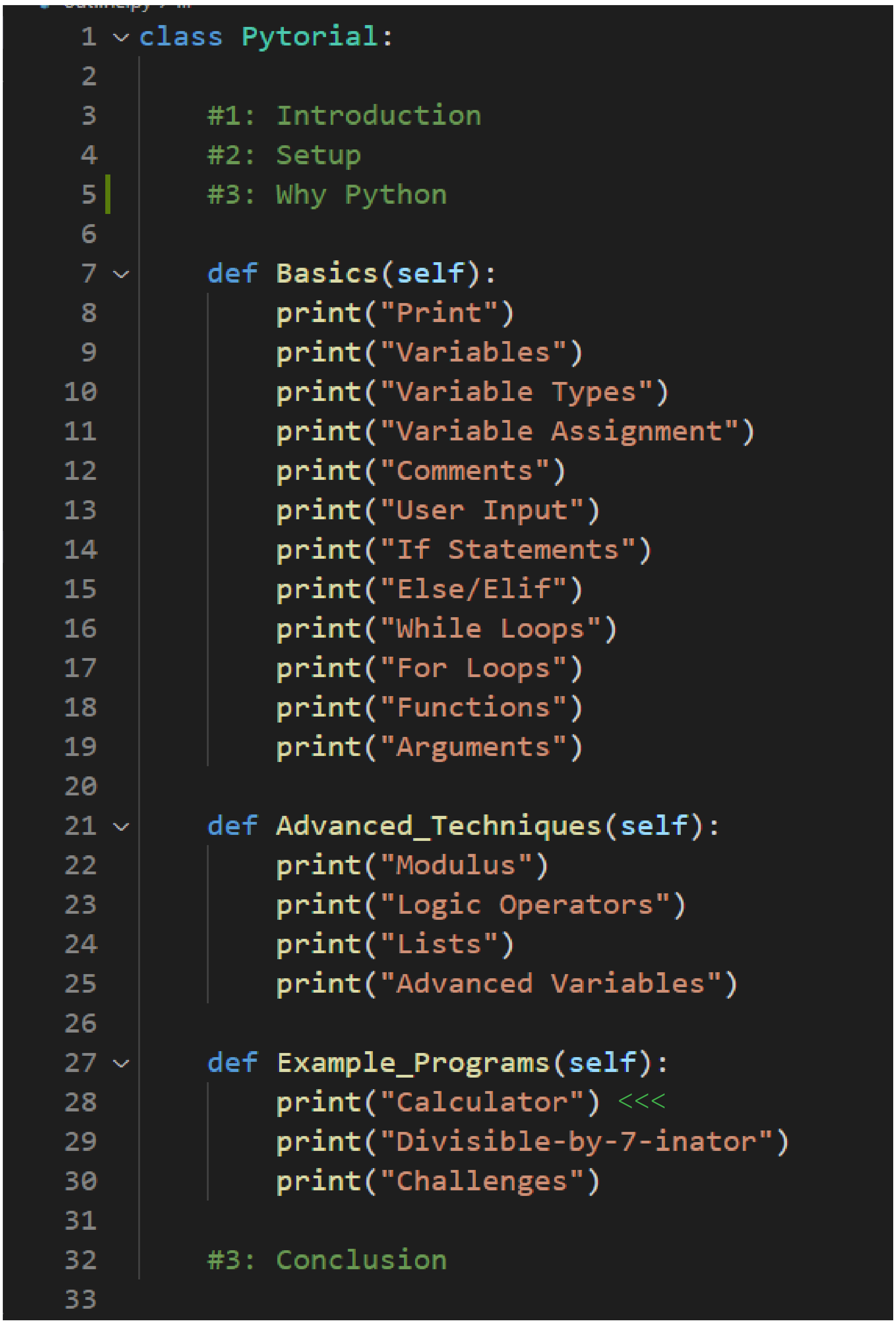
Our final subject is what I like to call “advanced variables”, or in other words, the complicated stuff with variables. First up is concatenation, or combining strings together. By adding two strings with a plus sign, they’ll be combined into one string. Second is casting variables, which is to switch a piece of data from one type to another. We can cast data types with the syntax in the example, and each cast has its own ins and outs. Finally is the concept of scope, which is the property of variables and other items to only exist within certain parts of the program. Basically, variables only exist within whatever they’re created in, so if you create a variable within a function, it will only exist and be accessible within that function. A global variable is a file-wide variable that exists everywhere, which can be brought into methods by typing global and then the variable’s name. While they are useful, global variables can become messy, and it’s a good idea to keep variables within only the necessary scope when possible.



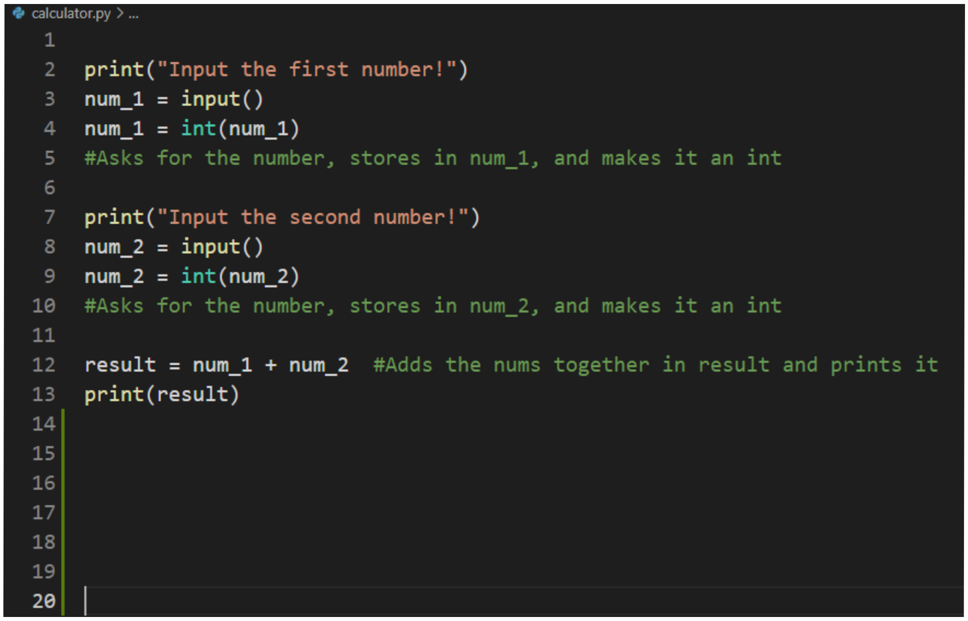
```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self): <<<
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion

```

```
EEEEEEEEEEEEEEEEEEEEEEXXXXXXX       XXXXXXX               AAA               MMMMMMMM               MMMMMMMMPPPPPPPPPPPPPPPPP   LLLLLLLLLLL             EEEEEEEEEEEEEEEEEEEEEE
E::::::::::::::::::::EX:::::X       X:::::X              A:::A              M:::::::M             M:::::::MP::::::::::::::::P  L:::::::::L             E::::::::::::::::::::E
E::::::::::::::::::::EX:::::X       X:::::X             A:::::A             M::::::::M           M::::::::MP::::::PPPPPP:::::P L:::::::::L             E::::::::::::::::::::E
EE::::::EEEEEEEEE::::EX::::::X     X::::::X            A:::::::A            M:::::::::M         M:::::::::MPP:::::P     P:::::PLL:::::::LL             EE::::::EEEEEEEEE::::E
  E:::::E       EEEEEEXXX:::::X   X:::::XXX           A:::::::::A           M::::::::::M       M::::::::::M  P::::P     P:::::P  L:::::L                 E:::::E       EEEEEE
  E:::::E                X:::::X X:::::X             A:::::A:::::A          M:::::::::::M     M:::::::::::M  P::::P     P:::::P  L:::::L                 E:::::E
  E::::::EEEEEEEEEE       X:::::X:::::X             A:::::A A:::::A         M:::::::M::::M   M::::M:::::::M  P::::PPPPPP:::::P   L:::::L                 E::::::EEEEEEEEEE
  E:::::::::::::::E        X:::::::::X             A:::::A   A:::::A        M::::::M M::::M M::::M M::::::M  P:::::::::::::PP    L:::::L                 E:::::::::::::::E
  E:::::::::::::::E        X:::::::::X            A:::::A     A:::::A       M::::::M  M::::M::::M  M::::::M  P::::PPPPPPPPP      L:::::L                 E:::::::::::::::E
  E::::::EEEEEEEEEE       X:::::X:::::X          A:::::AAAAAAAAA:::::A      M::::::M   M:::::::M   M::::::M  P::::P              L:::::L                 E::::::EEEEEEEEEE
  E:::::E                X:::::X X:::::X        A:::::::::::::::::::::A     M::::::M    M:::::M    M::::::M  P::::P              L:::::L                 E:::::E
  E:::::E       EEEEEEXXX:::::X   X:::::XXX    A:::::AAAAAAAAAAAAA:::::A    M::::::M     MMMMM     M::::::M  P::::P              L:::::L         LLLLLL  E:::::E       EEEEEE
EE::::::EEEEEEEE:::::EX::::::X     X::::::X   A:::::A             A:::::A   M::::::M               M::::::MPP::::::PP          LL:::::::LLLLLLLLL:::::LEE::::::EEEEEEEE:::::E
E::::::::::::::::::::EX:::::X       X:::::X  A:::::A               A:::::A  M::::::M               M::::::MP::::::::P          L::::::::::::::::::::::LE::::::::::::::::::::E
E::::::::::::::::::::EX:::::X       X:::::X A:::::A                 A:::::A M::::::M               M::::::MP::::::::P          L::::::::::::::::::::::LE::::::::::::::::::::E
EEEEEEEEEEEEEEEEEEEEEEXXXXXXX       XXXXXXXAAAAAAA                   AAAAAAAMMMMMMMM               MMMMMMMMPPPPPPPPPP          LLLLLLLLLLLLLLLLLLLLLLLLEEEEEEEEEEEEEEEEEEEEEE
```

While these abstract concepts are fairly easy to understand on their own, it can be hard to understand how they all fit together in a real program. Which is why I've included a few example programs with detailed comments on how everything works. I also encourage you to take these and play around with them, maybe even try modifying them to do something else.

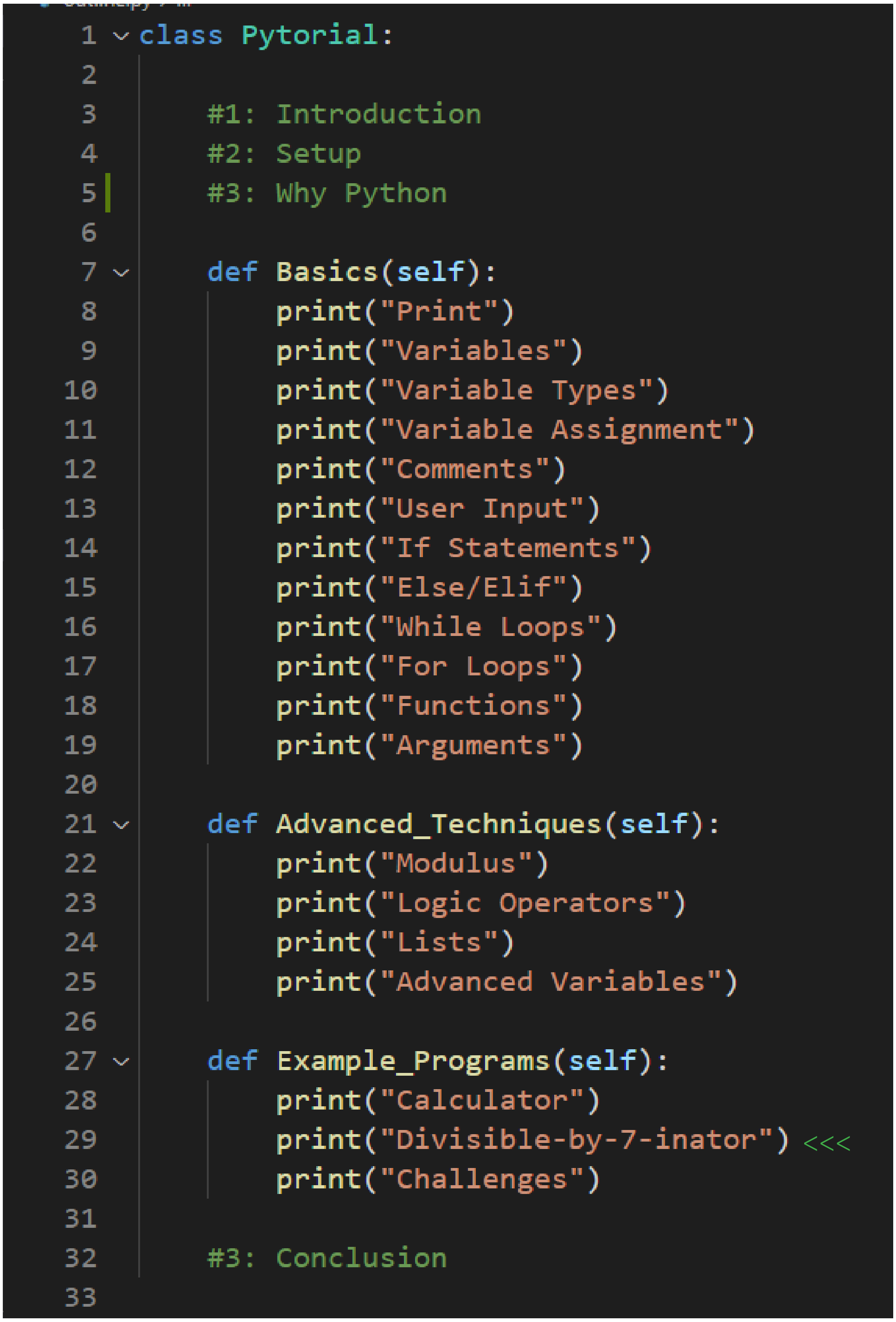


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator") <<<
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion
```

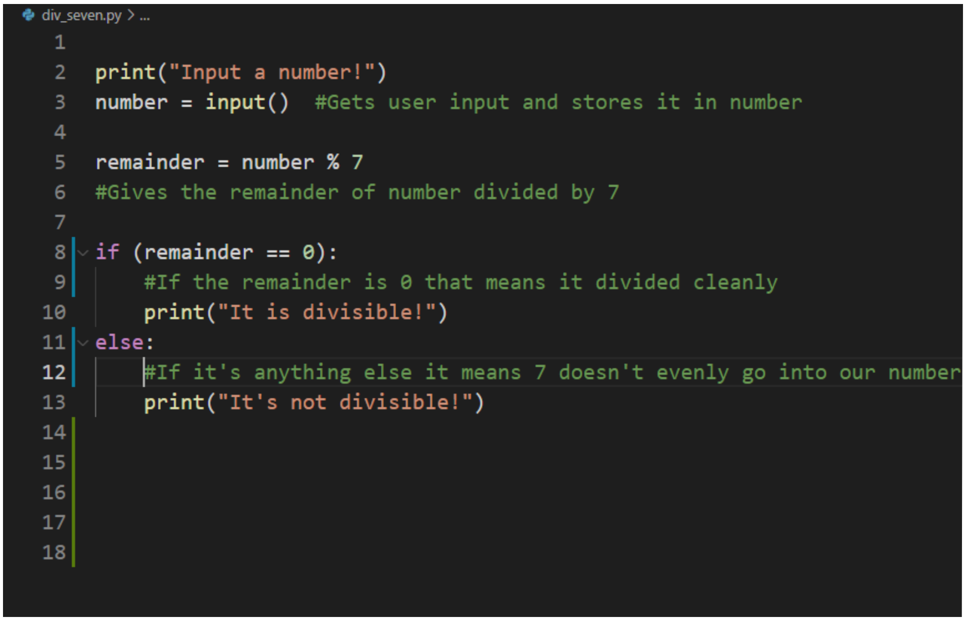


```python

print("Input the first number!")
num_1 = input()
num_1 = int(num_1)
#Asks for the number, stores in num_1, and makes it an int

print("Input the second number!")
num_2 = input()
num_2 = int(num_2)
#Asks for the number, stores in num_2, and makes it an int

result = num_1 + num_2  #Adds the nums together in result and prints it
print(result)
```

The first example program is a basic calculator, it takes in two numbers and outputs the sum of the two. First it asks for the first number from the user, then it stores that into num_1, then it does that same for num_2. Then it takes them and casts them into Finally it takes both numbers and adds them together in result, then prints out result.

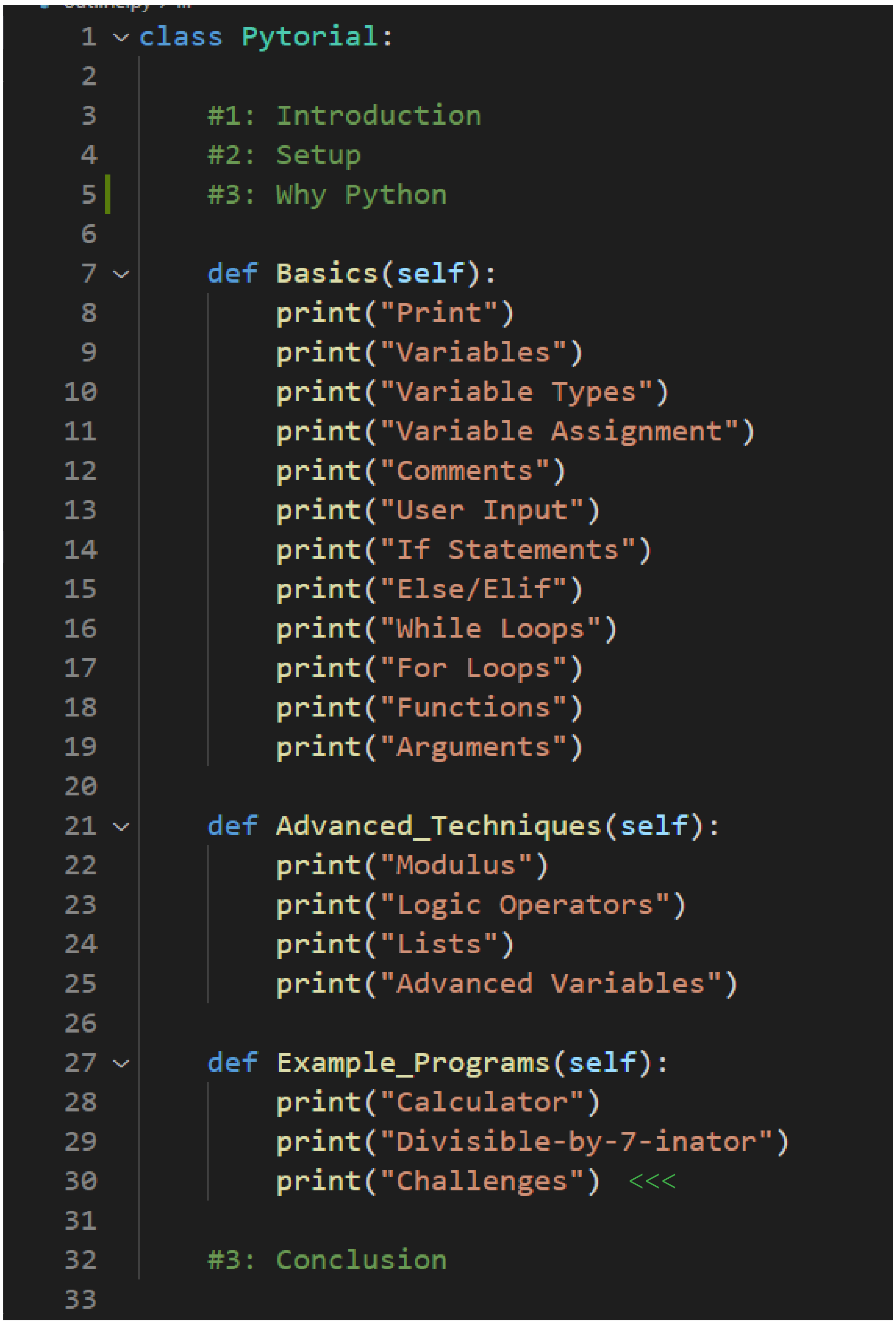


```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator") <<<
        print("Challenges")

    #3: Conclusion

```

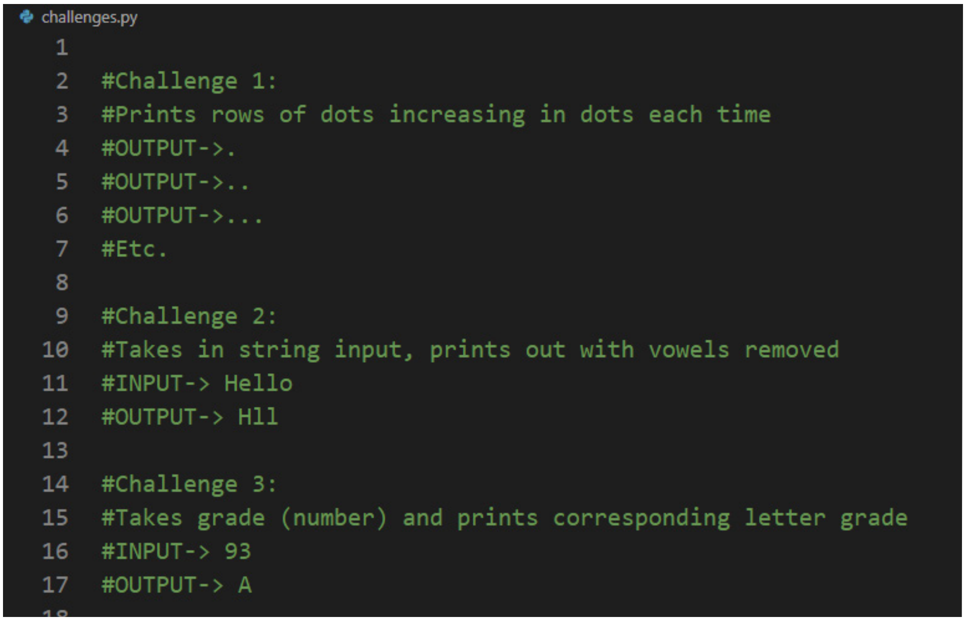


```python

print("Input a number!")
number = input()  #Gets user input and stores it in number

remainder = number % 7
#Gives the remainder of number divided by 7

if (remainder == 0):
    #If the remainder is 0 that means it divided cleanly
    print("It is divisible!")
else:
    #If it's anything else it means 7 doesn't evenly go into our number
    print("It's not divisible!")
```

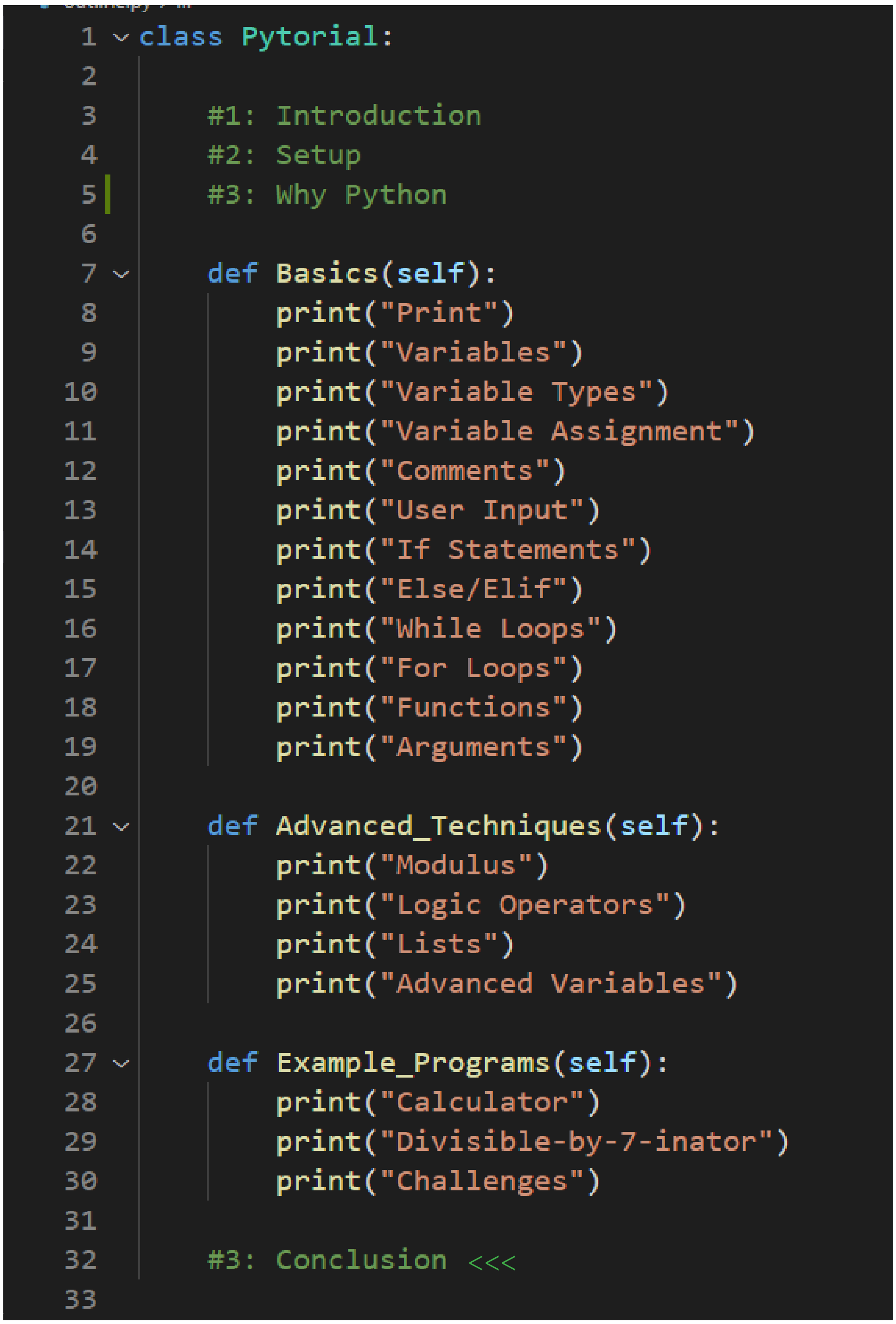
Our second example program is what I like to call the "divisible-by-7-inator", it's a program that takes a number and tells if it's divisible by 7. It first stores the user's input in number, then it set remainder to number mod 7. If the remainder is 0, that means it was cleanly divided by 7 and therefore is divisible by 7, if not then it is not divisible by 7. The if statement checks that and prints an according answer.



```
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")  <<<

    #3: Conclusion

```


```
challenges.py

#Challenge 1:
#Prints rows of dots increasing in dots each time
#OUTPUT->.
#OUTPUT->..
#OUTPUT->...
#Etc.

#Challenge 2:
#Takes in string input, prints out with vowels removed
#INPUT-> Hello
#OUTPUT-> Hll

#Challenge 3:
#Takes grade (number) and prints corresponding letter grade
#INPUT-> 93
#OUTPUT-> A
```

Now that you have a good grasp of how to program, here are some challenge programs to test your skills. The first challenge is to write a program that prints out rows of dots with one more dot after each successive row, the number of rows is up to you. The second challenge program is one that takes an input string, and then prints that string back out with every vowel removed. The final challenge is a program that takes a grade (as a number between 0 and 100) and prints out the equivalent letter grade. The solutions to these challenges can be found on the Pytorial GitHub page, with the rest of the files, but I encourage you to try them yourself before looking at the answers.



```python
class Pytorial:

    #1: Introduction
    #2: Setup
    #3: Why Python

    def Basics(self):
        print("Print")
        print("Variables")
        print("Variable Types")
        print("Variable Assignment")
        print("Comments")
        print("User Input")
        print("If Statements")
        print("Else/Elif")
        print("While Loops")
        print("For Loops")
        print("Functions")
        print("Arguments")

    def Advanced_Techniques(self):
        print("Modulus")
        print("Logic Operators")
        print("Lists")
        print("Advanced Variables")

    def Example_Programs(self):
        print("Calculator")
        print("Divisible-by-7-inator")
        print("Challenges")

    #3: Conclusion <<<

```

```
EEEEEEEEEEEEEEEEEEEEEENNNNNNNN        NNNNNNNNDDDDDDDDDDDDD
E::::::::::::::::::::EN:::::::N       N::::::ND::::::::::::DDD
E::::::::::::::::::::EN::::::::N      N::::::ND:::::::::::::::DD
EE::::::EEEEEEEEE::::EN:::::::::N     N::::::NDDD:::::DDDDD:::::D
  E:::::E       EEEEEEN::::::::::N    N::::::N  D:::::D    D:::::D
  E:::::E             N:::::::::::N   N::::::N  D:::::D     D:::::D
  E::::::EEEEEEEEEE   N:::::::N::::N  N::::::N  D:::::D     D:::::D
  E:::::::::::::::E   N::::::N N::::N N::::::N  D:::::D     D:::::D
  E:::::::::::::::E   N::::::N  N::::N:::::::N  D:::::D     D:::::D
  E::::::EEEEEEEEEE   N::::::N   N:::::::::::N  D:::::D     D:::::D
  E:::::E             N::::::N    N::::::::::N  D:::::D     D:::::D
  E:::::E       EEEEEEN::::::N     N:::::::::N  D:::::D    D:::::D
EE::::::EEEEEEEE:::::EN::::::N      N::::::::NDDD:::::DDDDD:::::D
E::::::::::::::::::::EN::::::N       N:::::::ND:::::::::::::::DD
E::::::::::::::::::::EN::::::N        N::::::ND::::::::::::DDD
EEEEEEEEEEEEEEEEEEEEEENNNNNNNN         NNNNNNNDDDDDDDDDDDDD
```

Well, the end is here, I've taught you everything you need to know. Hopefully now you have the skills you need to go out and write whatever programs you want (with a bit of research). If you're looking for more to learn about Python, there are some subjects that are far too complicated to explain here. Try looking up: Object-oriented programming (OOP), libraries (and by extension, pip), GUIs (and by extension, Tkinter), and even new programming languages entirely. Good luck in your coding adventures to come.

Reflection

I’m pretty happy with the way my book turned out, I enjoy that I got to give it that part of interactivity with the online files and the challenges (by the way, that is a real site that really does have all the files used in this and the challenge solutions). I also like the Python themed navigation tree (which actually is a valid Python program itself), I think it fits with the aesthetic and organization of this book. I’d say it really suffered from one huge problem, image resolution. I tried everything I could to get those code samples to look less grainy, but what I ended up with was them being readable enough. With more time I would have loved to try to fix the images so they’re sharper, but for now I’m kind of stuck with it like this. I also would have liked to include more subjects if I had more space (I have a laundry list of topics I left out for space). The hardest step for me was either the layout itself or making the code samples. I had a hard time making the layout look nice, especially the left page of each spread. I realized early on that code is mostly horizontal, and so it would not work well on its own page, so I figured it would fit better on the same page as the text. But that left me with the left page to fill with something interesting, so I put the navigation tree, which filled the space nicely and doubled as a remedy to my forgetting of headers.

```
BBBBBBBBBBBBBBBBB   IIIIIIIIII     OOOOOOOOO
B::::::::::::::::B  I::::::::I   OO:::::::::OO
B::::::BBBBBB:::::B I::::::::I OO:::::::::::::OO
BB:::::B     B:::::BII::::::IIO:::::::OOO:::::::O
  B::::B     B:::::B  I::::I  O::::::O   O::::::O
  B::::B     B:::::B  I::::I  O:::::O     O:::::O
  B::::BBBBBB:::::B   I::::I  O:::::O     O:::::O
  B:::::::::::::BB    I::::I  O:::::O     O:::::O
  B::::BBBBBB:::::B   I::::I  O:::::O     O:::::O
  B::::B     B:::::B  I::::I  O:::::O     O:::::O
  B::::B     B:::::B  I::::I  O:::::O     O:::::O
  B::::B     B:::::B  I::::I  O::::::O   O::::::O
BB:::::BBBBBB::::::BII::::::IIO:::::::OOO:::::::O
B:::::::::::::::::B I::::::::I OO:::::::::::::OO
B::::::::::::::::B  I::::::::I   OO:::::::::OO
BBBBBBBBBBBBBBBBB   IIIIIIIIII     OOOOOOOOO
```

Hello my name is Parker, I'm a senior at Ballard High, and I'm going to Western next year. I'm pretty interested in coding (hence why I made this book about it) and have been doing it for about 4 years or so. Outside of that I'm a really boring person, so this bio is going to end up pretty measly. I spend my time with friends mostly since I don't have any sports or clubs. I'm pretty hit or miss on school work but I usually pull through in the end, and usually I'm an attentive student (when I'm not at home on a computer).