python wormhole

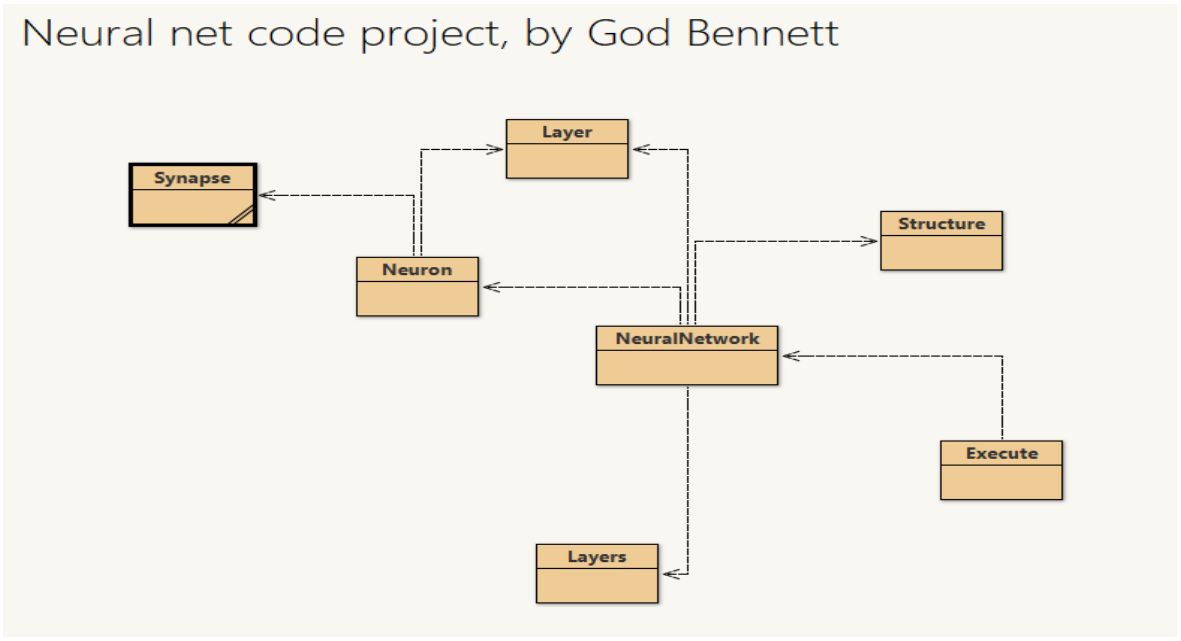
God Bennett

iCognium – God Bennett's "Python Wormhole"

30 minutes to 1 hour: Reasonably rapid movement from 0 python practice to absorption of sensible python programming

Introduction

As a pedagogical tool, in Python, via iCognium, we show an overview of how a neural network's components connect through the use of visual maps of how code units relate (where code units are described as partial and main realities on page 4 and beyond) and thereafter, we show an intuitive programming flow.



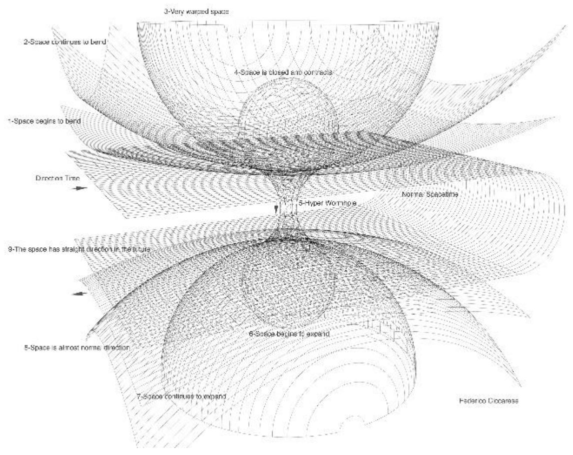
godquestbennett@gmail.com | November 2024+

Why look at an overview of fundamental neural networks?

Libraries typically hide away lots of work, be it Ai libraries *(Think of Ai/Machine learning libraries like compactified software that makes it reasonably easy to perform complicated tasks using premade structures;* ***like using a 3d printer and a digital blueprint to print a copy of a car's entire body, instead of painstakingly building each body section manually. We won't delve into building the fundamental model from scratch in python now, but we will review an overview of such.***

Note that this strategy is reasonably general, and can present as a viable way of helping to solve problems through a variety of languages.

Python Wormhole – Begin!


3-Very warped space
2-Space continues to bend
4-Space is closed and contracts
1-Space begins to bend
Direction Time
Normal Spacetime
5-Hyper Wormhole
9-The space has straight direction in the future
6-Space begins to expand
8-Space is almost normal direction
7-Space continues to expand
Federico Ciccarese

Imagine yourself as the creator of a universe. Programming normally consists of

1. Blueprints/Partial Realities (i.e. your blueprints/plans for stuff in your universe)
2. Main Reality (i.e., where you run instances of your blueprints/plans)

All programming essentially makes use of Objects/Blueprints Partial Realities as well as "Object/Main Reality" i.e., somewhere to see those blueprints doing things, i.e. **the scripts/character descriptions in a TV show** can be likened to these blueprints/plans/partial realities, while **the tv show itself being broadcast** can be likened to "Main reality" where the aforesaid scripts or plans show those characters in action or "instantiated".

Artificial Neural networks, are essentially loops that expose their structure to supervised pairs of data or examples related to a task/objective, while making use of Blueprints and Main Reality (i.e. somewhere to run instances of the objects that comprise the neural network)

Our sample project

Blueprints/Partial Realities: Planet, Tree, Human ← Main Reality

Our Sample Project: Python point of view

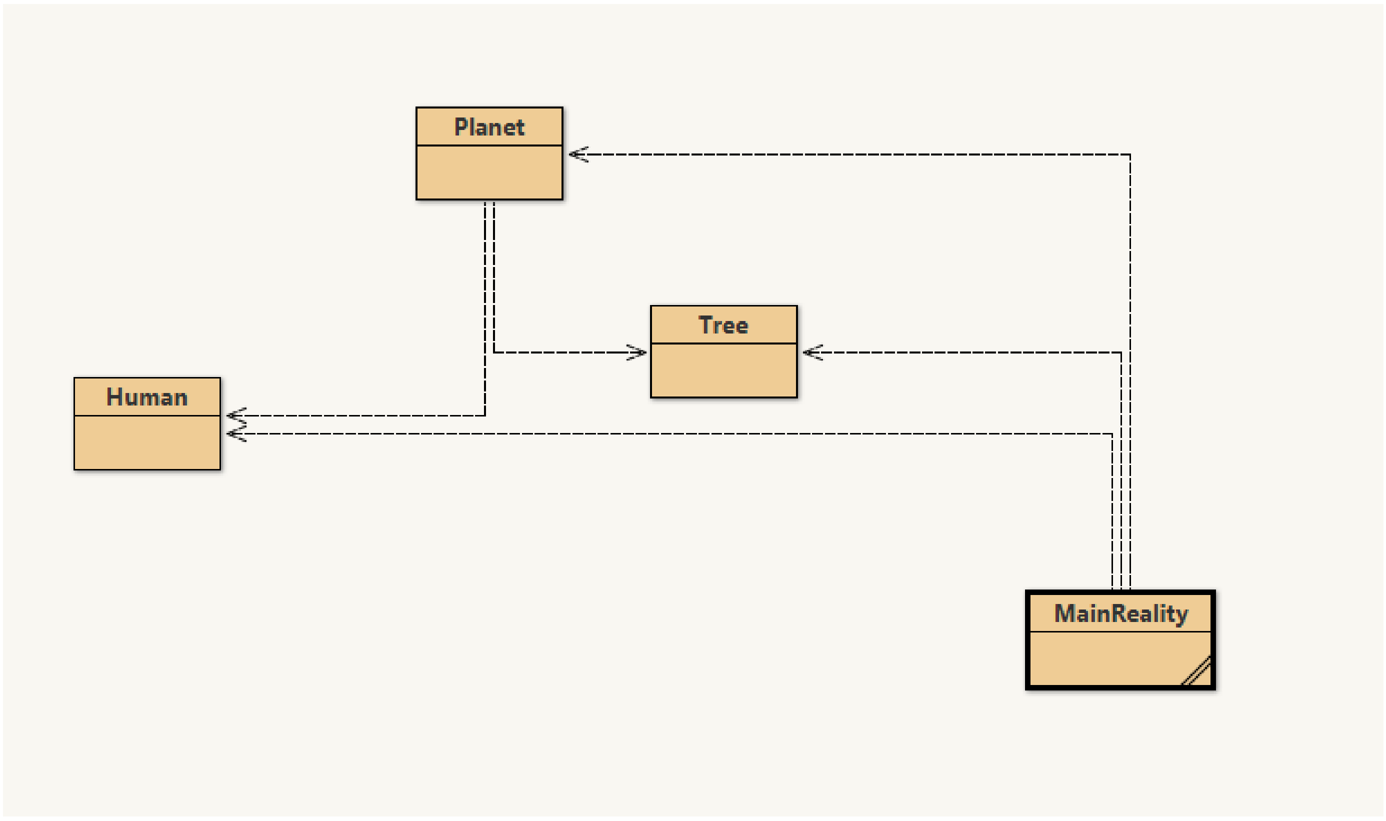
Blueprints/Partial Realities (Classes in Python): Planet, Tree, Human ← Main Reality (Main Class where blueprints are shown in action)

Typically, in programming, for a project, we normally have **partial realities/blueprints** and one **main reality** where all **blueprints** are shown in action through **a final "screen", the main reality.**

Any coding project we do typically consists of:

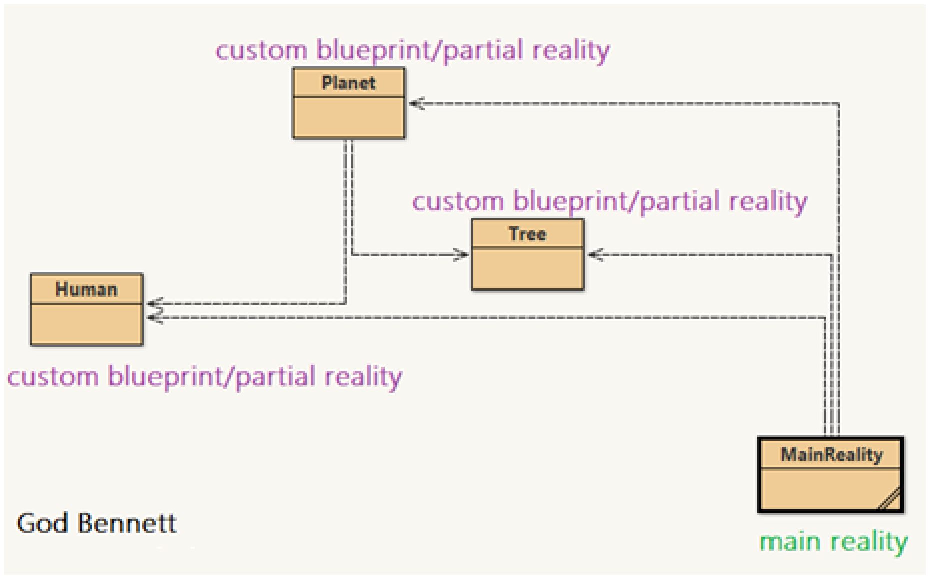
a. A combination our own **custom-classes/blueprints/"partial realities"** with in-built **classes/blueprints**, specified in the programming language. These can be likened to partial realities, because we call blueprints specified in the language where they are "called to action" in our blueprints.
b. A **main reality** where everything we build/refer to above are shown in action.

Our Sample Project: Python Code Map



Our Sample Project: Python Code Map (Annotated)

Blueprints/Partial Realities (Classes in Python): Planet, Tree, Human ← Main Reality (Main Class where blueprints are shown in action)

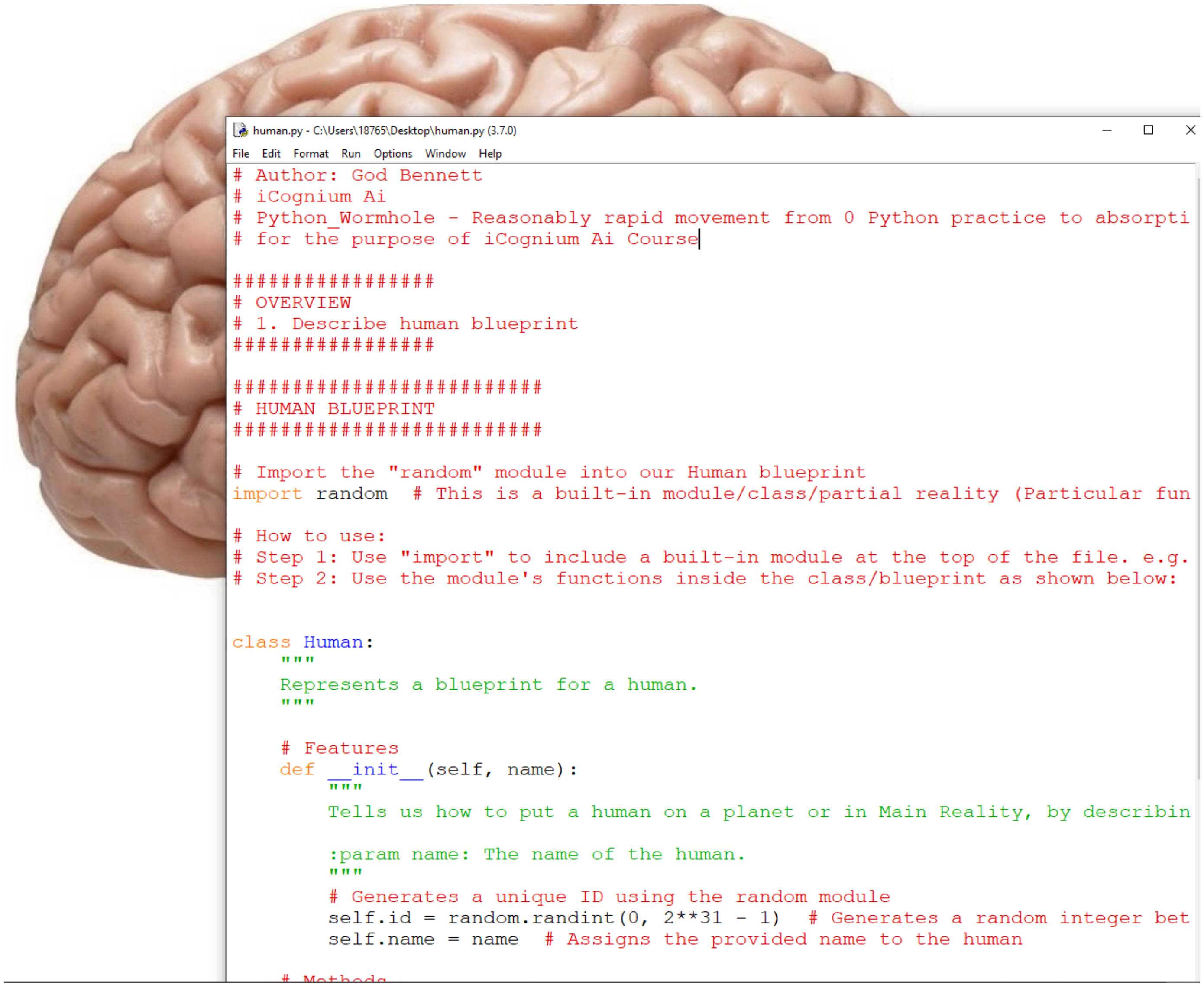

custom blueprint/partial reality
Planet
custom blueprint/partial reality
Tree
Human
custom blueprint/partial reality
MainReality
God Bennett
main reality

<u>Note visual map done in separate language, but code is python!</u>

Our Sample Project: Python Code Map (Blueprint/partial reality sample code)

Typically, each custom blueprint will have:

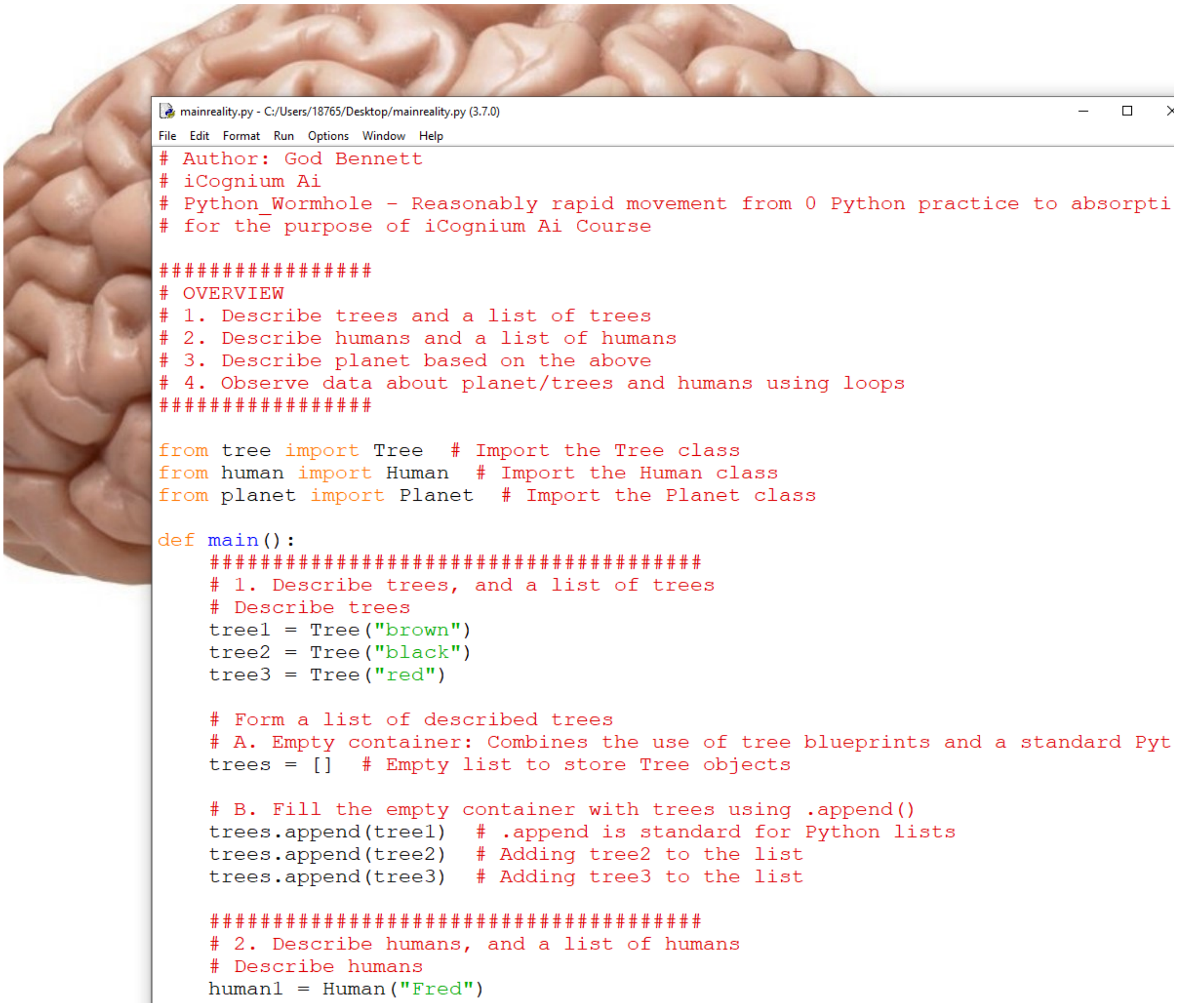
1. Features (characteristics/variables, i.e. human name, id
2. Constructor (For eg: Tells us how to put a human on a planet or in Main Reality, by describing a name)
3. Methods (For eg: Tells us what we can do with a human on a planet or in Main Reality, for eg, getting data – getName() about human is an example of what we can do with a human)


human.py - C:\Users\18765\Desktop\human.py (3.7.0)
File Edit Format Run Options Window Help

```python
# Author: God Bennett
# iCognium Ai
# Python_Wormhole - Reasonably rapid movement from 0 Python practice to absorpti
# for the purpose of iCognium Ai Course

#################
# OVERVIEW
# 1. Describe human blueprint
#################

##########################
# HUMAN BLUEPRINT
##########################

# Import the "random" module into our Human blueprint
import random  # This is a built-in module/class/partial reality (Particular fun

# How to use:
# Step 1: Use "import" to include a built-in module at the top of the file. e.g.
# Step 2: Use the module's functions inside the class/blueprint as shown below:


class Human:
    """
    Represents a blueprint for a human.
    """

    # Features
    def __init__(self, name):
        """
        Tells us how to put a human on a planet or in Main Reality, by describin

        :param name: The name of the human.
        """
        # Generates a unique ID using the random module
        self.id = random.randint(0, 2**31 - 1)   # Generates a random integer bet
        self.name = name  # Assigns the provided name to the human

    # Methods
```

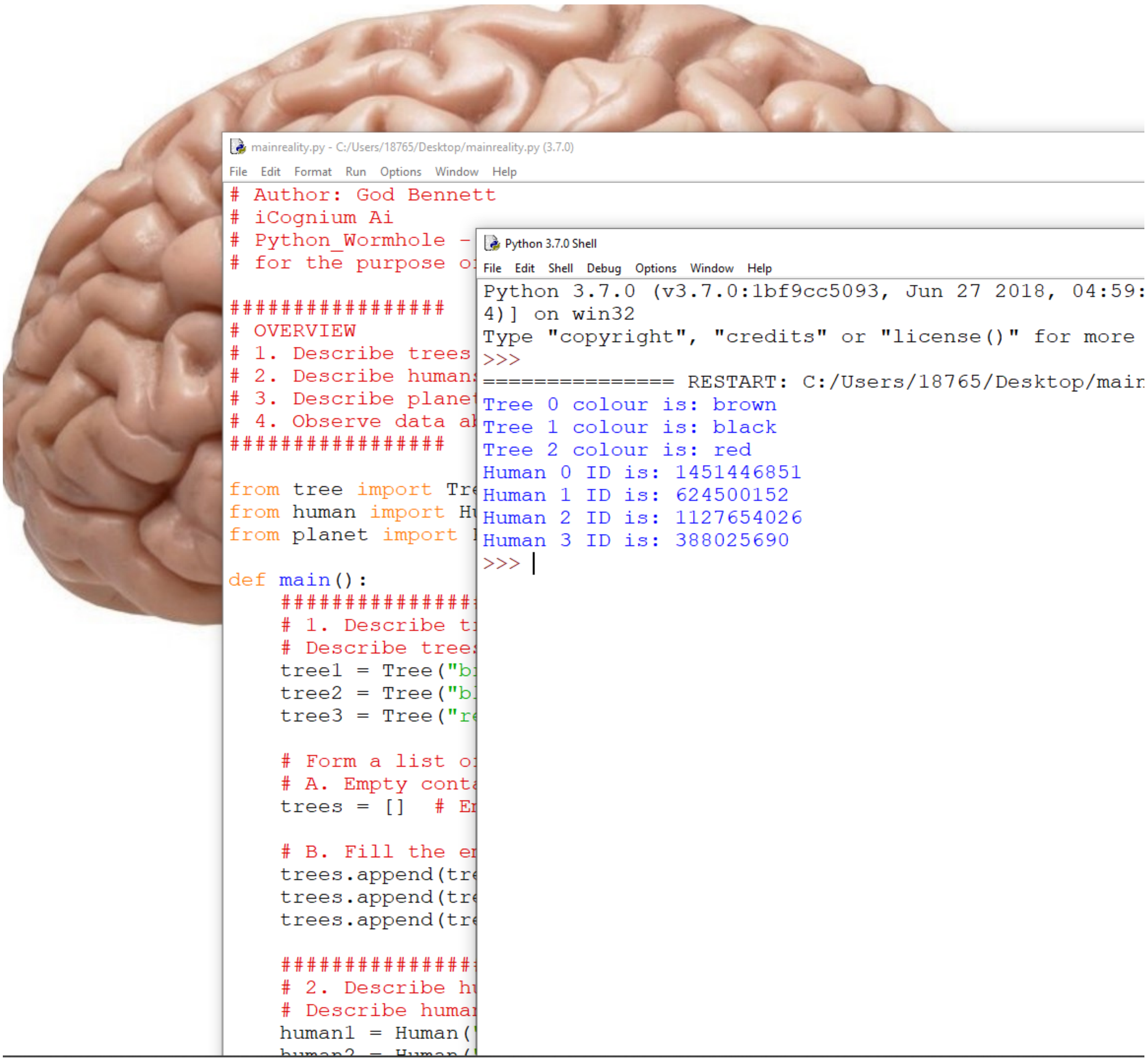
Our Sample Project: Python Code Map (Main Reality sample code)

Similar to partial realities, or main realities can have features (the partial realities), and methods, including a main function which forms our main screen, or other methods like "System.out.println ("message here") for revealing data about our partial realities.


mainreality.py - C:/Users/18765/Desktop/mainreality.py (3.7.0)
File Edit Format Run Options Window Help

```
# Author: God Bennett
# iCognium Ai
# Python_Wormhole - Reasonably rapid movement from 0 Python practice to absorpti
# for the purpose of iCognium Ai Course

#################
# OVERVIEW
# 1. Describe trees and a list of trees
# 2. Describe humans and a list of humans
# 3. Describe planet based on the above
# 4. Observe data about planet/trees and humans using loops
#################

from tree import Tree  # Import the Tree class
from human import Human  # Import the Human class
from planet import Planet  # Import the Planet class

def main():
    #######################################
    # 1. Describe trees, and a list of trees
    # Describe trees
    tree1 = Tree("brown")
    tree2 = Tree("black")
    tree3 = Tree("red")

    # Form a list of described trees
    # A. Empty container: Combines the use of tree blueprints and a standard Pyt
    trees = []  # Empty list to store Tree objects

    # B. Fill the empty container with trees using .append()
    trees.append(tree1)  # .append is standard for Python lists
    trees.append(tree2)  # Adding tree2 to the list
    trees.append(tree3)  # Adding tree3 to the list

    #######################################
    # 2. Describe humans, and a list of humans
    # Describe humans
    human1 = Human("Fred")
```

Result after executing our main reality:


mainreality.py - C:/Users/18765/Desktop/mainreality.py (3.7.0)
File Edit Format Run Options Window Help
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

```
# Author: God Bennett
# iCognium Ai
# Python_Wormhole -
# for the purpose o

#################
# OVERVIEW
# 1. Describe trees
# 2. Describe human
# 3. Describe plane
# 4. Observe data a
#################

from tree import Tr
from human import H
from planet import 

def main():
    ###############
    # 1. Describe t
    # Describe tree
    tree1 = Tree("b
    tree2 = Tree("b
    tree3 = Tree("r

    # Form a list o
    # A. Empty cont
    trees = []   # E

    # B. Fill the e
    trees.append(tr
    trees.append(tr
    trees.append(tr

    ###############
    # 2. Describe h
    # Describe huma
    human1 = Human(
    human2 = Human(
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:
4)] on win32
Type "copyright", "credits" or "license()" for more
>>>
=============== RESTART: C:/Users/18765/Desktop/main
Tree 0 colour is: brown
Tree 1 colour is: black
Tree 2 colour is: red
Human 0 ID is: 1451446851
Human 1 ID is: 624500152
Human 2 ID is: 1127654026
Human 3 ID is: 388025690
>>> 
```

Your Sample Project: Test

Write out a Python project, with any set of 4 objects, eg Partial Realities: Lion, Elephant, Zoo, and your MainReality to showcase these. Compose sensible features for your PartialRealities, and display data about them as seen in the example project.

Your Long Term Project

After the live fundamental neural network programming session component from the Universal Ai Diploma, after achieving the diploma, translate the neural network code into your language of choice, like Python (if not comfortable with Python), to help gauge how well you've absorbed the principles. This should be reasonably followed by cyclical practice roughly every 6 months in your selected language, without using anything but your memory. (i.e. no internet, no looking back at the original code)