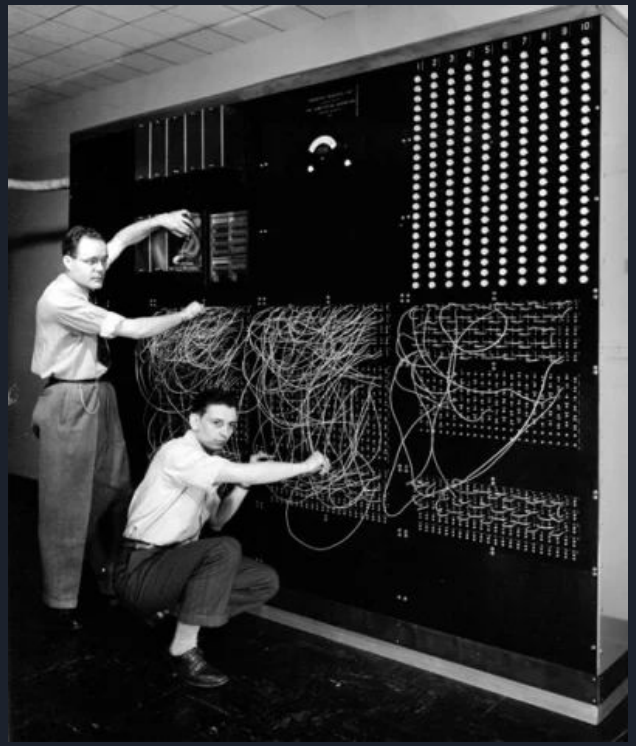
Debugging
(o depuración)

TB022 - Esteban - Riesgo - Kristal

Qué es Debuggear?

Acción de analizar un sistema en búsqueda de *bugs*

La palabra *bug* para definir un error o problema en un sistema informático surge del año *1947* cuando los ingenieros que estaban trabajando en la calculadora Mark II Aiken Relay se encontraron con que el sistema no estaba funcionando correctamente porque había una polilla dentro del hardware.



Debugging

[di:ˈbʌgɪŋ]

1. Being the detective in a crime movie where you are also the murderer

Debugger

El debuggear un programa puntual puede hacerse de muchas maneras distintas:

- Agregar sentencias adicionales al mismo, como pueden ser prints, para obtener información de variables, flujos tomados, etc.
- Guardar el input y output en lugares determinados para analizar luego

Pero la manera más eficiente suele ser usando un *Debugger*.

Un debugger es un programa que sirve como wrapper para correr el programa que presenta bugs que queremos solucionar.

GDB: The GNU Project Debugger (link)

El programa GDB le permite al usuario ver qué está ocurriendo “dentro” de otro programa mientras el mismo está siendo ejecutado.

GDB permite hacer cuatro operaciones básicas (y otras más en base a éstas) para poder detectar bugs mientras ocurren:

- Iniciar un programa especificando su input o cualquier cosa que pueda afectar su comportamiento
- Hacer que el programa *se detenga* en cualquier momento en base a ciertas condiciones dadas
- Examinar el estado del programa: ver qué se ejecutó, qué efectos secundarios generó, el estado de la memoria y más cuando el programa está detenido
- Cambiar cosas en el programa para poder experimentar alternativas y efectos y solucionar el bug presente (y repetir el proceso con el siguiente bug que se encuentre)

PDB: The Python Debugger (link)

El módulo *pdb* define un debugger de código fuente interactivo para programas Python. Soporta setear *breakpoints* condicionales, single steps para avanzar línea por línea del código fuente, inspección de stack traces, listado de código fuente y evaluación de código Python en el contexto de cualquier stack frame, entre otras cosas más.

El debugger es extensible. Cualquiera puede ampliarlo y agregarle funcionalidad según sus necesidades.

Modo de uso

Generalmente la forma de operar con un debugger es siempre igual:

- Setear uno o más *breakpoints* en el código para analizar el estado del programa en momentos clave donde creemos que podemos encontrar el *bug*
- Iniciar el programa ejecutándose normalmente hasta el primer breakpoint
- Analizar el estado del programa en el breakpoint (viendo los valores en memoria de las distintas variables).
- Identificar si todo está en el estado correspondiente (las variables tienen el valor que deberían tener)
- Avanzar línea por línea o hasta el siguiente *breakpoint* repitiendo los pasos anteriores hasta encontrar el bug

PDB Modo de uso: código fuente

En la línea que se desee insertar un *breakpoint* simplemente agregar la sentencia

```
breakpoint()
```

Desde la terminal ejecutar pdb y el módulo a debuggear

```
python -m pdb <modulo>.py
```

Al finalizar, borrar los `breakpoint()` del código fuente

Operaciones de PDB

`next (n)`
Avanzar a la siguiente línea

`continue/cont (c)`
Avanzar hasta el siguiente breakpoint

`step (s)/down (d)`
Entra a la función

`up (u)`
Sale de la función y vuelve al surrounding scope

`where (w)`
Imprime el stack trace actual

`quit (q)/exit (e)`
Termina el programa (ejecutar una segunda vez para salir de pdb

`help (h)`
Ayuda

Ejecución desde un IDE

Visual Studio Code

Por única vez: `Ctrl+Shift+D -> Add Configuration -> Python File`

Cada vez: `F5` (o click en la flecha verde) iniciará el debugger con el archivo actual (si se tienen muchos módulos en el proyecto se puede cambiar la configuración para que corra desde otro archivo, el main por ejemplo).

Pycharm

Link a la documentacion

Ejercicio: Tateti

Nos compartieron el código de un TaTeTi que ya está implementado. Parece que hay algo que no está andando bien, pero no nos supieron decir qué.

Nos piden que revisemos el código para ver qué anda mal.

Podemos debuggear el código para ver cuál es el problema.

Fin