

# Progetto M122

---

MODULO 122

Ramphy Aquino Nova  
CPT LOCARNO

## Contents

Presentazione .....	2
Da installare .....	2
Classi .....	3
Disassembler .....	3
Hexdump .....	3
Gethexdump .....	3
Getsections .....	3
Disassembler .....	3
Getstrings .....	3
Searchstring .....	3
Terminal .....	4
Bannerhelp .....	4
Clear .....	4
Graphic .....	4
Setaddress .....	4
Banner .....	4
Run .....	4
MainTui .....	5
Create .....	5
Addcontent .....	5
Setcontent .....	5
Deletecontent .....	5
Main .....	5
Onstart .....	5
CustomColor .....	6
__str__ .....	6
__add__ .....	6

## Presentazione

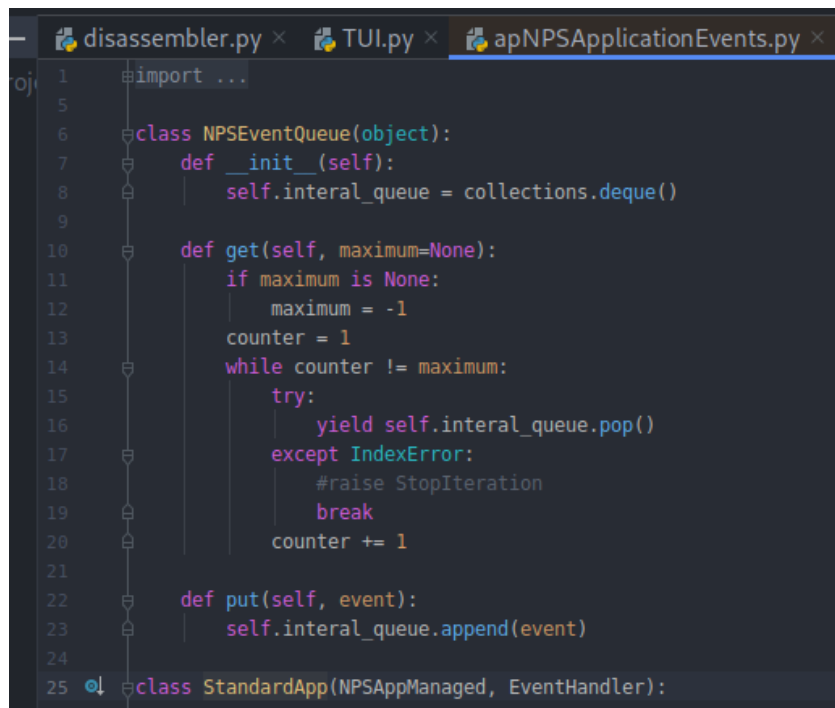
Il progetto consisteva nel scegliere un gioco oppure un programma da sviluppare in Python. Per interesse personale ho deciso di creare un decompilatore prendendo come spunto altri da me conosciuti come Radare2 e GDB di Linux.

## Da installare

Per il buon funzionamento del progetto i seguenti moduli devono prima essere installati:

- npyscreen
- capstone
- binascii

**Nota:** La classe Main nel file TUI.py eredita da StandardApp, dentro il file dove si trova StandardApp c'è una classe NPSEventQueue, quella classe con alcune versioni di Python 3.x crea problemi con la Keyword "StopIteration" per evitare il problema ho cambiato la stringa "raise StopIteration" con un "break"



```
1  import ...
5
6  class NPSEventQueue(object):
7      def __init__(self):
8          self.internal_queue = collections.deque()
9
10     def get(self, maximum=None):
11         if maximum is None:
12             maximum = -1
13         counter = 1
14         while counter != maximum:
15             try:
16                 yield self.internal_queue.pop()
17             except IndexError:
18                 #raise StopIteration
19                 break
20             counter += 1
21
22     def put(self, event):
23         self.internal_queue.append(event)
24
25 class StandardApp(NPSAppManaged, EventHandler):
26     #MAINQUEUE TYPE = NPSEventQueue
```

## Classi

L'intero programma è composto da 5 classi, 1 per tutto l'aspetto logico e tutti i tool che offre il programma per analizzare un certo eseguibile o file binario e il resto per tutto quello che riguarda la parte grafica.

### Disassembler

La classe Disassembler contiene tutte le funzioni necessarie per il buon funzionamento della parte logica.

#### Hexdump

La funzione hexdump() serve per ricavare dati in esadecimale dal file dato come input nel costruttore. I dati vengono ordinati in base ad un certo indirizzo di memoria e per ogni riga esadecimale vengono ricavati i valori ascii.

#### Gethexdump

La funzione gethexdump() serve per dare una formattazione ai dati della funzione hexdump()

#### Getsections

La funzione getsections() serve per ricavare gli indirizzi e nomi delle varie sezioni. Per questa funzione ho usato il modulo elftools.

#### Disassembler

La funzione disassembler serve per ricavare dal file dato come input nel costruttore il codice disassembler. Per questa funzione ho fatto uso del modulo Capstone.

#### Getstrings

La funzione getStrings() serve per ricavare tutte le stringhe in chiaro dentro il file.

#### Searchstring

La funzione searchString() serve per cercare una stringa specifica o substringa dentro i risultati della funzione getStrings().

## Terminal

La classe Terminal si occupa di interfacciare tutte le funzioni della classe Disassembler con l'utente tramite terminale. La classe si occupa anche dei riferimenti tra i comandi esistenti e le funzioni da eseguire in caso d'uso di tali comandi.

Alcune funzioni non le citerò perché vengono usate per la stampa dei risultati delle funzioni della classe Disassembler in modo formattato.

### Bannerhelp

La funzione `bannerHelp()` si occupa di stampare un piccolo manuale sul come usare la terminale del programma.

### Clear

La funzione `clear()` serve per pulire la terminale, come il `clear` o il `cls` di Windows.

### Graphic

La funzione `graphic()` serve per attivare la modalità grafica del programma, per farlo utilizza l'oggetto `tui` che viene passato alla classe tramite il costruttore.

### Setaddress

La funzione `setAddress` serve per settare una variabile "globale" con l'indirizzo di memoria specificato con lo scopo di stampare i dati partendo da tale indirizzo.

### Banner

La funzione `banner()` serve per stampare un piccolo banner con il nome del programma e il nome del creatore.

### Run

La funzione `run()` viene usata per gestire le altre funzioni della classe, è il "main" della classe Terminal.

## MainTui

La classe MainTui serve per creare le sezioni per la TUI. Questa classe eredita da npyscreen alcuni elementi.

### Create

La funzione create() serve per istanziare le sezioni principali della tui, specificando colore del testo, posizione e grandezza.

### Addcontent

La funzione addContent() serve per aggiungere contenuto ad una sezione specifica, riceve come parametro il nome della sezione ed un valore

### Setcontent

La funzione setContent() serve per settare un determinato contenuto ad una sezione. Questa funzione però sostituisce il contenuto già presente.

### Deletecontent

La funzione deleteContent serve per rimuovere tutto il contenuto dentro una specifica sezione.

## Main

La classe Main serve per inizializzare la schermata principale che contiene tutte le sezioni.

### Onstart

La funzione onStart serve per aggiungere nella schermata principale le sezioni create e per inserire del contenuto nelle sezioni.

## CustomColor

La classe CustomColor serve per dare colore alle stringhe.

### `__str__`

Ho sovrascritto la funzione `__str__()` in modo da usare direttamente l'oggetto che ritorna la classe.

### `__add__`

Ho dovuto creare la funzione `__add__()` in modo da poter concatenare altre stringhe con le stringhe che genera la classe.

## Commenti

Sinceramente mi sono divertito e ho imparato nuove cose con questo progetto nonostante i tempi ridotti per farlo.

Per questo progetto sono riuscito a raggiungere tutti i miei obbiettivi tranne per la parte grafica visto che ho avuto un po' di problemi con Npyscreen però sono soddisfatto del risultato finale.