DEPARTMENT OF ELECTRONICS AND COMMUNICATION

B. M. S. COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELAGAVI)

BANGALORE – 560019

2020-21

7TH SEMESTER SELF STUDY

IN

EMBEDDED SYSTEM DESIGN

(16EC7DCESD)

PYTHON PROGRAMS

BY

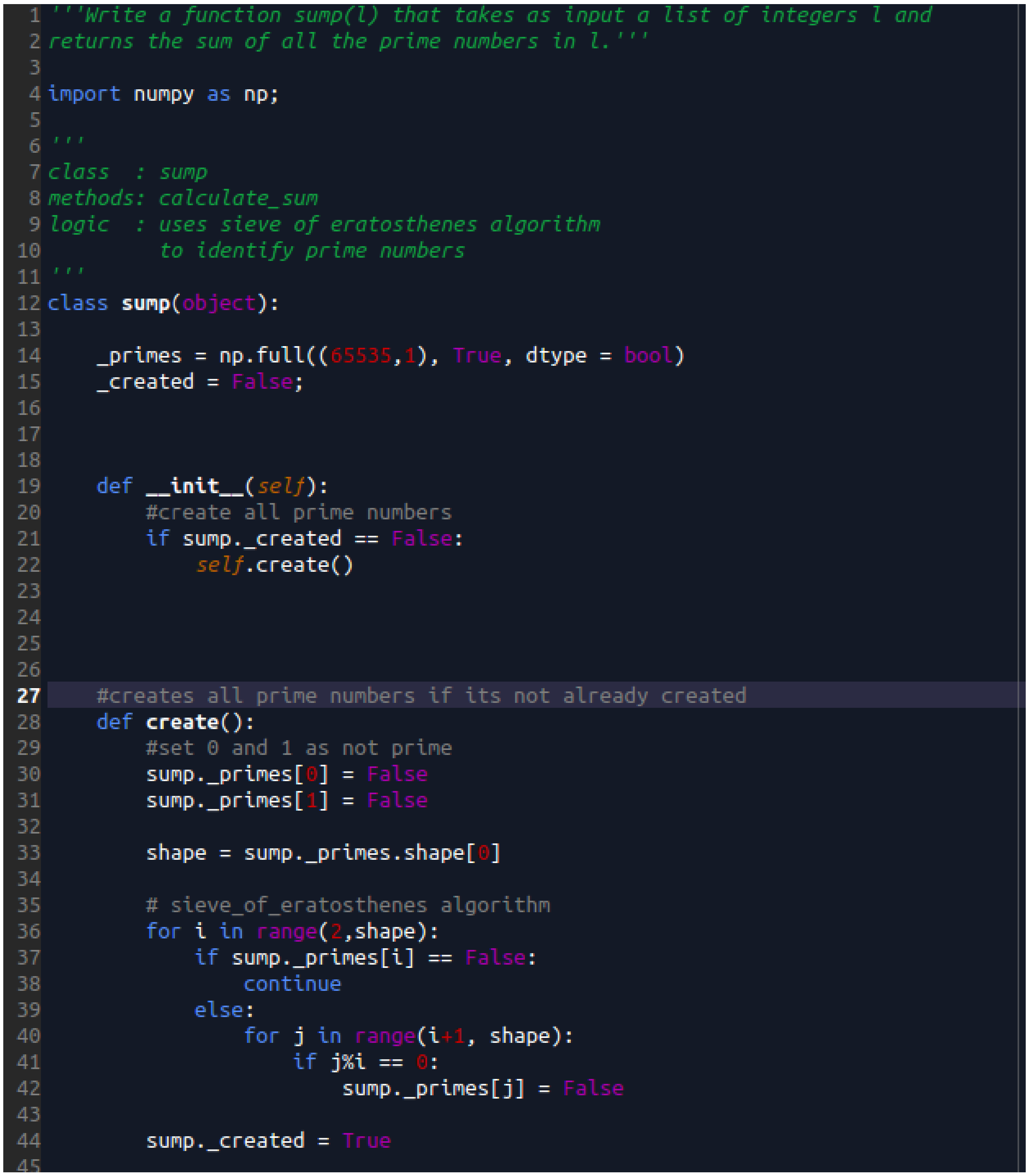
AKSHAY S RAO 1BM17EC007

Course Instructor

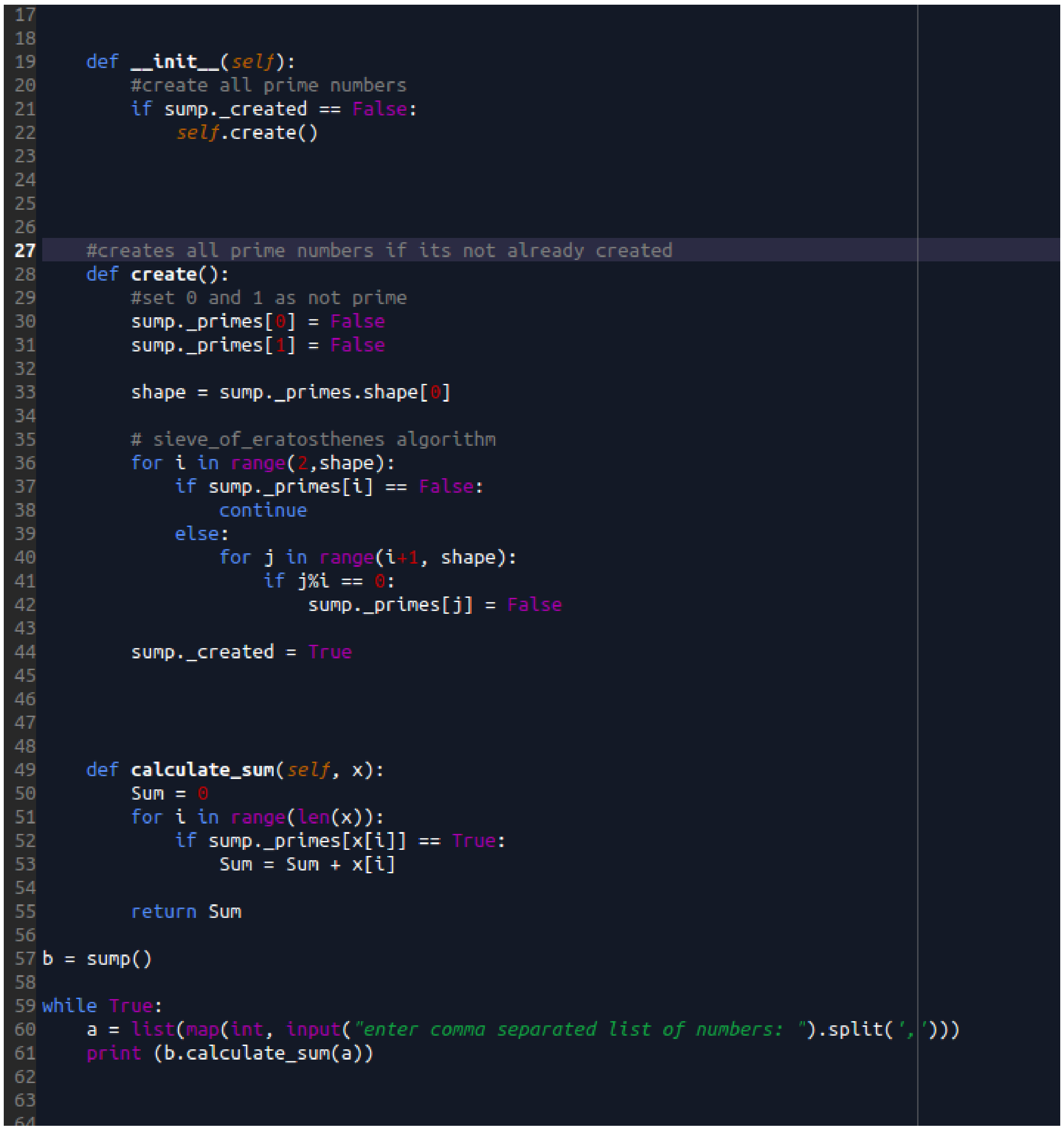
Dr. Kiran Bailey

Assistant Professor

1. Write a function sump(l) that takes as input a list of integers l and returns the sum of all the prime numbers in l.

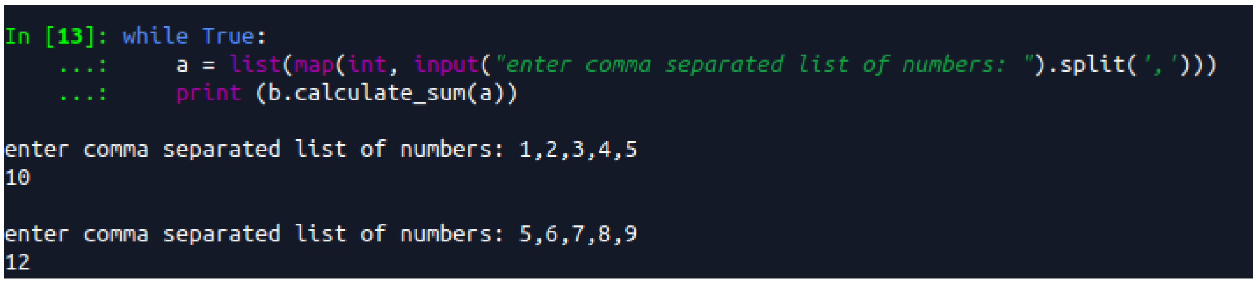


```python
'''Write a function sump(l) that takes as input a list of integers l and
returns the sum of all the prime numbers in l.'''

import numpy as np;

'''
class  : sump
methods: calculate_sum
logic  : uses sieve of eratosthenes algorithm
         to identify prime numbers
'''
class sump(object):

    _primes = np.full((65535,1), True, dtype = bool)
    _created = False;



    def __init__(self):
        #create all prime numbers
        if sump._created == False:
            self.create()




    #creates all prime numbers if its not already created
    def create():
        #set 0 and 1 as not prime
        sump._primes[0] = False
        sump._primes[1] = False

        shape = sump._primes.shape[0]

        # sieve_of_eratosthenes algorithm
        for i in range(2,shape):
            if sump._primes[i] == False:
                continue
            else:
                for j in range(i+1, shape):
                    if j%i == 0:
                        sump._primes[j] = False

        sump._created = True
```



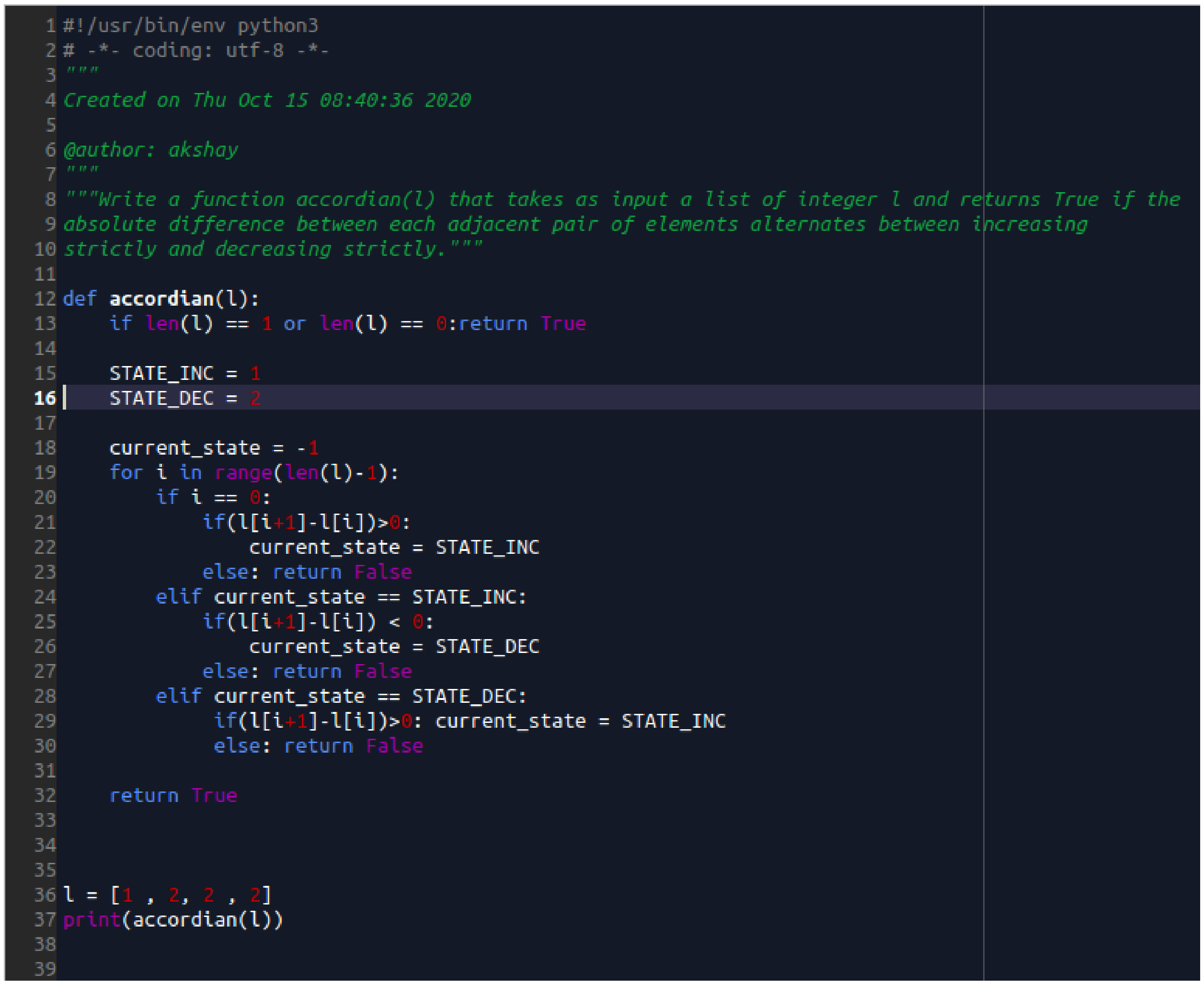
```python


    def __init__(self):
        #create all prime numbers
        if sump._created == False:
            self.create()




    #creates all prime numbers if its not already created
    def create():
        #set 0 and 1 as not prime
        sump._primes[0] = False
        sump._primes[1] = False

        shape = sump._primes.shape[0]

        # sieve_of_eratosthenes algorithm
        for i in range(2,shape):
            if sump._primes[i] == False:
                continue
            else:
                for j in range(i+1, shape):
                    if j%i == 0:
                        sump._primes[j] = False

        sump._created = True




    def calculate_sum(self, x):
        Sum = 0
        for i in range(len(x)):
            if sump._primes[x[i]] == True:
                Sum = Sum + x[i]

        return Sum

b = sump()

while True:
    a = list(map(int, input("enter comma separated list of numbers: ").split(',')))
    print (b.calculate_sum(a))


```

Results



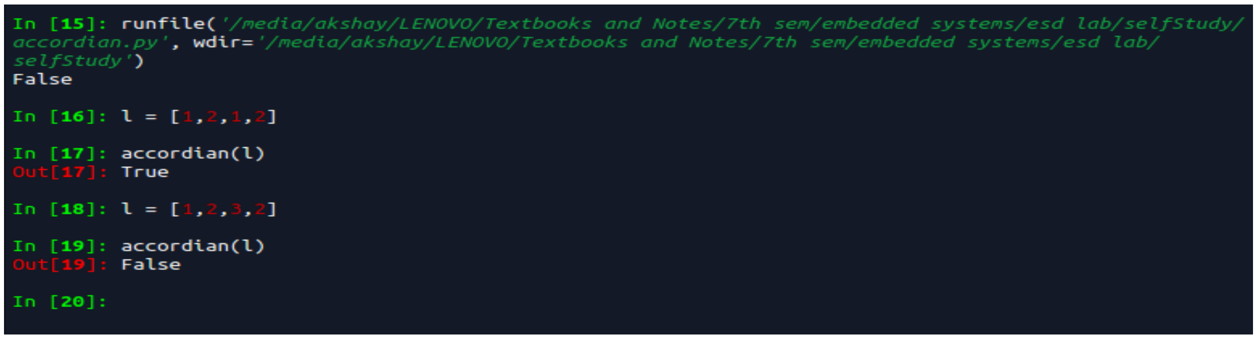
```
In [13]: while True:
    ...:     a = list(map(int, input("enter comma separated list of numbers: ").split(',')))
    ...:     print (b.calculate_sum(a))

enter comma separated list of numbers: 1,2,3,4,5
10

enter comma separated list of numbers: 5,6,7,8,9
12
```

2. Write a function accordian(l) that takes as input a list of integer l and returns True if the absolute difference between each adjacent pair of elements alternates between increasing strictly and decreasing strictly.



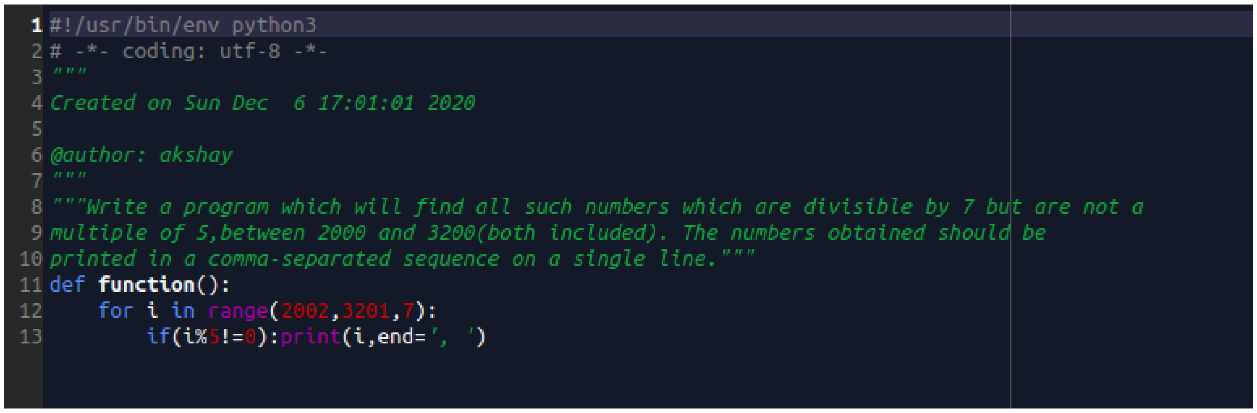
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 15 08:40:36 2020

@author: akshay
"""
"""Write a function accordian(l) that takes as input a list of integer l and returns True if the
absolute difference between each adjacent pair of elements alternates between increasing
strictly and decreasing strictly."""

def accordian(l):
    if len(l) == 1 or len(l) == 0:return True

    STATE_INC = 1
    STATE_DEC = 2

    current_state = -1
    for i in range(len(l)-1):
        if i == 0:
            if(l[i+1]-l[i])>0:
                current_state = STATE_INC
            else: return False
        elif current_state == STATE_INC:
            if(l[i+1]-l[i]) < 0:
                current_state = STATE_DEC
            else: return False
        elif current_state == STATE_DEC:
             if(l[i+1]-l[i])>0: current_state = STATE_INC
             else: return False

    return True



l = [1 , 2, 2 , 2]
print(accordian(l))


```

Results



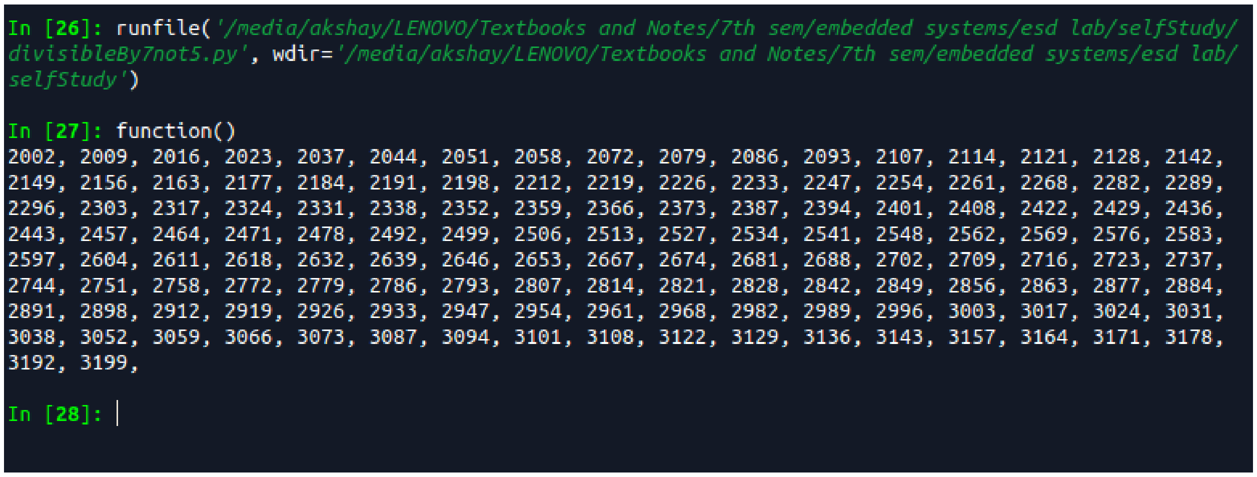
```
In [15]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/
accordian.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/
selfStudy')
False

In [16]: l = [1,2,1,2]

In [17]: accordian(l)
Out[17]: True

In [18]: l = [1,2,3,2]

In [19]: accordian(l)
Out[19]: False

In [20]:
```

3. Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200(both included). The numbers obtained should be printed in a comma-separated sequence on a single line.



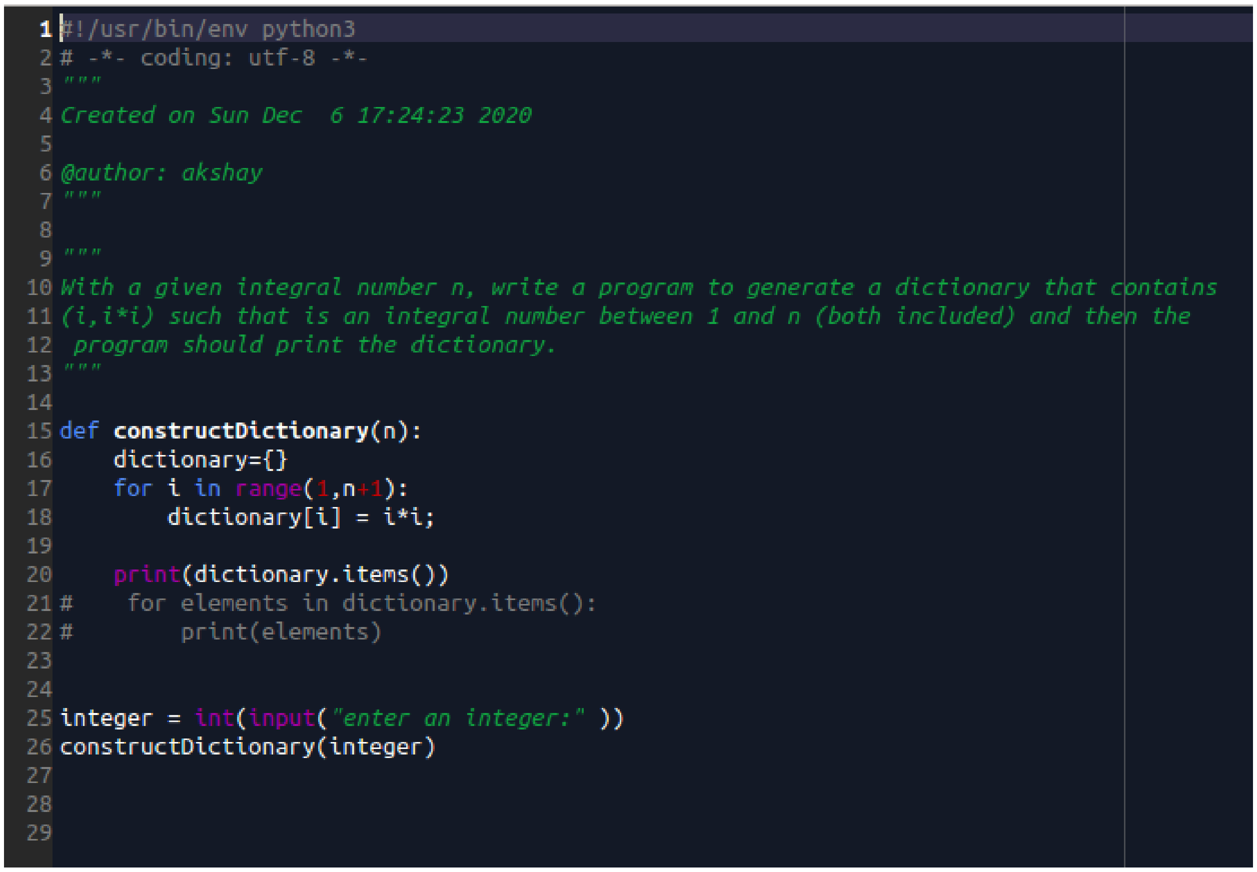
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  6 17:01:01 2020

@author: akshay
"""
"""Write a program which will find all such numbers which are divisible by 7 but are not a
multiple of 5,between 2000 and 3200(both included). The numbers obtained should be
printed in a comma-separated sequence on a single line."""
def function():
    for i in range(2002,3201,7):
        if(i%5!=0):print(i,end=', ')
```

Results:



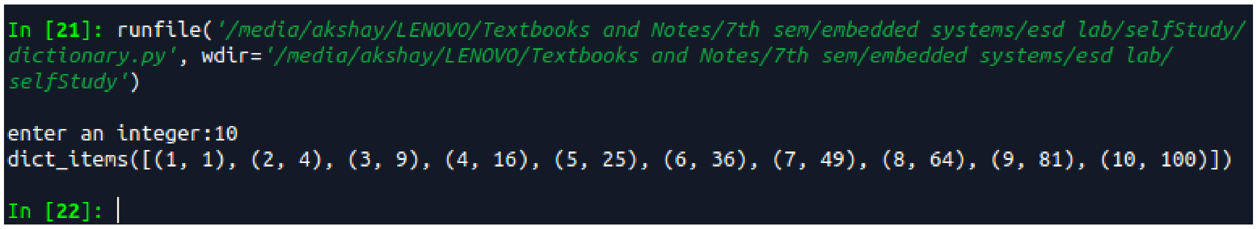
```
In [26]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/
divisibleBy7not5.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/
selfStudy')

In [27]: function()
2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114, 2121, 2128, 2142,
2149, 2156, 2163, 2177, 2184, 2191, 2198, 2212, 2219, 2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289,
2296, 2303, 2317, 2324, 2331, 2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436,
2443, 2457, 2464, 2471, 2478, 2492, 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562, 2569, 2576, 2583,
2597, 2604, 2611, 2618, 2632, 2639, 2646, 2653, 2667, 2674, 2681, 2688, 2702, 2709, 2716, 2723, 2737,
2744, 2751, 2758, 2772, 2779, 2786, 2793, 2807, 2814, 2821, 2828, 2842, 2849, 2856, 2863, 2877, 2884,
2891, 2898, 2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 2989, 2996, 3003, 3017, 3024, 3031,
3038, 3052, 3059, 3066, 3073, 3087, 3094, 3101, 3108, 3122, 3129, 3136, 3143, 3157, 3164, 3171, 3178,
3192, 3199,

In [28]:
```

4. With a given integral number n, write a program to generate a dictionary that contains (i,i*i) such that is an integral number between 1 and n (both included) and then the program should print the dictionary.



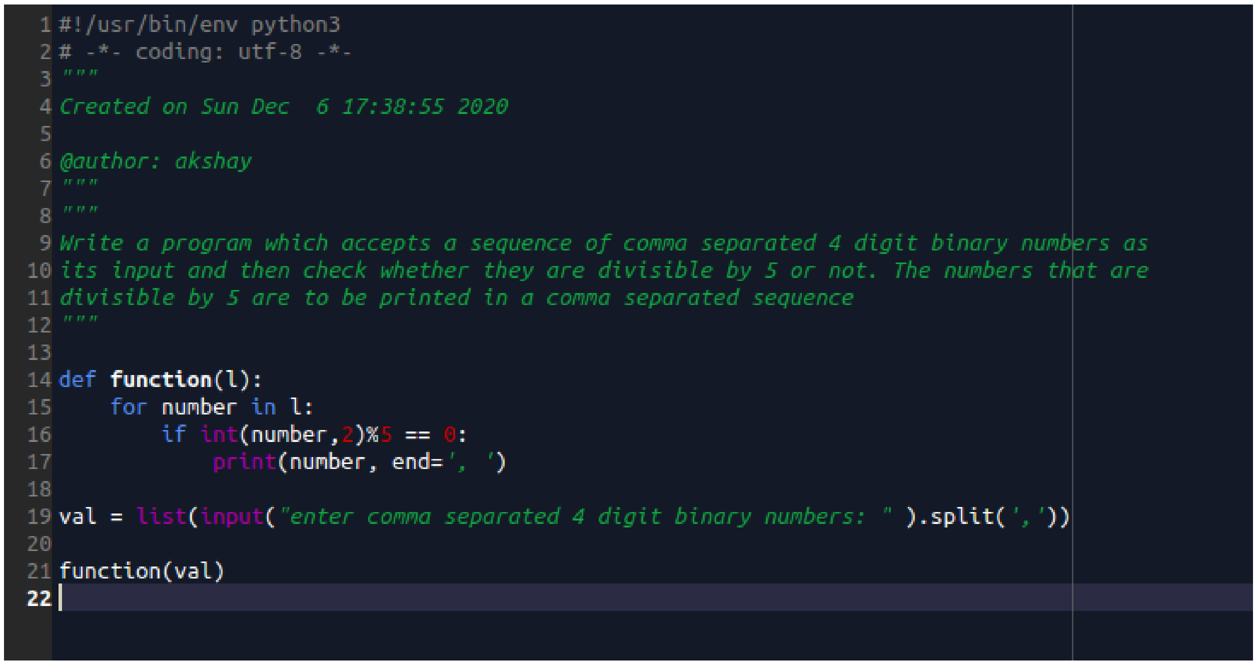
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  6 17:24:23 2020

@author: akshay
"""

"""
With a given integral number n, write a program to generate a dictionary that contains
(i,i*i) such that is an integral number between 1 and n (both included) and then the
 program should print the dictionary.
"""

def constructDictionary(n):
    dictionary={}
    for i in range(1,n+1):
        dictionary[i] = i*i;

    print(dictionary.items())
#     for elements in dictionary.items():
#         print(elements)


integer = int(input("enter an integer:" ))
constructDictionary(integer)
```

Results



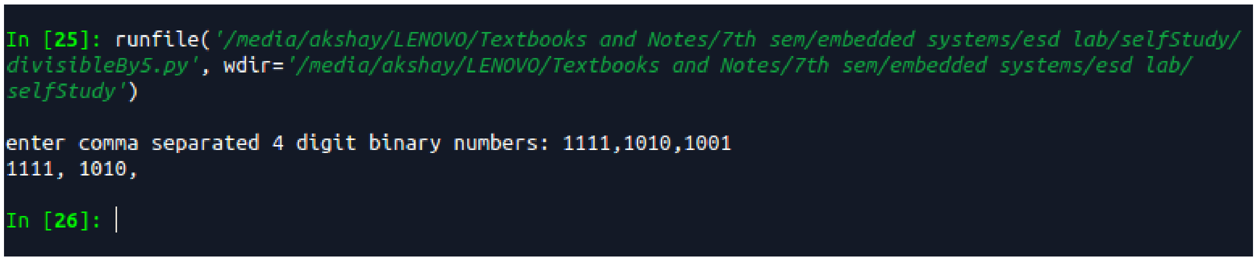
```
In [21]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/
dictionary.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/
selfStudy')

enter an integer:10
dict_items([(1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81), (10, 100)])

In [22]: 
```

5. Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.



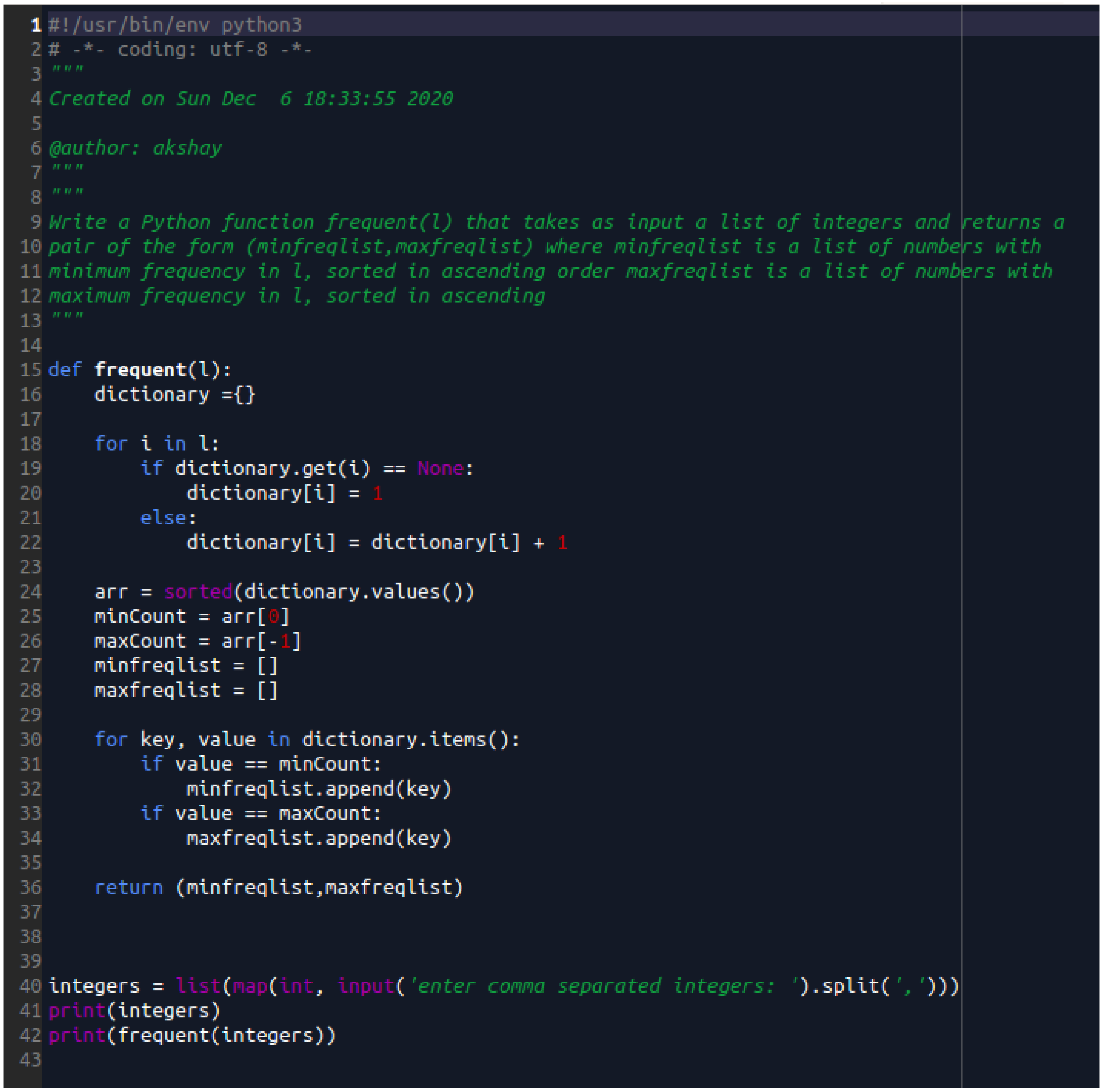
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  6 17:38:55 2020

@author: akshay
"""
"""
Write a program which accepts a sequence of comma separated 4 digit binary numbers as
its input and then check whether they are divisible by 5 or not. The numbers that are
divisible by 5 are to be printed in a comma separated sequence
"""

def function(l):
    for number in l:
        if int(number,2)%5 == 0:
            print(number, end=', ')

val = list(input("enter comma separated 4 digit binary numbers: " ).split(','))

function(val)

```

Results



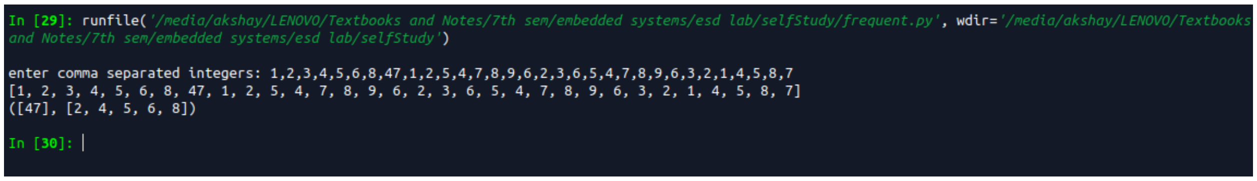
```
In [25]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/
divisibleBy5.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/
selfStudy')

enter comma separated 4 digit binary numbers: 1111,1010,1001
1111, 1010,

In [26]:
```

6. Write a Python function frequent(l) that takes as input a list of integers and returns a pair of the form (minfreqlist,maxfreqlist) where minfreqlist is a list of numbers with minimum frequency in l, sorted in ascending order maxfreqlist is a list of numbers with maximum frequency in l, sorted in ascending



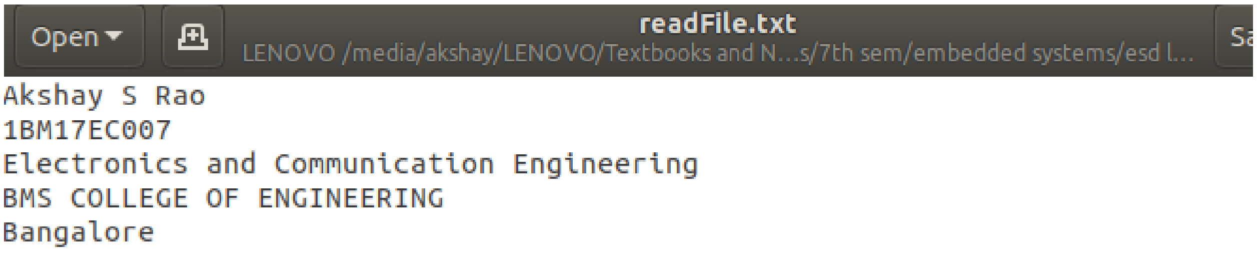
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  6 18:33:55 2020

@author: akshay
"""
"""
Write a Python function frequent(l) that takes as input a list of integers and returns a
pair of the form (minfreqlist,maxfreqlist) where minfreqlist is a list of numbers with
minimum frequency in l, sorted in ascending order maxfreqlist is a list of numbers with
maximum frequency in l, sorted in ascending
"""

def frequent(l):
    dictionary ={}

    for i in l:
        if dictionary.get(i) == None:
            dictionary[i] = 1
        else:
            dictionary[i] = dictionary[i] + 1

    arr = sorted(dictionary.values())
    minCount = arr[0]
    maxCount = arr[-1]
    minfreqlist = []
    maxfreqlist = []

    for key, value in dictionary.items():
        if value == minCount:
            minfreqlist.append(key)
        if value == maxCount:
            maxfreqlist.append(key)

    return (minfreqlist,maxfreqlist)



integers = list(map(int, input('enter comma separated integers: ').split(',')))
print(integers)
print(frequent(integers))
```

Results



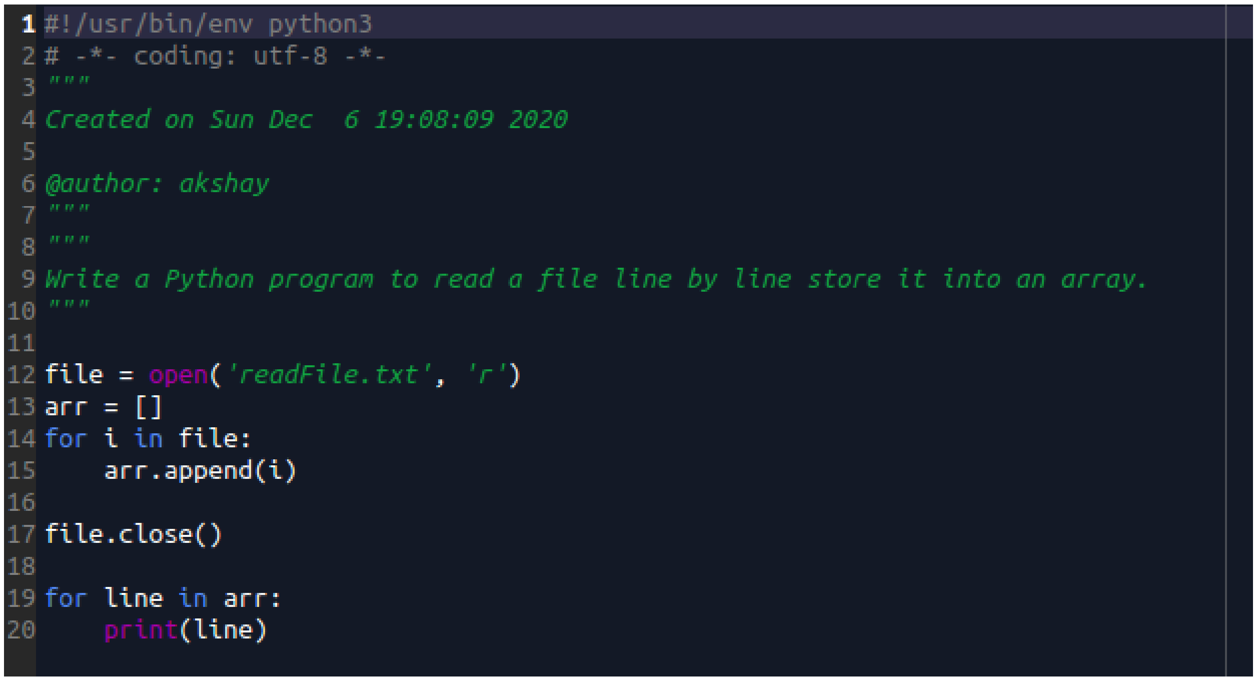
```
In [29]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/frequent.py', wdir='/media/akshay/LENOVO/Textbooks
and Notes/7th sem/embedded systems/esd lab/selfStudy')

enter comma separated integers: 1,2,3,4,5,6,8,47,1,2,5,4,7,8,9,6,2,3,6,5,4,7,8,9,6,3,2,1,4,5,8,7
[1, 2, 3, 4, 5, 6, 8, 47, 1, 2, 5, 4, 7, 8, 9, 6, 2, 3, 6, 5, 4, 7, 8, 9, 6, 3, 2, 1, 4, 5, 8, 7]
([47], [2, 4, 5, 6, 8])

In [30]:
```

7. Write a Python program to read a file line by line store it into an array.



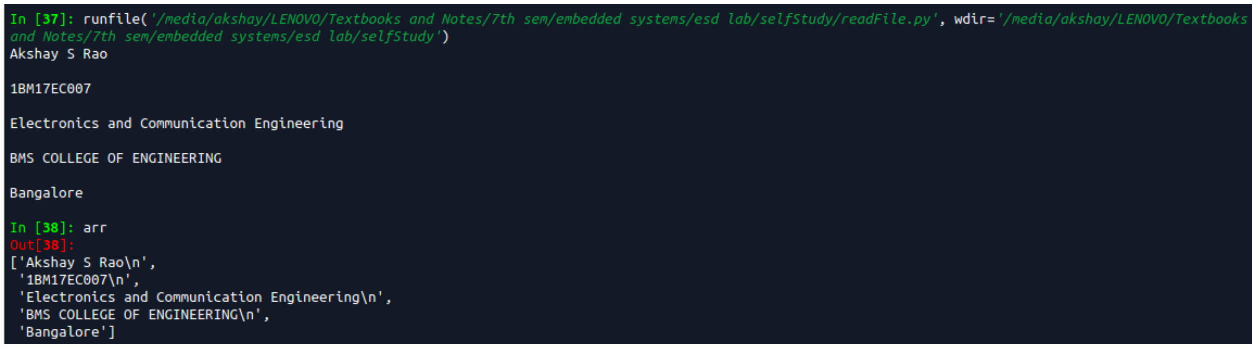
Open
readFile.txt
LENOVO /media/akshay/LENOVO/Textbooks and N...s/7th sem/embedded systems/esd l...
Sa

```
Akshay S Rao
1BM17EC007
Electronics and Communication Engineering
BMS COLLEGE OF ENGINEERING
Bangalore
```



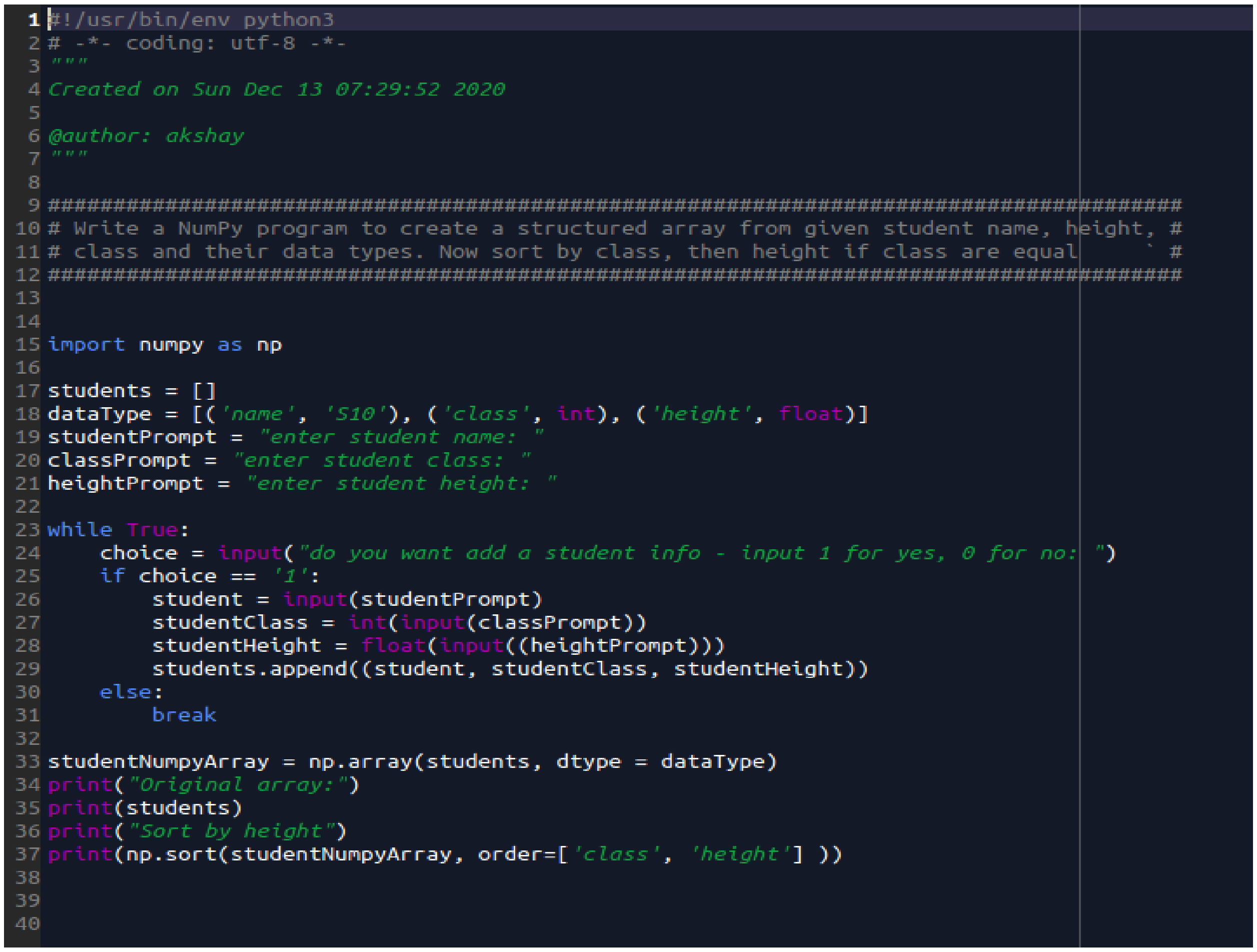
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  6 19:08:09 2020

@author: akshay
"""
"""
Write a Python program to read a file line by line store it into an array.
"""

file = open('readFile.txt', 'r')
arr = []
for i in file:
    arr.append(i)

file.close()

for line in arr:
    print(line)
```

Results



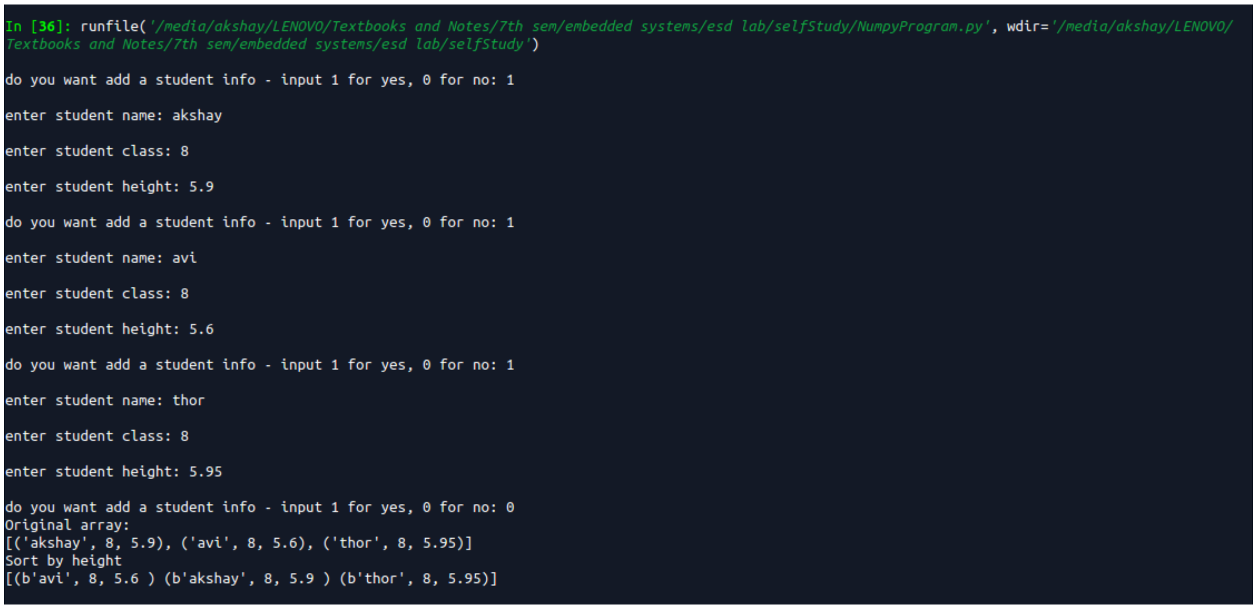
```
In [37]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/readFile.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy')
Akshay S Rao

1BM17EC007

Electronics and Communication Engineering

BMS COLLEGE OF ENGINEERING

Bangalore

In [38]: arr
Out[38]:
['Akshay S Rao\n',
 '1BM17EC007\n',
 'Electronics and Communication Engineering\n',
 'BMS COLLEGE OF ENGINEERING\n',
 'Bangalore']
```

8. Write a NumPy program to create a structured array from given student name, height,class and their data types. Now sort by class, then height if class are equal



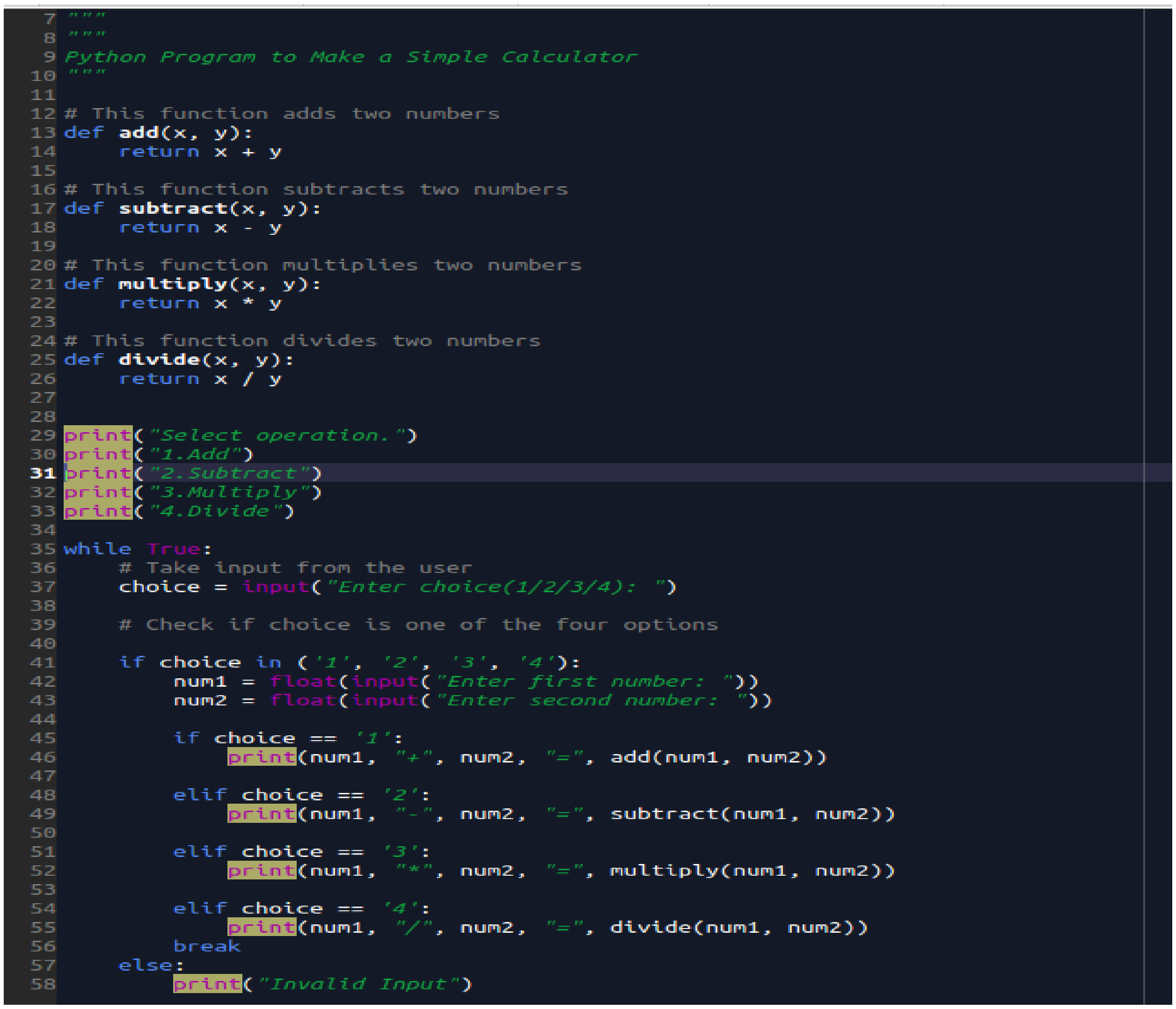
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 13 07:29:52 2020

@author: akshay
"""

########################################################################################
# Write a NumPy program to create a structured array from given student name, height, #
# class and their data types. Now sort by class, then height if class are equal      ` #
########################################################################################


import numpy as np

students = []
dataType = [('name', 'S10'), ('class', int), ('height', float)]
studentPrompt = "enter student name: "
classPrompt = "enter student class: "
heightPrompt = "enter student height: "

while True:
    choice = input("do you want add a student info - input 1 for yes, 0 for no: ")
    if choice == '1':
        student = input(studentPrompt)
        studentClass = int(input(classPrompt))
        studentHeight = float(input((heightPrompt)))
        students.append((student, studentClass, studentHeight))
    else:
        break

studentNumpyArray = np.array(students, dtype = dataType)
print("Original array:")
print(students)
print("Sort by height")
print(np.sort(studentNumpyArray, order=['class', 'height'] ))



```

Results



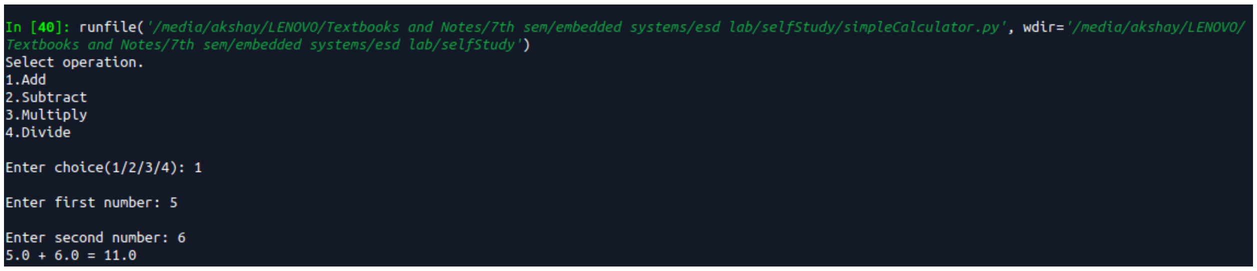
```
In [36]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/NumpyProgram.py', wdir='/media/akshay/LENOVO/
Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy')

do you want add a student info - input 1 for yes, 0 for no: 1

enter student name: akshay

enter student class: 8

enter student height: 5.9

do you want add a student info - input 1 for yes, 0 for no: 1

enter student name: avi

enter student class: 8

enter student height: 5.6

do you want add a student info - input 1 for yes, 0 for no: 1

enter student name: thor

enter student class: 8

enter student height: 5.95

do you want add a student info - input 1 for yes, 0 for no: 0
Original array:
[('akshay', 8, 5.9), ('avi', 8, 5.6), ('thor', 8, 5.95)]
Sort by height
[(b'avi', 8, 5.6 ) (b'akshay', 8, 5.9 ) (b'thor', 8, 5.95)]
```

9. Python Program to Make a Simple Calculator



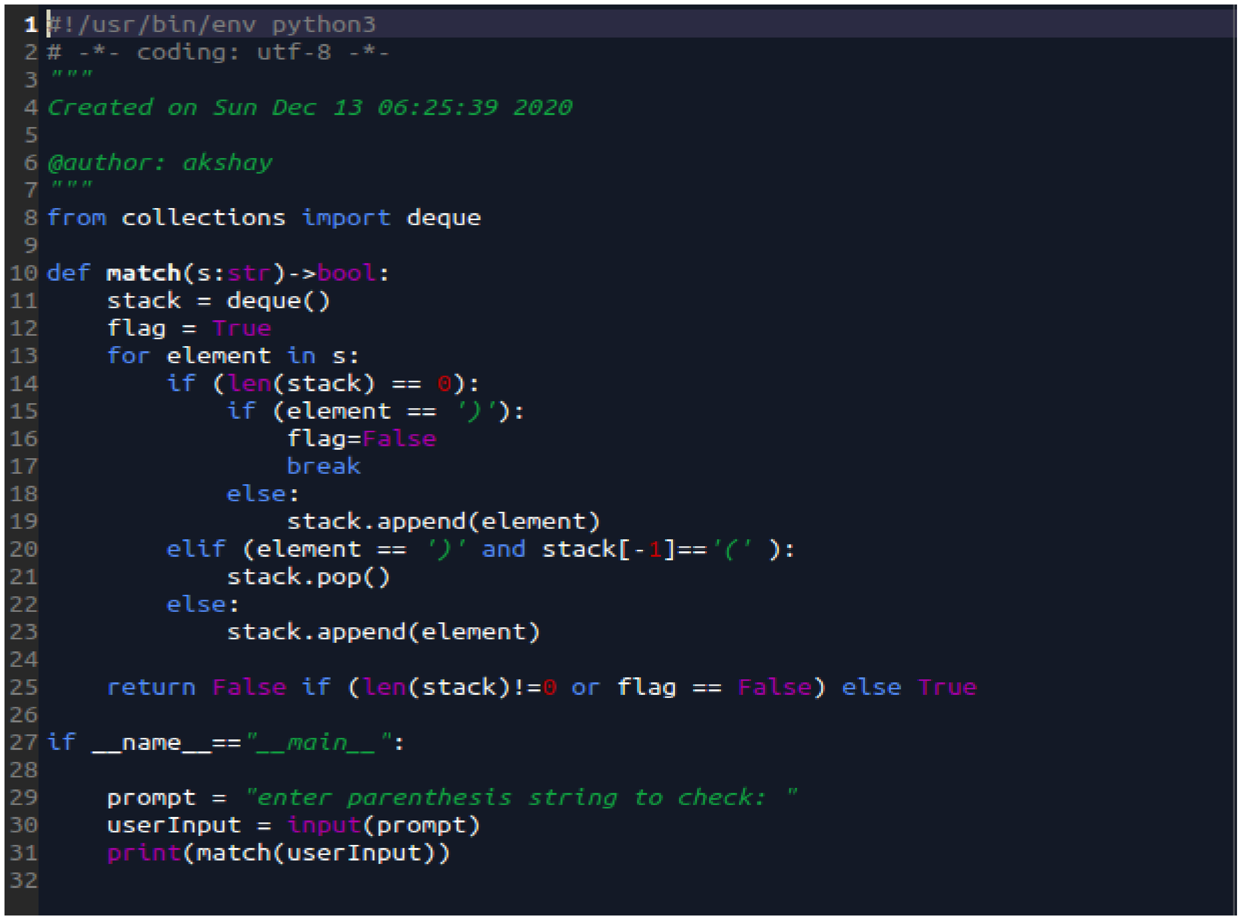
```python
"""
"""
Python Program to Make a Simple Calculator
"""

# This function adds two numbers
def add(x, y):
    return x + y

# This function subtracts two numbers
def subtract(x, y):
    return x - y

# This function multiplies two numbers
def multiply(x, y):
    return x * y

# This function divides two numbers
def divide(x, y):
    return x / y


print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

while True:
    # Take input from the user
    choice = input("Enter choice(1/2/3/4): ")

    # Check if choice is one of the four options

    if choice in ('1', '2', '3', '4'):
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))

        elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))

        elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))

        elif choice == '4':
            print(num1, "/", num2, "=", divide(num1, num2))
        break
    else:
        print("Invalid Input")
```

Results



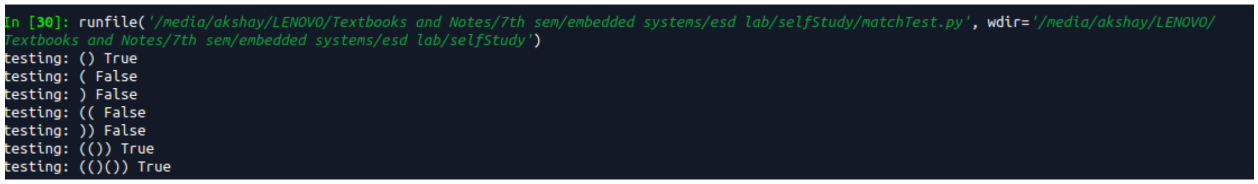
```
In [40]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/simpleCalculator.py', wdir='/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy')
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide

Enter choice(1/2/3/4): 1

Enter first number: 5

Enter second number: 6
5.0 + 6.0 = 11.0
```

10. Write a function match(s) that takes as input a string s and checks if the brackets "(" and ")" in s are matched: that is, every "(" has a matching ")" after it and every ")" has a matching "(" before it. Your function should ignore all other symbols that appear in s . Your function should return True if s has matched brackets and False if it does not.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 13 06:25:39 2020

@author: akshay
"""
from collections import deque

def match(s:str)->bool:
    stack = deque()
    flag = True
    for element in s:
        if (len(stack) == 0):
            if (element == ')'):
                flag=False
                break
            else:
                stack.append(element)
        elif (element == ')' and stack[-1]=='(' ):
            stack.pop()
        else:
            stack.append(element)

    return False if (len(stack)!=0 or flag == False) else True

if __name__=="__main__":

    prompt = "enter parenthesis string to check: "
    userInput = input(prompt)
    print(match(userInput))
```

Results



```
In [30]: runfile('/media/akshay/LENOVO/Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy/matchTest.py', wdir='/media/akshay/LENOVO/
Textbooks and Notes/7th sem/embedded systems/esd lab/selfStudy')
testing: () True
testing: ( False
testing: ) False
testing: (( False
testing: )) False
testing: (()) True
testing: (()()) True
```

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

B. M. S. COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELAGAVI)

BANGALORE – 560019

2020-21

7TH SEMESTER SELF STUDY

IN

EMBEDDED SYSTEM DESIGN (16EC7DCESD)

SELF STUDY ON ROBOT OPERATING SYSTEM

BY

AKSHAY S RAO 1BM17EC007

Course Instructor
Dr. Kiran Bailey
Assistant Professor

INTRODUCTION

ROS is an open-source robot operating system. It is a set of software libraries and tools that help you build robot applications that work across a wide variety of robotic platforms. Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow Garage. Since 2013 managed by OSRF (Open Source Robotics Foundation).

ROS has two “sides”. The operating system side, which provides standard operating system services such as: hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, package management. The other side is a suite of user contributed packages that implement common robot functionality such as SLAM, planning, perception, vision, manipulation, etc

ROS has certain philosophy

Peer to Peer : ROS systems consist of many small programs (nodes) which connect to each other and continuously exchange messages.

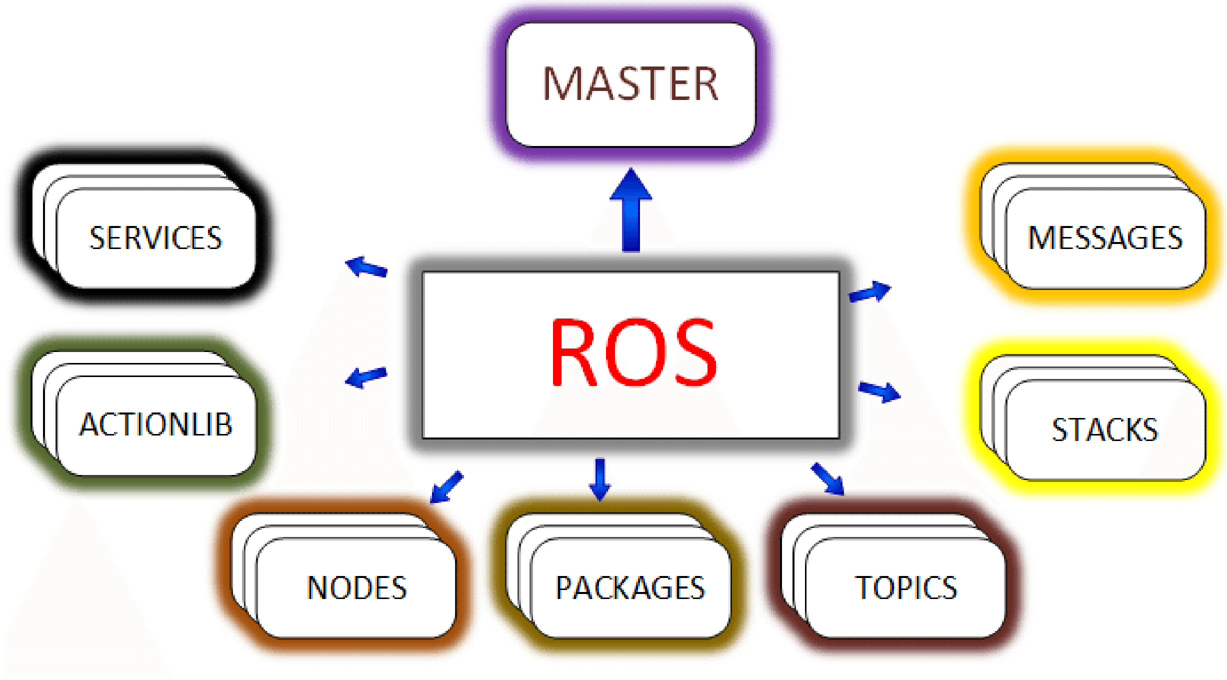
Tools-based: There are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc.

Multi-Lingual: ROS software modules can be written in any language for which a client library has been written. Currently client libraries exist for C++, Python, LISP, Java, JavaScript, MATLAB, Ruby, and more.

Thin: The ROS conventions encourage contributors to create stand-alone libraries/packages and then wrap those libraries so they send and receive messages to/from other ROS modules.

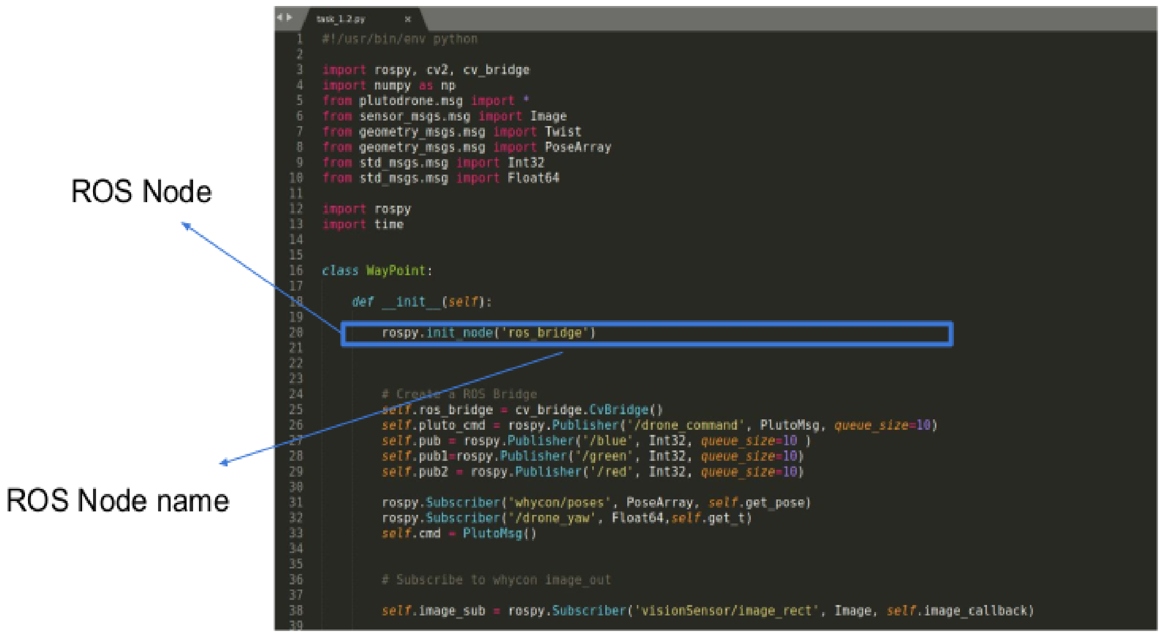
It also has free & open source, community-based, repositories

ROS Core Concepts



ROS Nodes

ROS Nodes are single-purposed executable programs for example it may represent sensor driver(s), actuator driver(s), map building, planner, UI, etc. It is Individually compiled, executed, and managed. Nodes are written using a ROS client library. Libraries exist for for both Python(rospy) and C++(roscpp). Nodes can publish or subscribe to a Topic. Nodes can also provide or use a Service or an Action.



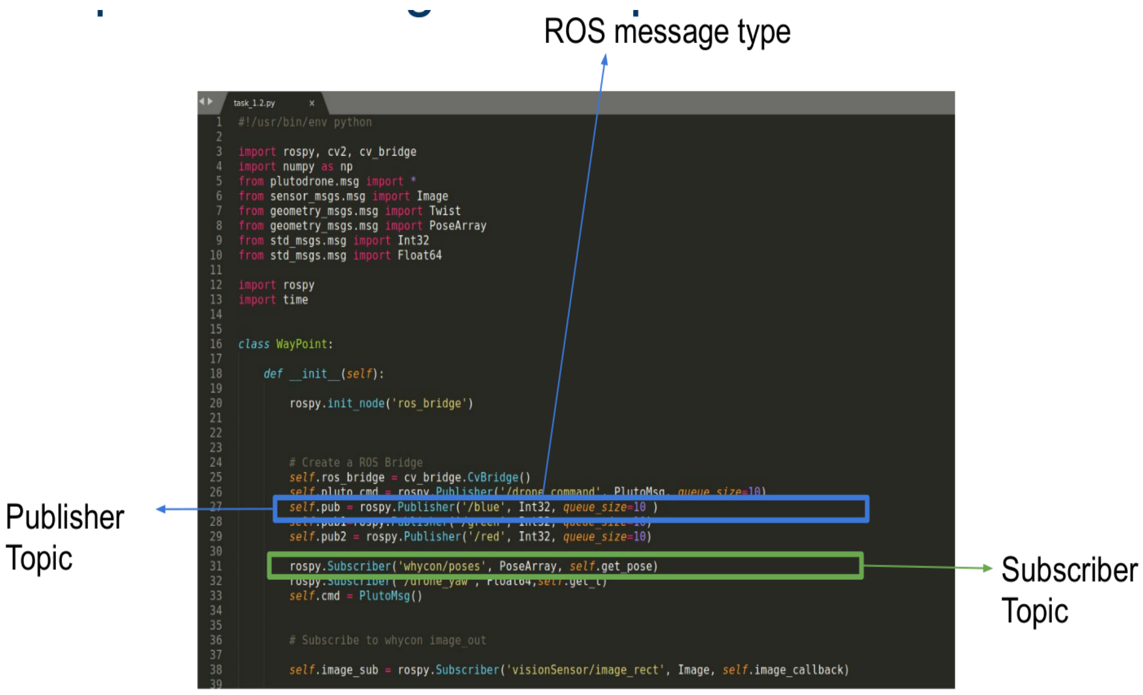
```python
#!/usr/bin/env python

import rospy, cv2, cv_bridge
import numpy as np
from plutodrone.msg import *
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist
from geometry_msgs.msg import PoseArray
from std_msgs.msg import Int32
from std_msgs.msg import Float64

import rospy
import time


class WayPoint:

    def __init__(self):

        rospy.init_node('ros_bridge')



        # Create a ROS Bridge
        self.ros_bridge = cv_bridge.CvBridge()
        self.pluto_cmd = rospy.Publisher('/drone_command', PlutoMsg, queue_size=10)
        self.pub = rospy.Publisher('/blue', Int32, queue_size=10 )
        self.pub1=rospy.Publisher('/green', Int32, queue_size=10)
        self.pub2 = rospy.Publisher('/red', Int32, queue_size=10)

        rospy.Subscriber('whycon/poses', PoseArray, self.get_pose)
        rospy.Subscriber('/drone_yaw', Float64,self.get_t)
        self.cmd = PlutoMsg()


        # Subscribe to whycon image_out

        self.image_sub = rospy.Subscriber('visionSensor/image_rect', Image, self.image_callback)

```

ROS Topics and ROS Messages

Topic: named stream of messages with a defined type. For example Data from a range-finder might be sent on a topic called scan, with a message of type LaserScan.

Nodes communicate with each other by publishing messages to topics and it has Publish/Subscribe model.

ROS Messages are Strictly-typed data structures for inter-node communication, for example geometry_msgs/Twist is used to express velocity commands: Vector3 linear, Vector3 angular.



```python
#!/usr/bin/env python

import rospy, cv2, cv_bridge
import numpy as np
from plutodrone.msg import *
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist
from geometry_msgs.msg import PoseArray
from std_msgs.msg import Int32
from std_msgs.msg import Float64

import rospy
import time


class WayPoint:

    def __init__(self):

        rospy.init_node('ros_bridge')



        # Create a ROS Bridge
        self.ros_bridge = cv_bridge.CvBridge()
        self.pluto_cmd = rospy.Publisher('/drone_command', PlutoMsg, queue_size=10)
        self.pub = rospy.Publisher('/blue', Int32, queue_size=10 )
        self.pub1=rospy.Publisher('/green', Int32, queue_size=10)
        self.pub2 = rospy.Publisher('/red', Int32, queue_size=10)

        rospy.Subscriber('whycon/poses', PoseArray, self.get_pose)
        rospy.Subscriber('/drone_yaw', Float64,self.get_t)
        self.cmd = PlutoMsg()


        # Subscribe to whycon image_out

        self.image_sub = rospy.Subscriber('visionSensor/image_rect', Image, self.image_callback)

```

ROS Bags

Bags are the primary mechanism in ROS for data logging. Bags subscribe to one or more ROS topics, and store the serialized message data in a file as it is received. Bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

Problem Statement

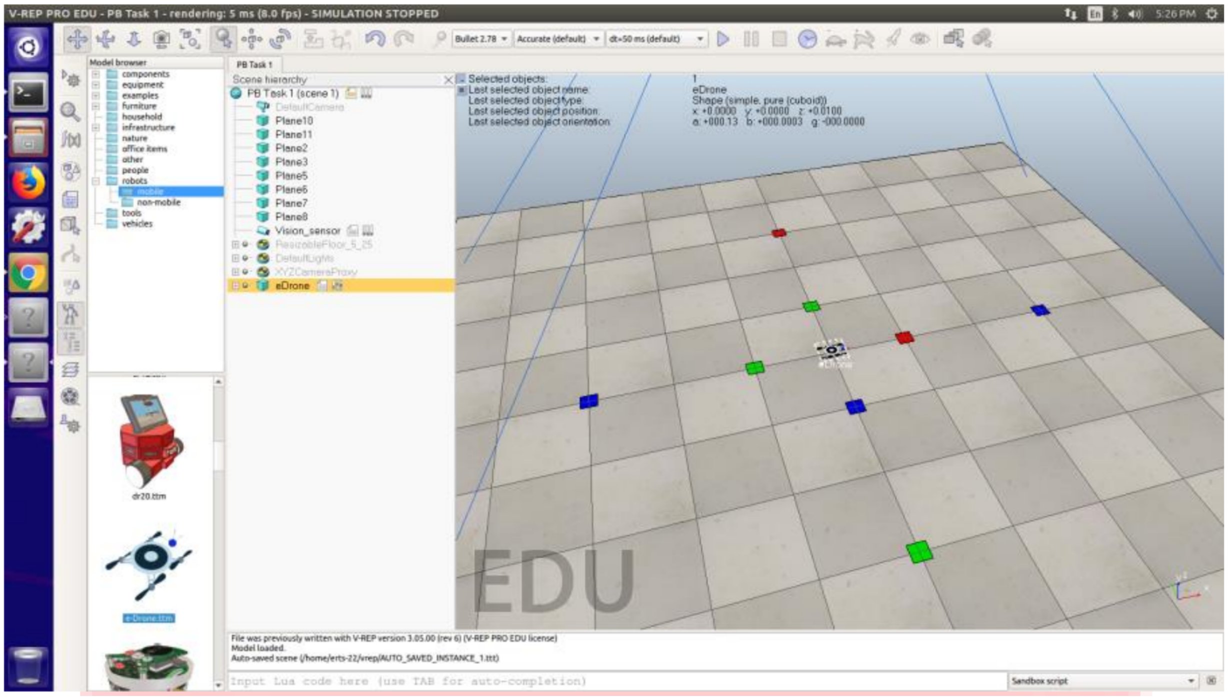
Implement marker based localization of quadcopter using ROS in a simulator.

- The drone should visit the given waypoint coordinates in the simulation using the PID control algorithm. Waypoints are in the form (x, y, z) :

[(-5.63, -5.63, 30), (5.57, -5.63, 30), (5.55, 5.54, 30), (-5.6, 5.54, 30), (0.0, 0.0, 30)]

- Within the same python node you must write code that would detect the colors placed in the simulation scene and publish them on their respective topics.

Simulation Scene

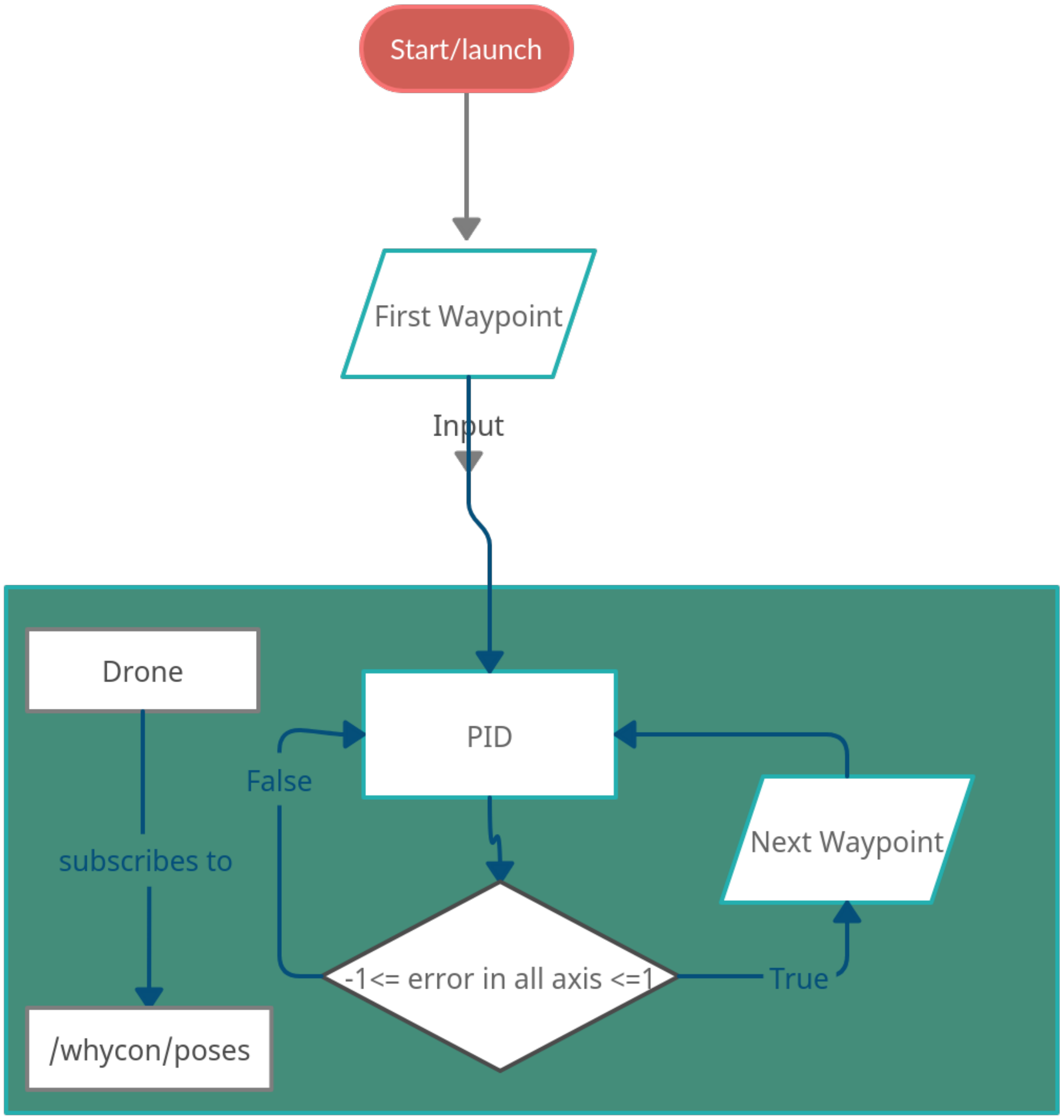


```
Selected objects:                    1
Last selected object name:           eDrone
Last selected object type:           Shape (simple, pure (cuboid))
Last selected object position:       x: +0.0000   y: +0.0000   z: +0.0100
Last selected object orientation:    a: +000.13   b: +000.0003   g: -000.0000
```

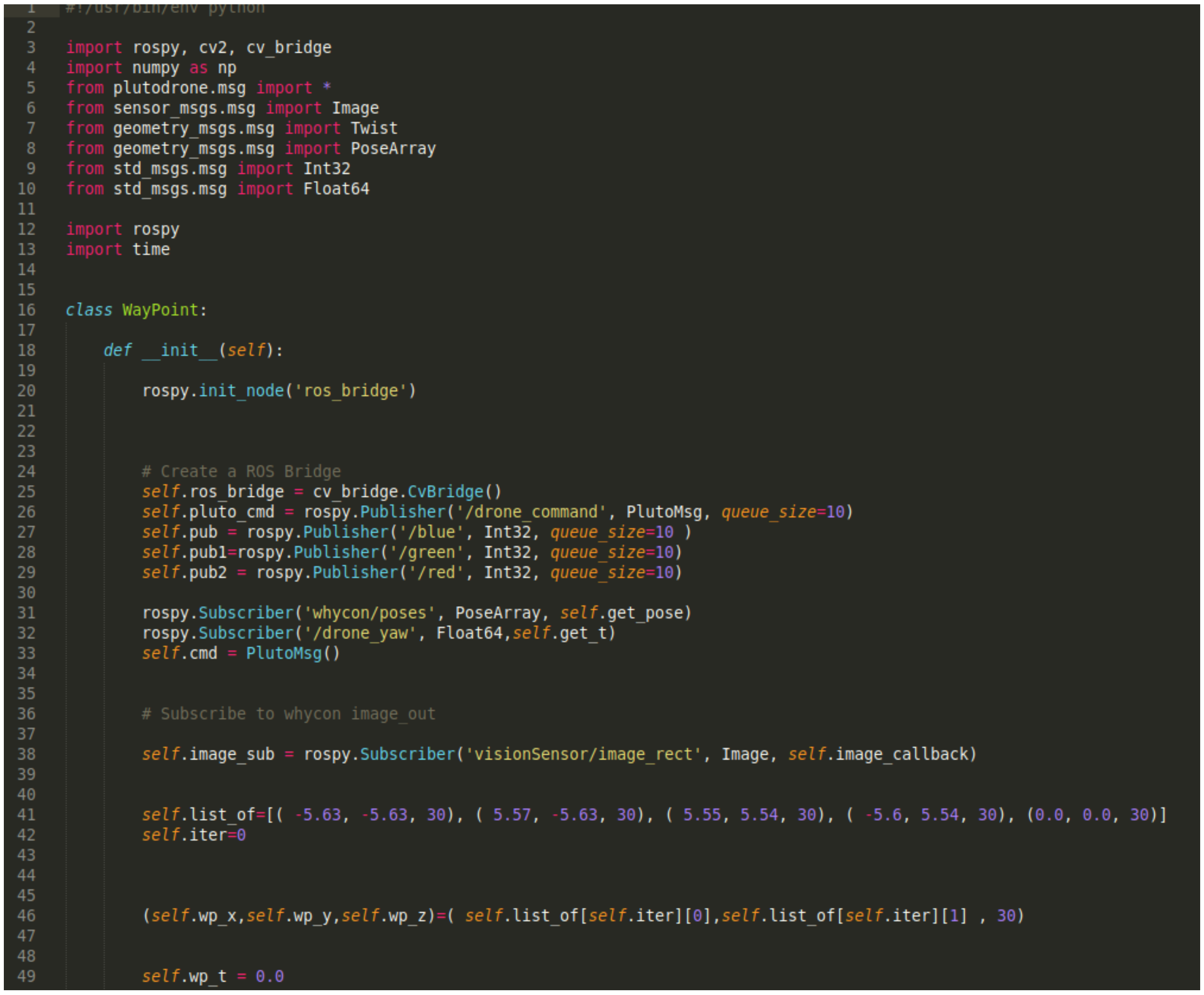
```
File was previously written with V-REP version 3.05.00 (rev 6) (V-REP PRO EDU license)
Model loaded.
Auto-saved scene (/home/erts-22/vrep/AUTO_SAVED_INSTANCE_1.ttt)
Input Lua code here (use TAB for auto-completion)
```

The above simulation scene shows a quadcopter and 3 unique colored patches.

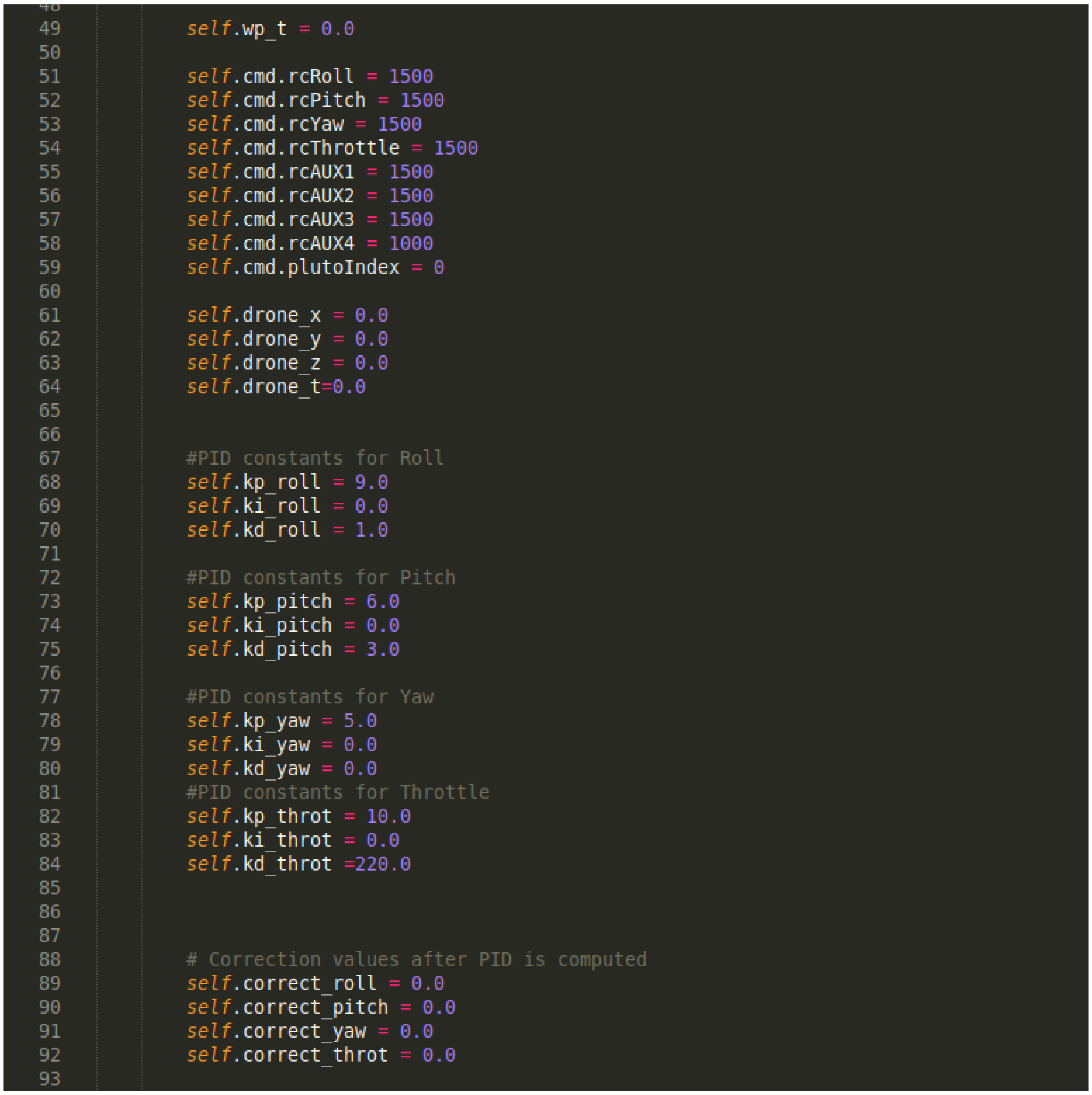
Flow Chart


Start/launch
First Waypoint
Input
Drone
subscribes to
/whycon/poses
PID
False
Next Waypoint
-1<= error in all axis <=1
True

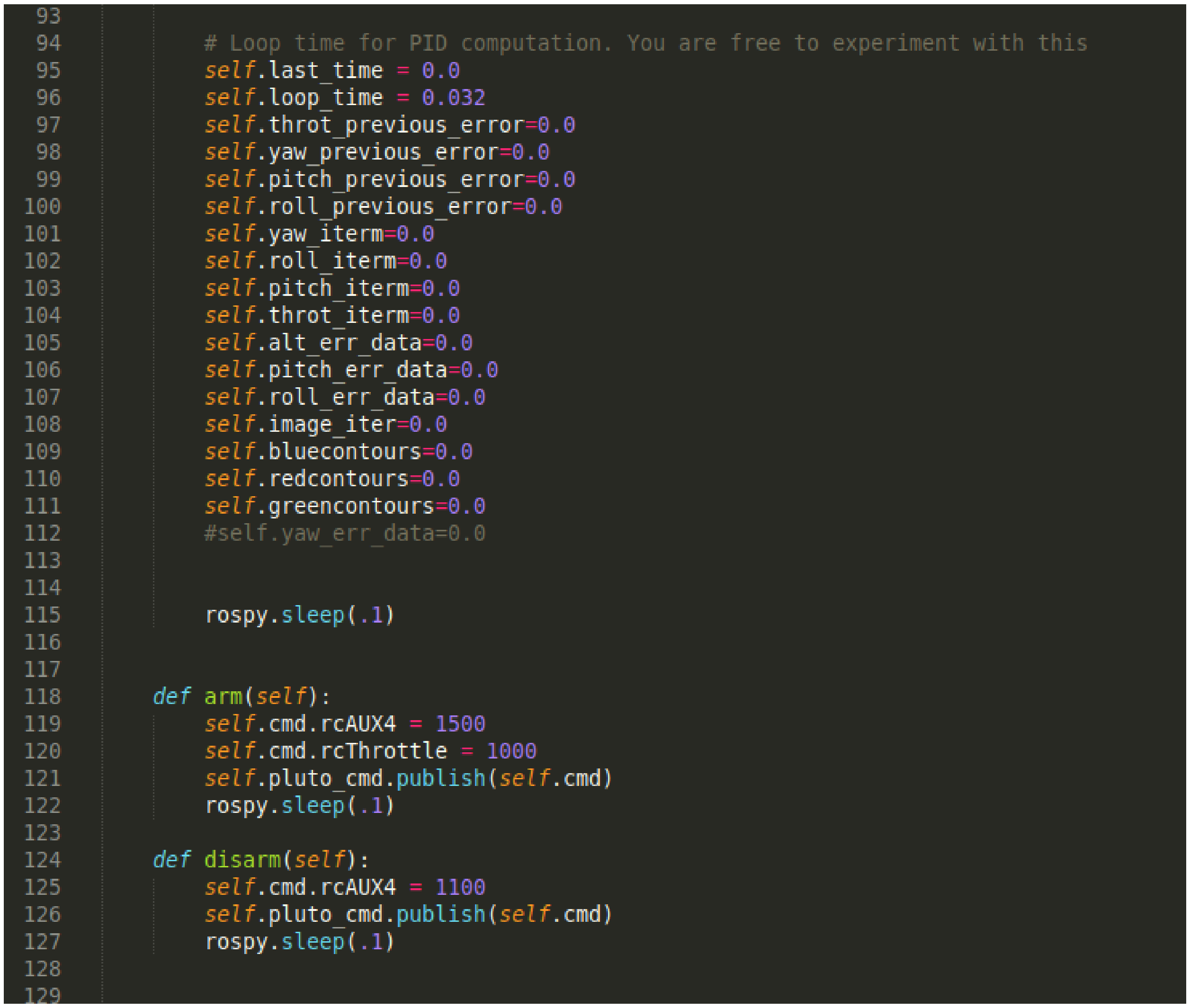
Code



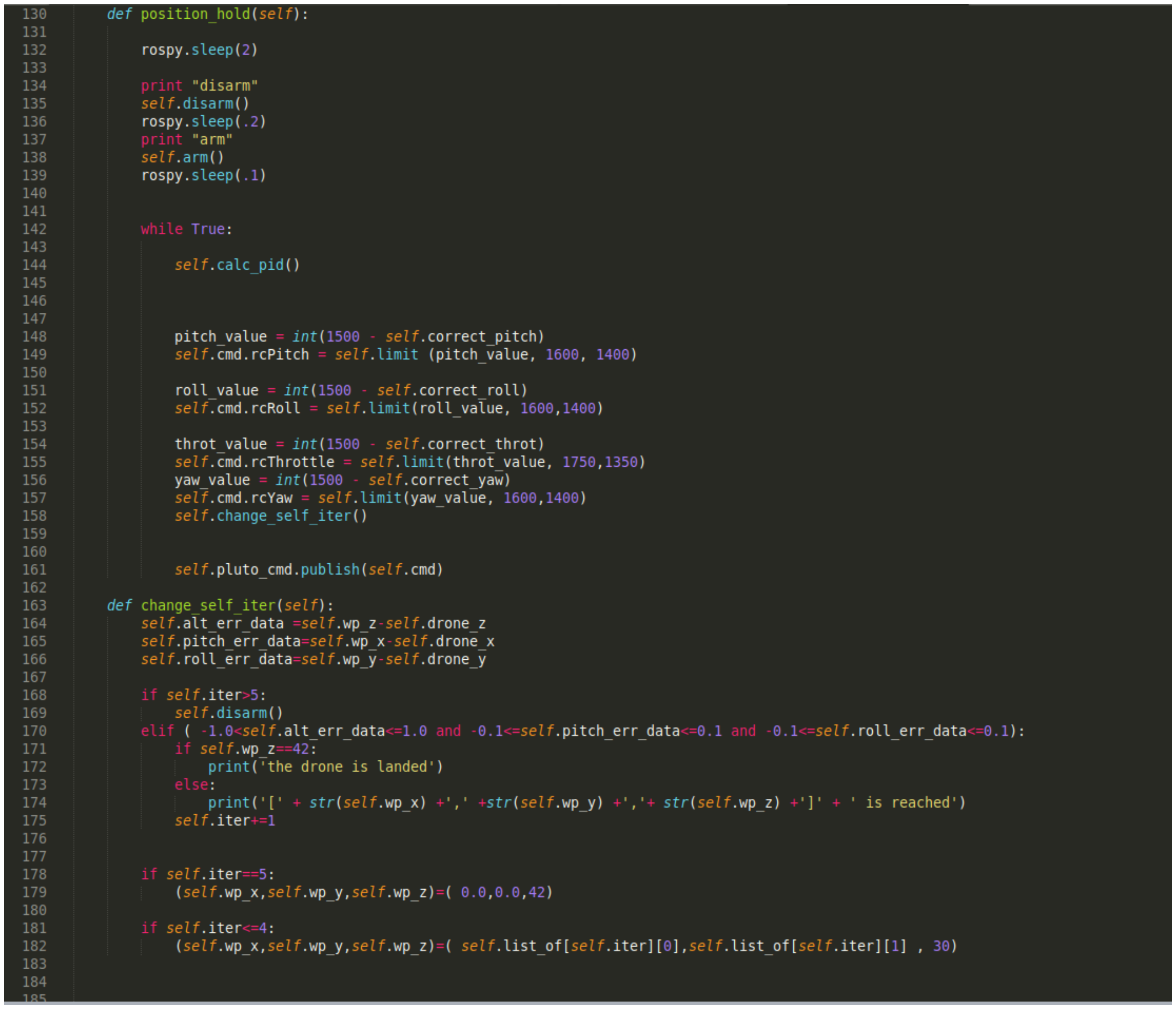
```python
#!/usr/bin/env python

import rospy, cv2, cv_bridge
import numpy as np
from plutodrone.msg import *
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist
from geometry_msgs.msg import PoseArray
from std_msgs.msg import Int32
from std_msgs.msg import Float64

import rospy
import time


class WayPoint:

    def __init__(self):

        rospy.init_node('ros_bridge')



        # Create a ROS Bridge
        self.ros_bridge = cv_bridge.CvBridge()
        self.pluto_cmd = rospy.Publisher('/drone_command', PlutoMsg, queue_size=10)
        self.pub = rospy.Publisher('/blue', Int32, queue_size=10 )
        self.pub1=rospy.Publisher('/green', Int32, queue_size=10)
        self.pub2 = rospy.Publisher('/red', Int32, queue_size=10)

        rospy.Subscriber('whycon/poses', PoseArray, self.get_pose)
        rospy.Subscriber('/drone_yaw', Float64,self.get_t)
        self.cmd = PlutoMsg()


        # Subscribe to whycon image_out

        self.image_sub = rospy.Subscriber('visionSensor/image_rect', Image, self.image_callback)


        self.list_of=[( -5.63, -5.63, 30), ( 5.57, -5.63, 30), ( 5.55, 5.54, 30), ( -5.6, 5.54, 30), (0.0, 0.0, 30)]
        self.iter=0



        (self.wp_x,self.wp_y,self.wp_z)=( self.list_of[self.iter][0],self.list_of[self.iter][1] , 30)


        self.wp_t = 0.0
```



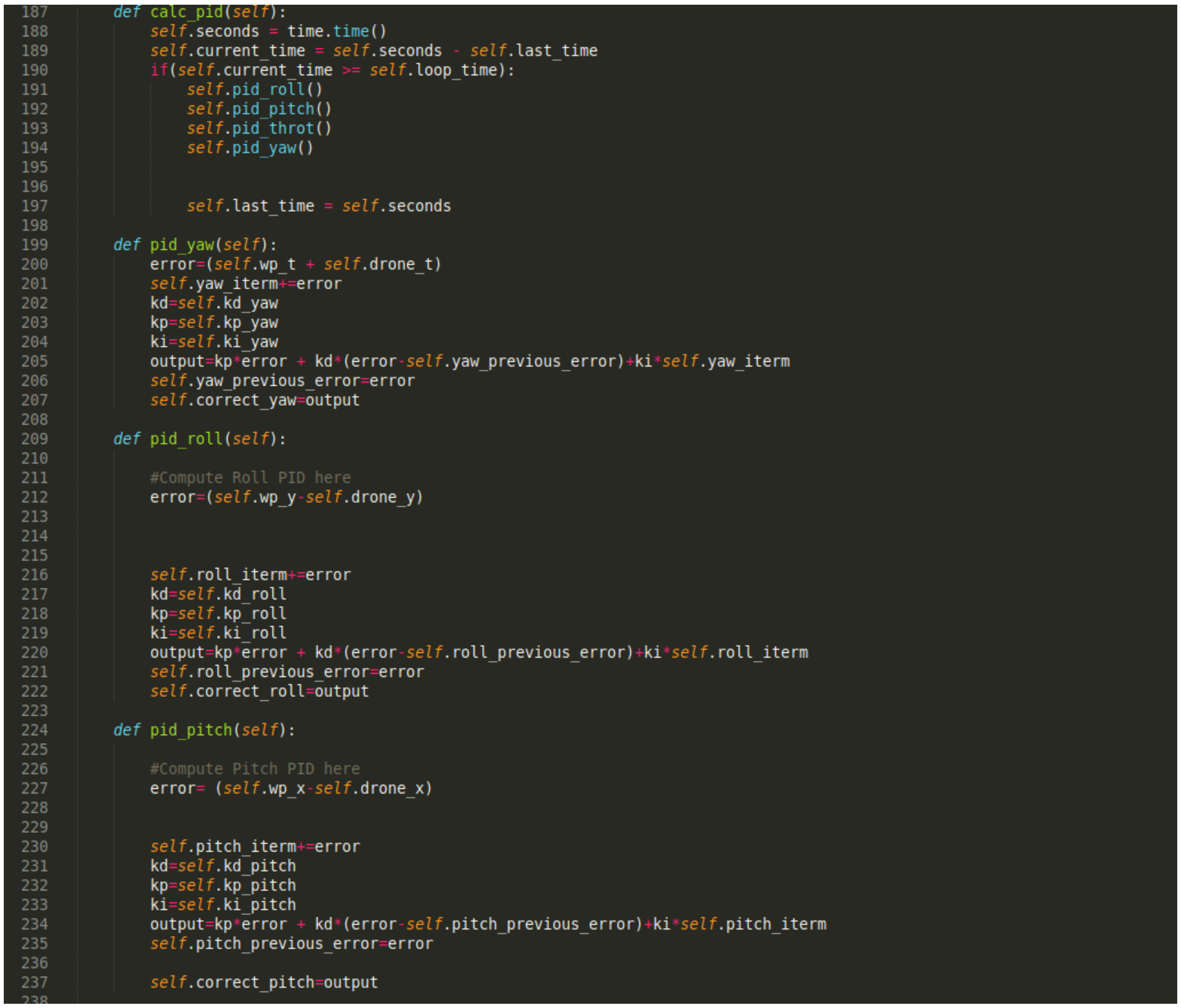
```
        self.wp_t = 0.0

        self.cmd.rcRoll = 1500
        self.cmd.rcPitch = 1500
        self.cmd.rcYaw = 1500
        self.cmd.rcThrottle = 1500
        self.cmd.rcAUX1 = 1500
        self.cmd.rcAUX2 = 1500
        self.cmd.rcAUX3 = 1500
        self.cmd.rcAUX4 = 1000
        self.cmd.plutoIndex = 0

        self.drone_x = 0.0
        self.drone_y = 0.0
        self.drone_z = 0.0
        self.drone_t=0.0


        #PID constants for Roll
        self.kp_roll = 9.0
        self.ki_roll = 0.0
        self.kd_roll = 1.0

        #PID constants for Pitch
        self.kp_pitch = 6.0
        self.ki_pitch = 0.0
        self.kd_pitch = 3.0

        #PID constants for Yaw
        self.kp_yaw = 5.0
        self.ki_yaw = 0.0
        self.kd_yaw = 0.0
        #PID constants for Throttle
        self.kp_throt = 10.0
        self.ki_throt = 0.0
        self.kd_throt =220.0



        # Correction values after PID is computed
        self.correct_roll = 0.0
        self.correct_pitch = 0.0
        self.correct_yaw = 0.0
        self.correct_throt = 0.0

```



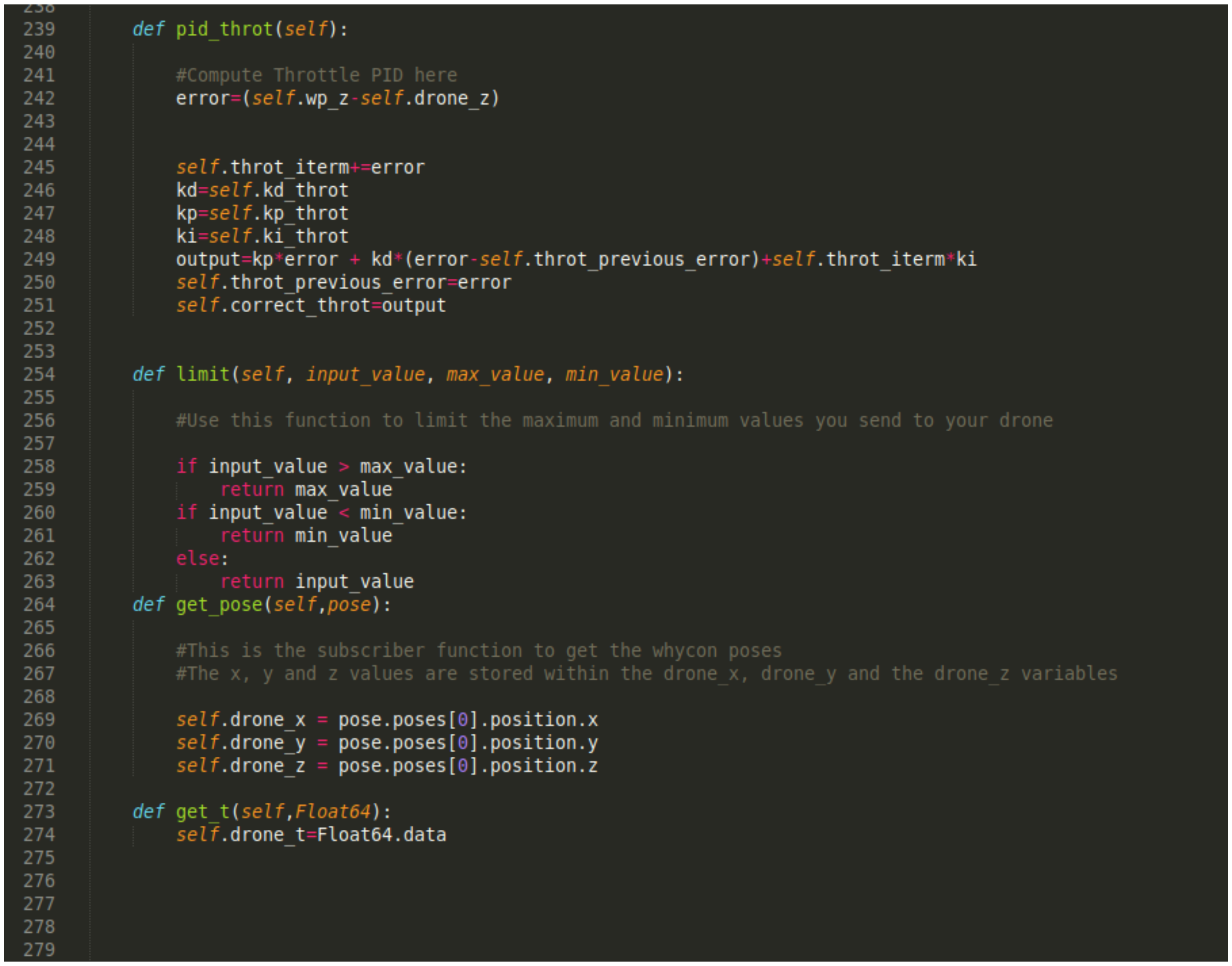
```
        # Loop time for PID computation. You are free to experiment with this
        self.last_time = 0.0
        self.loop_time = 0.032
        self.throt_previous_error=0.0
        self.yaw_previous_error=0.0
        self.pitch_previous_error=0.0
        self.roll_previous_error=0.0
        self.yaw_iterm=0.0
        self.roll_iterm=0.0
        self.pitch_iterm=0.0
        self.throt_iterm=0.0
        self.alt_err_data=0.0
        self.pitch_err_data=0.0
        self.roll_err_data=0.0
        self.image_iter=0.0
        self.bluecontours=0.0
        self.redcontours=0.0
        self.greencontours=0.0
        #self.yaw_err_data=0.0


        rospy.sleep(.1)


    def arm(self):
        self.cmd.rcAUX4 = 1500
        self.cmd.rcThrottle = 1000
        self.pluto_cmd.publish(self.cmd)
        rospy.sleep(.1)

    def disarm(self):
        self.cmd.rcAUX4 = 1100
        self.pluto_cmd.publish(self.cmd)
        rospy.sleep(.1)

```



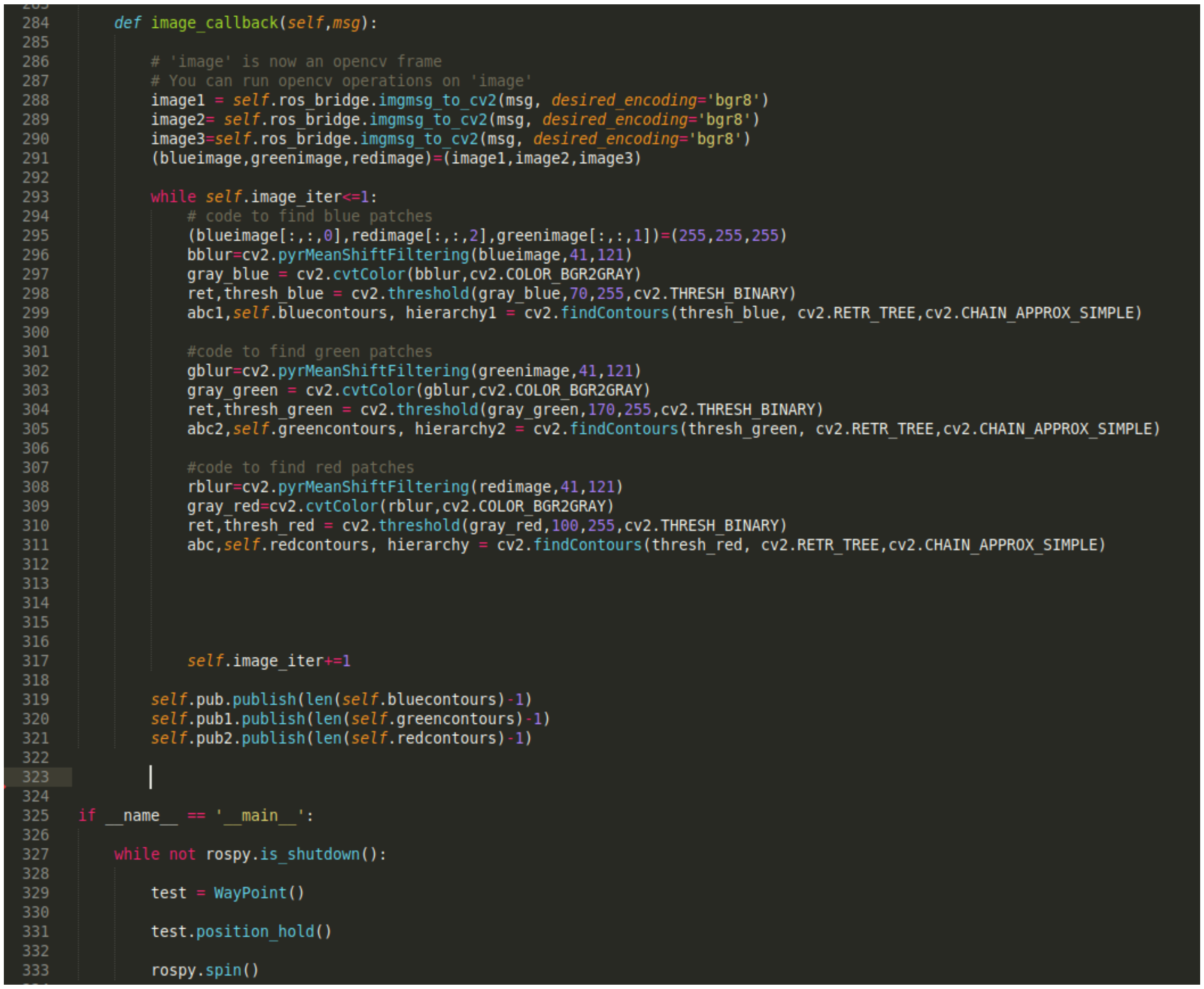
```python
    def position_hold(self):

        rospy.sleep(2)

        print "disarm"
        self.disarm()
        rospy.sleep(.2)
        print "arm"
        self.arm()
        rospy.sleep(.1)


        while True:

            self.calc_pid()



            pitch_value = int(1500 - self.correct_pitch)
            self.cmd.rcPitch = self.limit (pitch_value, 1600, 1400)

            roll_value = int(1500 - self.correct_roll)
            self.cmd.rcRoll = self.limit(roll_value, 1600,1400)

            throt_value = int(1500 - self.correct_throt)
            self.cmd.rcThrottle = self.limit(throt_value, 1750,1350)
            yaw_value = int(1500 - self.correct_yaw)
            self.cmd.rcYaw = self.limit(yaw_value, 1600,1400)
            self.change_self_iter()


            self.pluto_cmd.publish(self.cmd)

    def change_self_iter(self):
        self.alt_err_data =self.wp_z-self.drone_z
        self.pitch_err_data=self.wp_x-self.drone_x
        self.roll_err_data=self.wp_y-self.drone_y

        if self.iter>5:
            self.disarm()
        elif ( -1.0<self.alt_err_data<=1.0 and -0.1<=self.pitch_err_data<=0.1 and -0.1<=self.roll_err_data<=0.1):
            if self.wp_z==42:
                print('the drone is landed')
            else:
                print('[' + str(self.wp_x) +',' +str(self.wp_y) +','+ str(self.wp_z) +']' + ' is reached')
            self.iter+=1


        if self.iter==5:
            (self.wp_x,self.wp_y,self.wp_z)=( 0.0,0.0,42)

        if self.iter<=4:
            (self.wp_x,self.wp_y,self.wp_z)=( self.list_of[self.iter][0],self.list_of[self.iter][1] , 30)
```



```
    def calc_pid(self):
        self.seconds = time.time()
        self.current_time = self.seconds - self.last_time
        if(self.current_time >= self.loop_time):
            self.pid_roll()
            self.pid_pitch()
            self.pid_throt()
            self.pid_yaw()


            self.last_time = self.seconds

    def pid_yaw(self):
        error=(self.wp_t + self.drone_t)
        self.yaw_iterm+=error
        kd=self.kd_yaw
        kp=self.kp_yaw
        ki=self.ki_yaw
        output=kp*error + kd*(error-self.yaw_previous_error)+ki*self.yaw_iterm
        self.yaw_previous_error=error
        self.correct_yaw=output

    def pid_roll(self):

        #Compute Roll PID here
        error=(self.wp_y-self.drone_y)



        self.roll_iterm+=error
        kd=self.kd_roll
        kp=self.kp_roll
        ki=self.ki_roll
        output=kp*error + kd*(error-self.roll_previous_error)+ki*self.roll_iterm
        self.roll_previous_error=error
        self.correct_roll=output

    def pid_pitch(self):

        #Compute Pitch PID here
        error= (self.wp_x-self.drone_x)


        self.pitch_iterm+=error
        kd=self.kd_pitch
        kp=self.kp_pitch
        ki=self.ki_pitch
        output=kp*error + kd*(error-self.pitch_previous_error)+ki*self.pitch_iterm
        self.pitch_previous_error=error

        self.correct_pitch=output
```



```python
    def pid_throt(self):

        #Compute Throttle PID here
        error=(self.wp_z-self.drone_z)


        self.throt_iterm+=error
        kd=self.kd_throt
        kp=self.kp_throt
        ki=self.ki_throt
        output=kp*error + kd*(error-self.throt_previous_error)+self.throt_iterm*ki
        self.throt_previous_error=error
        self.correct_throt=output


    def limit(self, input_value, max_value, min_value):

        #Use this function to limit the maximum and minimum values you send to your drone

        if input_value > max_value:
            return max_value
        if input_value < min_value:
            return min_value
        else:
            return input_value
    def get_pose(self,pose):

        #This is the subscriber function to get the whycon poses
        #The x, y and z values are stored within the drone_x, drone_y and the drone_z variables

        self.drone_x = pose.poses[0].position.x
        self.drone_y = pose.poses[0].position.y
        self.drone_z = pose.poses[0].position.z

    def get_t(self,Float64):
        self.drone_t=Float64.data
```



```
    def image_callback(self,msg):

        # 'image' is now an opencv frame
        # You can run opencv operations on 'image'
        image1 = self.ros_bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
        image2= self.ros_bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
        image3=self.ros_bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
        (blueimage,greenimage,redimage)=(image1,image2,image3)

        while self.image_iter<=1:
            # code to find blue patches
            (blueimage[:,:,0],redimage[:,:,2],greenimage[:,:,1])=(255,255,255)
            bblur=cv2.pyrMeanShiftFiltering(blueimage,41,121)
            gray_blue = cv2.cvtColor(bblur,cv2.COLOR_BGR2GRAY)
            ret,thresh_blue = cv2.threshold(gray_blue,70,255,cv2.THRESH_BINARY)
            abc1,self.bluecontours, hierarchy1 = cv2.findContours(thresh_blue, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

            #code to find green patches
            gblur=cv2.pyrMeanShiftFiltering(greenimage,41,121)
            gray_green = cv2.cvtColor(gblur,cv2.COLOR_BGR2GRAY)
            ret,thresh_green = cv2.threshold(gray_green,170,255,cv2.THRESH_BINARY)
            abc2,self.greencontours, hierarchy2 = cv2.findContours(thresh_green, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

            #code to find red patches
            rblur=cv2.pyrMeanShiftFiltering(redimage,41,121)
            gray_red=cv2.cvtColor(rblur,cv2.COLOR_BGR2GRAY)
            ret,thresh_red = cv2.threshold(gray_red,100,255,cv2.THRESH_BINARY)
            abc,self.redcontours, hierarchy = cv2.findContours(thresh_red, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)





            self.image_iter+=1

        self.pub.publish(len(self.bluecontours)-1)
        self.pub1.publish(len(self.greencontours)-1)
        self.pub2.publish(len(self.redcontours)-1)



if __name__ == '__main__':

    while not rospy.is_shutdown():

        test = WayPoint()

        test.position_hold()

        rospy.spin()
```

Output:

Video demonstration can be found here: https://youtu.be/DpjnzrNq6PA

In the video you can see the drone visiting all the given waypoints and publishing the count of each colored patches on their respective topics namely /blue, /green, /red.