YCTT Python Practice

V1.0

Contents

1.Simple

1.1 Calculation

Create function named calculation() that will receive a string and 2 numbers.The function will print the string followed by ":" and the sum of the two numbers

----

1.2 Join

Create function named join() that receives 3 integer values lower than 255. The 3 numbers represent byte 1 , byte 2, and byte 3 of a number.

The function will join the 3 number in a single number and return the value.

1.3 Display

Create a function that receives a list of numbers as parameter. The function is named display() and prints the numbers as decimal and hex.

1.4 Biggest

Create a function named biggest () that receives 5 numbers and returns the biggest one.

1.5 Longest Word

Write a function named find_longest_word() that takes a sentence and returns the length of the longest word.

1.6 Structure

Write a function named structure() that takes a string and returns the number of vowels ,consons.

1.7 Histogram

Define a procedure histogram() that takes a list of integers and a list of names and prints a histogram to the screen. For example, histogram([4, 9, 7],["horizontal","vertical","empty"]) should print the following:

```
horizontal | ****
vertical   | *********
empty      | *******
```

1.8 Sorted Lists

Create function named structure().The function will receive a list as a parameter. The list will contain series of name and numbers.

The function will create a list containing a list for each (name, numbers) pair. If the name contains the string "sorted" the numbers following the name must be sorted in the final list.

Input:

["values_1_sorted",3,7,2,9,"values_2",34,71,689,4,"values_3",35,46]

Output:

[["values_1_sorted",2,3,7,9],["values_2",34,71,689,4],["values_3",35,46]]

1.9 Walk List

Create function named walk(). The function will receive a list containing other embedded lists. The function must transform this structure in a single dimensioned list.

Input:[4,[45,[67,7,[9],6,[8,9,[12,3]]],[5,9]]]

Output:[4,45,67,7,9,6,8,9,12,3,5,9]

1.10 Byte Stream

Create function named print_stream_byte().

Given a list of numbers (numbers will be lower than 255) display them as a single string of hex bytes separated by space.

Input: [2,56,23,89,12,234,89,35] Output: 02 38 17 59 0C EA 59 23

1.11 Byte List

Create a function named number_to_bytelist().Receives as parameter a number and returns a string containing hex bytes.

Input: 23475641238152

Output: 15 59 D9 41 C2 88

1.12 Set Signal

Create a function named set_signal().

The function will receive a list of numbers (smaller then 255) , a start bit ,an end bit and a value.

The function will insert the value in the data set (list of numbers).

The first bit is the one on the right. Bytes numbers from 0 to 7.Bit 0 of a byte is on the right.

Input:

data   = [12,34,56,23,78]

startbit = 5

endbit   = 15

value   = 165

Output:

```
0C 22 38 14 A0
```

1.13 Get Signal Data

Create a function named get_ signal().

The function will receive an address (a string) of the form "Frame::Signal"

The function will return the signal start bit and end bit.

A global structure will contain all frames. Each frame can have many signals and each signal has an end bit and start bit.

You must choose a correct data structure for the global structure that holds all information

----

1.14 Nice Print

Create a function named nice_print().This function will receive a list and print the list in nice way. See example below.

Input: [1,2,[4,5,[6,7],8]]

Output:

```
[
   1,
   2,
   [
      4,
      5,
      [
         6,
            7
        ],
        8
    ]
]
```

1.15 Date Time

Create a function that prints the PC user name and the current date and time like this:

“DATE: Monday 12.06.2016   TIME:12:35:55.99”

1.16 Trig

Create a function that calculated the following formula: sqrt(x)+ sin(x) - cos(y)/100.

1.17 All XML

Create a function that receives a path as parameter and returns all the xml files in the folder.

1.18 2 Seconds

Create function that receives a string and prints it on the console screen every 2 seconds.

1.19 Ping

Create a function that receives as parameter a web address (http) and return True if we have access to that web page.

1.20 Special Numbers

Write a function 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.

1.21 Duplicate Words

Write a function that receives a large text and returns all unique words.

1.22 Fibonacci

Create a function that can calculate the Fibonacci sequence value of a number.

2.Complicated

2.1 DBC to Excel

Receive as input a DBC file containing multiple frames. The DBC file is a ASCI format file.

Create a python script that will receive from the command line the DBC file and create a excel file with all frames and their Identifiers. The excel file will contain all frame names, their identifiers and the name of their signals.

2.2 INI to XML

Receive as input an INI configuration file.

Create a python script that will take all data from the INI file received form the command line and create a XML file with the INI data. The XML file will contain for each section an element containing other elements that are the values.

2.3 ASC To XML

Receive as input A Canoe log file (*.asc file) with CAN communication containing multiple frames.

Create a script that will receive from the command line the log file and will create a XML file containing the log file data(all frame with ID, name, data, timestamp and length).

2.4 ASC Average Cycle Time

Receive as input A Canoe log file (*.asc file) with CAN communication containing multiple frames.

Create a script that received form the command line the log file and calculate the average cycle time of each frame

The output will be an excel file containing the Ids of the frame and their average cycle time.

2.5 Starter

You will receive a folder structure (folders with other folders).

Some folders will contain executable file s(*.exe,*.bat).

The script that will be executed from the command line with the path of the folder structure as parameter.

The script will call each executable from the folder structure .Capture the output string of the executable.

The output is an Excel file containing on each row the executable and the output string of the executable.

2.6 Log Signals

As input you will receive a Canoe log file(*.asc file) with CAN communication and a Excel File with one sheet name "Data".

The excel file sheet will contain 4 columns:

1.Column 1: A Frame ID

2.Column 2: A Signal Name

3.Column 3: The signal Start Bit

4.Column 4: The signal Length

The python script must receive the files from the command line.

The script must extract from the Canoe Log File the signal values and timestamps.

The script must create an excel file.

The excel file will be placed where the Canoe log file is.

The excel file must contain a sheet with multiple columns.

For each signal there must be 2 column:

1.A column containing timestamps(in whatever format you like)

2.A second column with the signal values.

2.7 Log Filter

As input you will receive a Canoe log file(*.asc file) with CAN communication.

The log file will be receive from the command line.

The script will take the log file extract from it only the data for the 2 frames and create an excel file.

The excel file will be placed in the same path where the log file is.

The excel file will contain the following columns:

1.The Frame ID

2.The Frame length

3.The Timestamp

4.The Frame direction

5.The Frame Data(each byte on a separate column)

The script must extract all CAN data from the log file for the two frames and place the data in a excel file.

2.8 Diag

Create 2 clases:

1.Class DiagData

- ATTRIBUTES
  - data (containing a hex string of bytes)
  - length (the number of bytes in the attribute data)
- METHODS:
  - as_list - will return the hex string as a list of integer bytes
- the constructor will receive the hex string and populate the length attribute

2.Class DiagRequest

- ATTRIBUTES:
  - request of type DiagData
  - response of type DiagData
  - mask of type DiagData
- METHODS::
- display - will print the 3 Diag Data attributes one under the other

----

2.9 Git

Create a class named Git. Git is a versioning program that received command line arguments.

The class must implement the following command as methods:

- clone
- init
- commit
- pull
- push

- checkout

- create branch

A user must be able to import or inherited the class and use the git system via the class.

.---

2.10 Log Monitor

Receive as input a Canoe CAN Trace.

Receive from the command line the following commands:

--monitor=SignalName

--condition=equal/bigger/smaller

--value=a numerical value

Read the can trace and monitor if the signal from the command line is equal all the times or smaller or bigger to the value given.

Print on the console the timestamps when the signal was not respecting the monitoring condition.

.---

2.11 Program Check

Receive from the command line the following commands:

--check - this will trigger the program check operation

--hints - this will open the bowser installation download links if the tool is not installed.

Check if a list o programs is installed with the correct version on the PC. The programs can be installed in any folder.

Programs to check:

- Beyond Compare
- Total Commander
- Notepad++
- Git
- Git Extensions

Print on the console with color the status. Each program if it is installed, where is it installed and the version. For each program not installed open a browser with the link where to download it.

.---

2.12 Compare

From the command line receive two folders.

Create a comparison script. It will check if the two folders and everything inside are the same.

If they are not the same print on the console the line ,the file path and the difference.

.---

2.13 Scheduler

As input receive a XML file containing the path of the executable to start and the time of time to delay the start.

Create a python script that will run on the PC startup and start all programs that are received from the XML file. Also each program can have a delayed start.

Create a log file that logs every start operation taken and fails if there are any.

.---

2.14 Server

Receive as input a XML file containing programs executable path and their name.

Create simple SERVER CLIENT set of scripts that can allow someone to connect to the server and start a program (from the received XML file) on the server machine. The client server communication will be done using sockets.

The server and client communication will be encrypted (so a third party will not be able to listen.)

.---

2.15 Search

Receive from the command line the following commands:

 - --path - it will contain a folder path for searching in

--shearch - this is the text to search for. The text can contain regular expressions

--files - regular expression or file names where to search. If this is not given search in all files.

Search in all files, sub files and subfolders in the folder given for a certain text. The script must search VERY FAST for the text.

Print on the console the path of all the files in which the text has been found.

.--

2.16 Unused Bits

Receive a Canoe DBC file. Also receive from the command line the commands:

--dbc = the dbc file to check

-v - if this command is given print the results to the screen.

Search in the DBC file for empty frame spaces. If a certain set of bits are not used in a frame log it in a txt file and print it on the console.

A txt file with all empty spaces in frames and also print on the console the results.

.--

2.17 Dropbox

Receive from the command line a number of folders:

--sync=path1,path2,path3

Receive a list of folder paths from the command line

--start = start the sync

--stop = stop the sync

--v = allow printing progress on the screen

Create python script that when running syncs all the files from all the folders give.

The script must always monitor and resync the folders.

The script will automatically start the sync but it can be stopped or restarted from the command line.

The sync operation and the user command must work at the same time(Use Threads.)

2.18 Text Editor

Create a simple text editor that can load a text file (.txt or any other format) .

The user can edit the text and change the general properties of its font(bold, italic, underline, size).

After editing the user will be able to save the changes in the file.

GUI:

- the application window can be resizable
- a textbox which will allow the displaying and editing of the text.
- a toolbar or menubar that will allow the user to press certain functional actions or buttons

----

2.19 Temperature Converter

Create a small program that is able to convert a value from a temperature unit to another with the push of a button.

The user can select to convert to either Celsius, Kelvin or Fahrenheit from a combo box.

After conversion, the operation will be logged in a small textbox (ex.">Converted 23° Celsius to 73.4° Fahrenheit")

GUI:

- the application window can be resizable
- the interface will have two combo boxes in which the user selects the temperature units
- the conversion will occur when a button is pressed and printed in the textbox log

----

2.20 Temperature Converter

Create a small program that is able to convert a value from a temperature unit to another with the push of a button.

The user can select to convert to Celsius, Kelvin or Fahrenheit from a combo box.

After conversion, the operation will be logged in a small textbox (ex.">Converted 23° Celsius to 73.4° Fahrenheit")

GUI:

- the application window can be resizable
- the interface will have two combo boxes in which the user selects the temperature units
- the conversion will occur when a button is pressed and printed in the textbox log

2.21 List Maker

Create an application that can compile a list of elements in a tree widget.

The user adds items in the list by pressing a button which will display an input dialog popup.

After the list is completed, the user can generate a .txt file with the items in the list, by pressing a button.

GUI:

- The popup used for adding can be a premade one(see hints) or made manually.
- a tree or list widget to hold the items you will add.
- Upon pressing the button to generate a text file containing all list items.

2.22 Car Configurator

Create a small application that simulates the configurator of a new car you are buying.

The user can select from several checkboxes the features they want in their car(min. 5).

The car will have a starting price of 5000$ and based on the features selected, the user will be informed of the final price of the car.

GUI:

- the application window will display a picture of the car(see hints)
- there will be a 'calculate' button to display the calculated price of the car, based on the selected features
- the price will be displayed either in a label or a textbox

2.23 Triangle

Create a small application that shows a triangle and allows the user to input its side lengths.

Based on the side lenghts inputed, the program must display the type of triangle it is.

A textbox will log all the inputed values inside.

GUI:

- the application window will display a picture of a symbolical triangle(see hints)
- there will be three input boxes for each triangle side
- upon pressing a button, a text status will be displayed, showing the type of triangle the user has described and
- the values will be logged.

2.24 Text Searcher

Create a simple text viewer that can load a text file (.txt or any other format) .

The user can view the text in a textbox and search in it using a text input box.

After pressing a search button, the textbox will go to the matching item.

GUI:

- the application window can be resizable
- a textbox which will allow the displaying of the text.
- an input box and button to search for inputed string

2.25 Mass Converter

Create a program that is able to convert a value from a mass unit to another with the push of a button.

The user can convert from metric to imperial units and select from a combobox the subunit they want.

the units will be metric: tone, kg, g and imperial: stone, pound, ounce

After convertion, the operation will be logged in a small textbox (ex.">Converted 1kg to 2.2 lbs")

GUI:

- the application window can be resizable
- the interface will have two comboboxes in which the user selects the mass units
- the convertion will occur when a button is pressed

2.26 List Selector

Create an application that has a list of car models loaded(min. 3).

Upon selecting a car from the list, the program will display information about the car in a textbox.

The text will be formated as follows: it will have a green background, text point size of 12 and bold

GUI:

- the application window can be resizable
- a list or tree widget to display the car models
- a textbox which will allow the displaying of information text.

2.27 Form Tree

Create an application with a 3-column tree that holds a list of employees.

The user can add employees using 3 text fields signifying: name, position, phone number.

After pressing an add button, the employee will be added to the tree.

GUI:

- the tree widget must have 3 columns and headers(name, position, phone number)
- text boxes to input the data as strings.
- a button, that when pressed, will add the 3 inputs to the tree. Each input in a separate column of the same row

2.28 List Keeper

Create an application that can load a text file (with several items, each on a separate line).

The application parses the items in the file and populates a list or tree widget with them.

After the list widget is populated, the user can add new items by pressing a button. The new items are also added to the text file.

GUI:

- the list of items should be populated when the application starts.
- the user can create a new item in a small textbox and insert it in the list widget by pressing an add button.
- when pressing the add button, the item is also added in the text file.

2.29 Calendar Tasks

Create an application that lets the user input a task and select a particular date for it.

The user can select the date for the task using a calendar or a date widget with a calendar pop-up.

After pressing a button, the task allong with the date for it are inserted into a big textbox.

The user can export the data inside the big textbox into a text file by pressing a button.

GUI:

- a new task is added in a small textbox and the date for it is selected from a calendar
- the interface will have two buttons: one for adding a task and one for exporting the added tasks
- a new task is always appended at the end of the big textbox. This textbox will hold all inputed tasks

2.30 Toolbar Control

Create a small application that has a list or tree widget with elements.

The user can add, remove, rename and duplicate an element from the list.

The program will have a toolbar with a button or action for each operation previously mentioned.

GUI:

- to add an element in the tree or list, you should use a small popup.
- the popup used for adding can be a premade one(see hints) or made manually.
- the list or tree widget will have a yellow background

3.Interesting

3.1 Python Comment Search

Create a Python function that receives a path as parameter gets from that path all python scripts.

It will check in each python scripts for functions. If a function does not have a comment (immediately after the function declaration) store the function name and python script path. At the end of the run the function should return a dictionary of all python scripts and functions with no comment.

3.2 Resource Monitor

Create a python script that monitors the computer every 100 ms and checks if the CPU or the memory of a process exceeds a certain value .If the values are over a low limit print a warning, if they exceed a high limit shut the process down.

3.1 Port Monitor

Create a python script that checks every 100 ms what communication ports are used and prints a status.

3.1 Game of Life

Create a python implementation of the Game of Life by Professor Conway.

See https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
