eYRC 2020-21: Nirikshak Bot (NB)

Coding Standard

[*Last Updated on:* ***27th January 2021, 11:00 Hrs***]

- The following documentation and comment styles are to be used for the code submitted by tl teams.
- Replace all the **<Description>** tags from the comments below to add appropriate content for your application.

For Python

A. File Level Comments

- Each user's code file should start with **File Level Comments** in the format as follows:

```
'''
# Team ID:          < Team-ID >
# Theme:            < Theme Name >
# Author List:      < Names of team members worked on this file separated by Comma
# Filename:         < Filename >
# Functions:        < Comma separated list of functions in this file >
# Global variables: < List of global variables defined in this file, None if no gl
'''
```

B. Function Level Comments

- Each function should have the following comment section just after the `def function_name()` line:

```
'''
Purpose:
---
< Short-text describing the purpose of this function >

Input Arguments:
---
`< name of 1st input argument >` :  [ < type of 1st input argument > ]
    < one-line description of 1st input argument >

`< name of 2nd input argument >` :  [ < type of 2nd input argument > ]
    < one-line description of 2nd input argument >

Returns:
---
`< name of 1st return argument >` :  [ < type of 1st return argument > ]
    < one-line description of 1st return argument >

`< name of 2nd return argument >` :  [ < type of 2nd return argument > ]
    < one-line description of 2nd return argument >

Example call:
---
< Example of how to call this function >
'''
```

C. Variable Level Comments

- In general the variable / function names should be descriptive enough to give a good idea of what the variable is used for.
- For example, variable names like `table1_kp_val`, `table4_kd_val` are preferable and makes your code readable.
- Variable names like `a`, `b` and `temp` are not acceptable variable names.
- In some cases, variable names might require some desciption for which the following format can be used:

```
# < variable name >: < one-line description of the variable and its use >
```

D. Implementation Level Comments

- In your implementation / actual code, you should have comments for tricky, non-obvious, interesting, or important parts of the code.
- The comments can be of the format as below:

```
# < Short-text describing what the code below is doing >
```

An Illustrative Example

- We provide sample comments in a rudimentry program that outputs the Fibonacci Series.
- Please note that this is not the complete and perfect example of generating Fibonacci Numbe but acts as a simple way to illustrate the coding style and comments explained above.

```
'''
# Team ID:          9999
# Theme:            Nirikshak Bot
# Author List:      e-Yantra1, e-Yantra2
# Filename:         fibonacci.py
# Functions:        print_fibonacci_series, main
# Global variables: None
'''

import os, sys
```

```
def print_fibonacci_series(num_elements):
    '''
    Purpose:
    ---
    Prints the first num_elements of the Fibonnaci series.
    The next element of the series is given by:
        next_element = current_element + prev_element
    The code loops for num_elements and prints out the next element.

    Input Arguments:
    ---
    `num_elements` :  [ int ]
        number of elements in Fibonacci series to be printed

    Returns:
    ---
    None

    Example call:
    ---
    print_fibonacci_series(10)
    '''
    first = 0
    second = 1
    print('First ', num_elements, 'terms of Fibonacci series are:')
    print(first)
    print(second)

    for i in range(num_elements-2):
        # the next element is equal to the sum of the current element (second vari
        # and the previous element (first variable)
        next_element = first + second
        # first element becomes the second element and second element becomes the
        # next element for the next loop iteration
        first = second
        second = next_element
        print(next_element)

def main():
    '''
    Purpose:
    ---
    Asks the user to input the number of elements required from the Fibonacci Seri
    and call the function print_fibonacci_series.

    Input Arguments:
    ---
    None

    Returns:
    ---
    None

    Example call:
    ---
    Called automatically by the Operating System
    '''
    # Ask the user to input the number of elements required
    num_elements = input("Enter the number of terms: ")
    num_elements = int(num_elements)

    # Call the function print_fibonacci_series to print the first
    # num_elements of the Fibonacci Series
    print_fibonacci_series(num_elements)

# Function Name:    main (built in)
#        Inputs:    None
#       Outputs:    None
#       Purpose:    To call the main() function to print the Fibonacci series.
if __name__ == "__main__":
    main()
```

For Lua

A. File Level Comments

- Each user's code file should start with **File Level Comments** in the format as follows:

```
--[[
# Team ID:          < Team-ID >
# Theme:            < Theme Name >
# Author List:      < Names of team members worked on this file separated by Comma
# Filename:         < Filename >
# Functions:        < Comma separated list of functions in this file >
# Global variables: < List of global variables defined in this file, None if no gl
]]--
```

B. Function Level Comments

- Each function should have the following comment section just after the **`def function_name()`** line:

```
--[[
Purpose:
---
< Short-text describing the purpose of this function >

Input Arguments:
---
`< name of 1st input argument >` :  [ < type of 1st input argument > ]
    < one-line description of 1st input argument >

`< name of 2nd input argument >` :  [ < type of 2nd input argument > ]
    < one-line description of 2nd input argument >

Returns:
---
`< name of 1st return argument >` :  [ < type of 1st return argument > ]
    < one-line description of 1st return argument >

`< name of 2nd return argument >` :  [ < type of 2nd return argument > ]
    < one-line description of 2nd return argument >

Example call:
---
< Example of how to call this function >
]]--
```

C. Variable Level Comments

- In general the variable / function names should be descriptive enough to give a good idea of what the variable is used for.
- For example, variable names like `table1_kp_val`, `table4_kd_val` are preferable and makes your code readable.
- Variable names like `a`, `b` and `temp` are not acceptable variable names.
- In some cases, variable names might require some desciption for which the following format can be used:

```
-- < variable name >: < one-line description of the variable and its use >
```

D. Implementation Level Comments

- In your implementation / actual code, you should have comments for tricky, non-obvious, interesting, or important parts of the code.
- The comments can be of the format as below:

```
-- < Short-text describing what the code below is doing >
```

Following a coding style might look to be tedious at first but is one of the most important thing to be doen while developign any piece of code. This ensures that it is readable so that others can understand what your code is doing. Even you yourself may find it useful after some time!

"Any fool can write code that a computer can understand. Good programmers write code th humans can understand." - **Martin Fowler**

"Programs must be written for people to read, and only incidentally for machines to execute - **Hal Abelson and Gerald Jay Sussman**

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." - **Rick Osborne**

ALL THE BEST