ENGR 103 Studio 6: Classes and Functions

Goals:

In this studio, we will begin with a bulky and cumbersome piece of code, and we will simplify it by packing away sections into **functions**. We will also keep organized by using **classes**. By the time we're done, our complicated task should be completed in just a few lines.

We will create a fractal tree – it begins with a trunk, which splits into two branches. Then each of those branches splits again, into two more branches. The process can go on indefinitely, can be done with more than just two branches per split, and any angle of splitting that we want. See the figure below, and the figures at the very end.

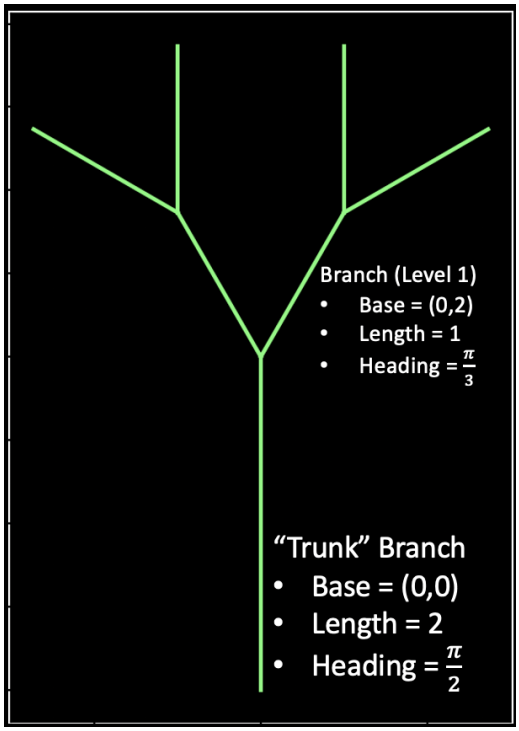

Figure 1. Fractal Tree with a trunk and two iterations of splitting, 60 degree split angle, and two branches per split. The "Heading" is the angle on a unit circle to which the branch points.

We will take advantage of three homemade classes –

- A **Tree** – this has attributes to control its growth, and we will write some supporting functions like Split_Branch that the tree can use to grow.
- A **Tree_Branch**, which has attributes describing its base, length and heading, as well as functions to grow the branch from its base, and draw it on a plot.
- A **Node**, which has properties X and Y. These nodes will keep track of the endpoints of each branch.

Part 0: Getting Familiar with the Code

Make it look nice:

Download the Studio 6 code from Canvas and open it in PyCharm. As usual, your first step should be to make the code look readable – press the arrows next to each function definition line to collapse each function into one line. When you are done, the entire code should fit on your screen without scrolling.

Later on, as you expand the functions to observe or edit them, remember to re-collapse them when you are finished. It is easy to get lost in long pieces of code, and the seconds of scrolling add up.

Make sure it's working:

Run the code once to see a demonstration and make sure that everything can run on your machine. You will have to download the packages `numpy` and `matplotlib`. Please use headphones if possible. You should see a Fractal Tree like that in Figure 1.

Read through and understand:

Go through each function and try to understand what it does. It is not necessary to understand each line – just read the function definitions and get a general idea of what is accomplished in each.

Try to get an idea of what process builds the example tree. First a trunk is defined. Next the trunk (parent) is split into branches (children). Next those branches are split into even more branches.

Tip: The left side of each assignment is always easier to understand than the right – for instance, if you see this line:

```
New_Heading =
      Parent_Branch.Heading + My_Tree.Branch_Angle
      *(Branch_Counter-(My_Tree.Branches_Per_Split-1)/2)
```

Just from reading the left side, you can say for certain that "A new heading is calculated." The actual calculation is not important for understanding the flow of the code – you can skip reading the right side.

Deliverable:

1. Describe what the class `Tree_Branch` represents. How many instances of `Tree_Branch` do you think were used to make Figure 1?
2. Describe what the class `Node` represents. How many instances of `Node` do you think were used to make Figure 1?
3. What function runs every time a `Node` is instantiated? What happens in that function?

Part 1: Write an Initialization Function

Find the lines of code that create the "trunk" branch:

(No editing yet – just locate them)

The trunk is created in the main function, beginning with

```
# ~~~~~ SECTION A ~~~~~~
```

Between the lines

```
#--- (BEGIN) CREATE BRANCH -----
#--- (END) CREATE BRANCH -------
```

Find the lines of code that create other Tree_Branches:

(No editing yet – just locate them)

Look through the rest of the code, and try to find the other **two** sections that lie between the same lines

```
#--- (BEGIN) CREATE BRANCH -----
#--- (END) CREATE BRANCH -------
```

Because it takes up so many lines to create the branch, define its attributes, grow the branch from its base, and then draw it, we will benefit from wrapping the process up into a single function that runs every time we create a branch.

Edit the Tree_Branch __init__ function:

Find the `Tree_Branch` __init__ function and follow the commented instructions to modify it.

By the time you're done, the function should accept inputs of `Base_Input`, `Length_Input`, and `Heading_Input`, and assign them to the attributes `self.Base`, `self.Length`, and `self.Heading`. It should also call the functions `Grow_Tip_From_Base()` and `Draw()`.

Note that, earlier, we had created a branch `My_Branch`, assigned attributes like `My_Branch.Length` and called functions like `My_Branch.Draw()`, etc.
Now, since we are working on the branch within its **own** function, we refer to the branch as `self`, assign attributes to `self.Length`, etc. and run functions `self.Draw`, etc.

Deliverable:

4. A screenshot of your `Tree_Branch` class's `__init__` function.

Beware! Your code **will** throw an error until you finish part 2.

Part 2: Replace Several Lines with Few Lines

Now that we've successfully changed the Tree_Branch's __init__ function, we can simplify the sections in which Tree_Branches are created.

Replace the first Tree_Branch creation:

Return to the main function's section `# ~~~~~ SECTION A ~~~~~~`

Replace the lines between

`#--- (BEGIN) CREATE BRANCH -----` and

`#--- (END) CREATE BRANCH -------`

With the line `My_Branch = Tree_Branch(BaseNode = …`

So that the entire `# ~~~~~ SECTION A ~~~~~~` looks like this:

```
# ~~~~~ SECTION A ~~~~~~
#-------------------------------------------------------------------------------------------
My_Trunk = Tree_Branch(Base_Input = Node(0,0), Length_Input = 2, Heading_Input = np.pi/2)
#-------------------------------------------------------------------------------------------
###########################################################################################
```

Replace the second Tree_Branch creation:

Find the main function's section `# ~~~~~ SECTION B ~~~~~~`

And replace the lines between

`#--- (BEGIN) CREATE BRANCH -----` and

`#--- (END) CREATE BRANCH -------`

With the line `My_Branch = Tree_Branch(BaseNode = …`

So that the entire `# ~~~~~ SECTION B ~~~~~~` looks like this:

```
# ~~~~~ SECTION B ~~~~~~
#-------------------------------------------------------------------------------------------
## SPLIT TRUNK INTO BRANCHES (LEVEL 1)
# The following lines split one parent branch into multiple children branches.

#--- (BEGIN) SPLIT BRANCH  -------
Parent_Branch = My_Trunk
Children      = []
for Branch_Counter in range(My_Tree.Branches_Per_Split):
    New_Heading       = Parent_Branch.Heading + My_Tree.Branch_Angle*(Branch_Counter-(N

    # --- (BEGIN) CREATE BRANCH -----
    My_Branch         = Tree_Branch(BaseInput = Parent_Branch.Tip, Length_Input = My_Tre
    # --- (END) CREATE BRANCH -------

    Children.append(My_Branch)
Branches_Level1 = Children
#--- (END) SPLIT BRANCH  -------

# In part 3, we will replace this --- SPLIT BRANCH --- section with the single line:
# Branches_Level1 = Split_Branch(Parent_Branch = My_Trunk)

#-------------------------------------------------------------------------------------------
###########################################################################################
```

Replace the third Tree_Branch creation:

Find the main function's section `# ~~~~~ SECTION C ~~~~~~`

And replace the lines between

`#--- (BEGIN) CREATE BRANCH -----` and

`#--- (END) CREATE BRANCH -------`

With the line `My_Branch = Tree_Branch(BaseNode = …`

So that the entire `# ~~~~~ SECTION C ~~~~~~` looks like this:

```
# ~~~~~ SECTION C ~~~~~~
#_____________________________________________________________________________
## SPLIT TRUNK INTO BRANCHES (LEVEL 2)
# The following lines split several parent branches into their own children branches

#--- (BEGIN) SPLIT GROUP OF BRANCHES ---------------
Parents = Branches_Level1
Group_Children  = []
for Parent_Branch in Parents:
    #--- (BEGIN) SPLIT BRANCH  --------
    Children      = []
    for Branch_Counter in range(My_Tree.Branches_Per_Split):
        New_Heading       = Parent_Branch.Heading + My_Tree.Branch_Angle*(Branch_Count

        # --- (BEGIN) CREATE BRANCH -----
        My_Branch        = Tree_Branch(Base_Input = Parent_Branch.Tip, Length_Input =
        # --- (END) CREATE BRANCH --------

        Children.append(My_Branch)
    # ----
    # In part 3, we will replace this --- SPLIT BRANCH --- section with the single lin
    # Children = Split_Branch(Parent_Branch = My_Trunk)
    # --- (END) SPLIT BRANCH  --------

    Group_Children.extend(Children)
Branches_Level2 = Group_Children
#--- (END) SPLIT GROUP OF BRANCHES ---------------

# In part 4, we will replace this entire section with the single line:
# Branches_Level2 = Split_Group(Parent_Branches = Branches_Level1)
#-----------------------------------------------------------------------------
###############################################################################
```

Deliverable:

5. A screenshot of the same example tree that shows up, after you've edited the three CREATE BRANCH sections.
   Beware! The code will throw an error unless you've edited all three.

Part 3: Write a Split_Branch function

We will follow a similar pattern, and use a function to replace all the code between lines.
`#--- (BEGIN) SPLIT BRANCH -----` and
`#--- (END) SPLIT BRANCH -------`

Edit the Split_Branch function:

Find the class Tree, and find its function Split_Branch.
Follow the commented instructions to modify it.

By the time you're done, the function should accept an input of `Parent_Branch,` and loop through each child branch, calculating a heading, creating the branch, and appending it to a list of children. The function should return the list `Children.`

Replace the first Branch Split:

Find the main function's section `# ~~~~~ SECTION B ~~~~~~`
And replace the lines between
`#--- (BEGIN) SPLIT BRANCH -----` and
`#--- (END) SPLIT BRANCH -------`
With the line `Branches_Level1 = Split_Branch(Parent_Branch =...`
So that the entire `# ~~~~~ SECTION B ~~~~~~` looks like this:

```
# ~~~~~ SECTION B ~~~~~~
#-------------------------------------------------------------------------------
Branches_Level1 = Split_Branch(Parent_Branch = My_Trunk)
#-------------------------------------------------------------------------------
###############################################################################
```

Replace the second Branch Split:

Find the main function's section `# ~~~~~ SECTION C ~~~~~~`
And replace the lines between
`#--- (BEGIN) SPLIT BRANCH -----` and
`#--- (END) SPLIT BRANCH -------`
With the line `Children = Split_Branch(Parent_Branch =...`
So that the entire `# ~~~~~ SECTION C ~~~~~~` looks like this:

```
# ~~~~~ SECTION C ~~~~~~
#_____________________________________________________________________________________
## SPLIT TRUNK INTO BRANCHES (LEVEL 2)
# The following lines split several parent branches into their own children branches

#--- (BEGIN) SPLIT GROUP OF BRANCHES ---------------
Parents = Branches_Level1
Group_Children  = []
for Parent_Branch in Parents:
    #--- (BEGIN) SPLIT BRANCH  -------
    Children = Split_Branch(Parent_Branch = My_Trunk)
    # --- (END) SPLIT BRANCH  -------

    Group_Children.extend(Children)
Branches_Level2 = Group_Children
#--- (END) SPLIT GROUP OF BRANCHES ---------------

# In part 4, we will replace this entire section with the single line:
# Branches_Level2 = Split_Group(Parent_Branches = Branches_Level1)
#-------------------------------------------------------------------------------------
#####################################################################################
```

Deliverable:

(None)

Part 4: Write a Split_Group function

We will follow a similar pattern, and use a function to replace all the code between lines.
`#--- (BEGIN) SPLIT GROUP OF BRANCHES -----` and
`#--- (END) SPLIT GROUP OF BRANCHES -------`

Edit the Split_Group function:

Find the class Tree, and find its function Split_Group.
Follow the commented instructions to modify it.
By the time you are done, it should accept inputs of `Parents` and return a list of `Group_Children`.

Replace the section that splits a group of branches

Find the main function's section `# ~~~~~ SECTION C ~~~~~~`
And replace the lines between
`#--- (BEGIN) SPLIT GROUP OF BRANCHES -----` and
`#--- (END) SPLIT GROUP OF BRANCHES -------`
With the line `Branches_Level2 = Split_Group(Parent_Branches…`
So that the entire `# ~~~~~ SECTION C ~~~~~~` looks like this:

```
# ~~~~~ SECTION C ~~~~~~
#_____________________________________________________________________________
Branches_Level2 = Split_Group(Parent_Branches = Branches_Level1)
#-----------------------------------------------------------------------------
#############################################################################
```

Sections A-C should now contain only 3 lines of actual code, like this:

```
# ~~~~~ SECTION A ~~~~~~
#-----------------------------------------------------------------------------
My_Trunk = Tree_Branch(Base_Input = Node(0,0), Length_Input = 2, Heading_Input = np.pi/2)
#-----------------------------------------------------------------------------
#############################################################################

# ~~~~~ SECTION B ~~~~~~
#-----------------------------------------------------------------------------
Branches_Level1 = Split_Branch(Parent_Branch = My_Trunk)
#-----------------------------------------------------------------------------
#############################################################################

# ~~~~~ SECTION C ~~~~~~
#_____________________________________________________________________________
Branches_Level2 = Split_Group(Parent_Branches = Branches_Level1)
#-----------------------------------------------------------------------------
#############################################################################
```

Deliverable:

6. A screenshot of your simplified sections A ,B, and C like above.

Part 5: Make pretty Artwork!

Use the loop:

If your code runs as pictured above, you can comment out or delete `# ~SECTION A~` through `# ~SECTION C~`, and uncomment `# ~~~~~ SECTION D ~~~~~~`

```
# ~~~~~ SECTION 4 ~~~~~~
#_____________________________________________________________________________
## GROW THE TREE USING FUNCTIONS
My_Trunk = Tree_Branch(Base_Input = Node(0,0), Length_Input = 2, Heading_Input = np.pi/2)
Branches = [My_Trunk]
for n in range(5):
    Branches = My_Tree.Split_Group(Branches)
```

We previously had close to 100 lines including comments and blank space, to accomplish only two iterations of splitting. Now, we've used only 4 lines of code accomplish five iterations (or however many we please).

This is why people like to use functions!

Deliverable:

7. A screenshot (or multiple) of the tree that you create. For inspiration, see the pictures below.