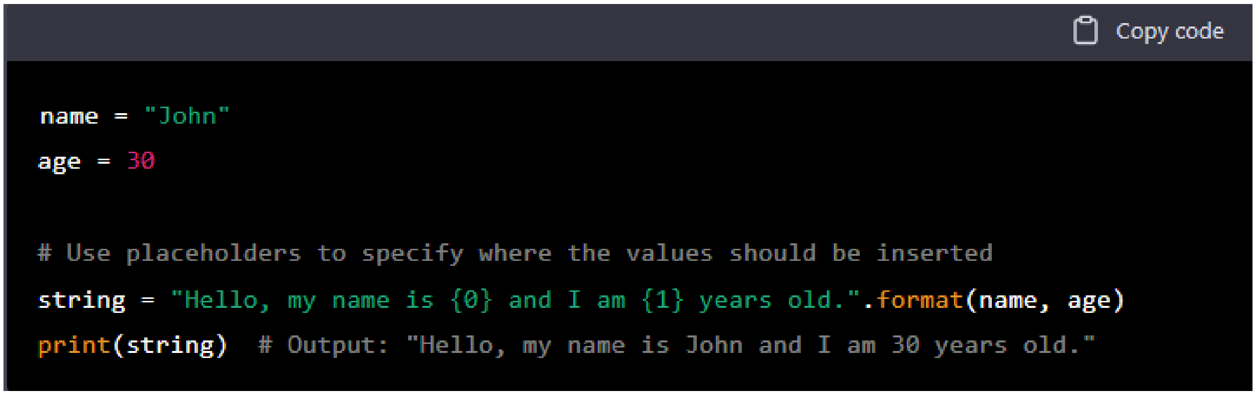
Introduction to Programming with Python Notes

- **Documentation**
  - docs.python.org
- **Functions, Variables**
  - When using print, you can assign multiple variables. Variables are separated by commas and each comma complies as a space. This means you do not need to manually add spacing in the case of concatenation.
  - There are multiple types of parameters.
    - Named parameters - A named parameter is a parameter that is passed to a function by name. It is a parameter that is specified in the function definition.
  - Regex is a feature in this language that allows for various means of manipulating data
    - \n - new line
    - \” - literal “
    - \t - tab
  - Functional strings are strings that accept variables without the need for concatenation
    - print(f”hello, {name}”)
      - would print the variable name without the addition operator
    - There are many built in methods to Strings in python
      - len(): Returns the length of a string.
      - lower(): Returns a copy of the string with all the uppercase characters converted to lowercase.
      - upper(): Returns a copy of the string with all the lowercase characters converted to uppercase.
      - strip(): Returns a copy of the string with leading and trailing whitespace removed.
      - split(): Splits a string into a list of substrings based on a specified delimiter.



```
name = "John"
age = 30

# Use placeholders to specify where the values should be inserted
string = "Hello, my name is {0} and I am {1} years old.".format(name, age)
print(string)  # Output: "Hello, my name is John and I am 30 years old."
```

  - In Python, it is necessary to define a function before you can use it. This is because, when you write a function call, Python needs to know what code to execute in order to carry out the function. If you try to call a function that has not been defined yet, Python will raise an error.

- In Python, the interpreter reads and executes the code you write in a top-down fashion. This means that, when the interpreter encounters a function definition, it stores the definition in memory, but it does not execute the function's code until the function is called.
- When you call a function, the interpreter looks up the function definition in memory and then executes the code that is defined in the function. If the function has not been defined yet, the interpreter will not be able to find the function definition in memory, and it will raise an error.