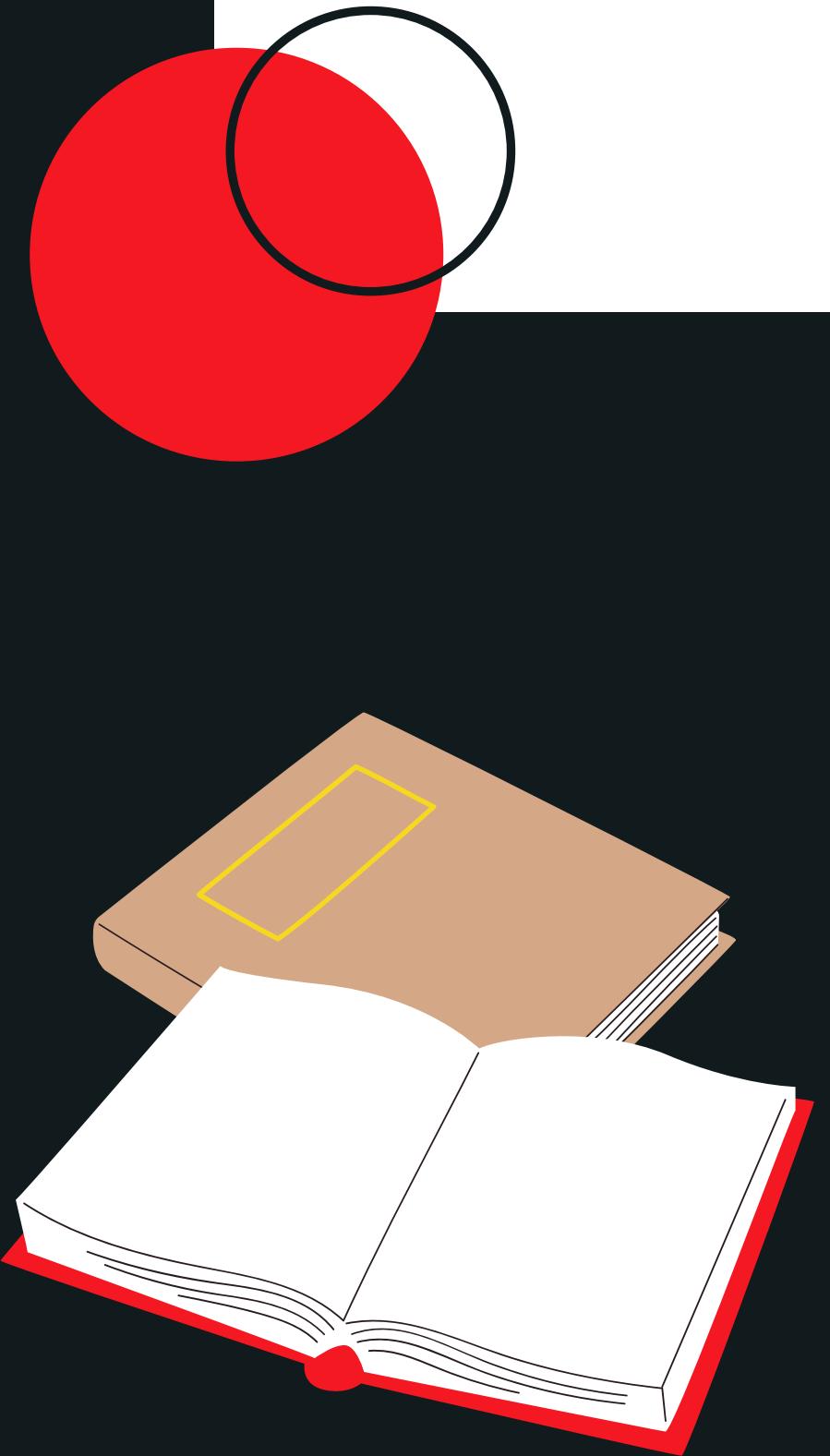
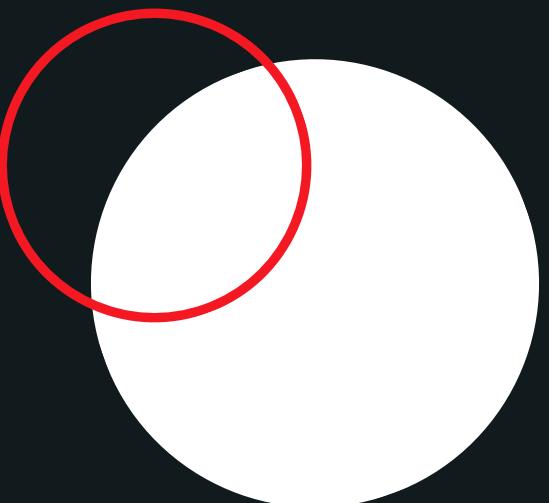


# Evaluando la complejidad de una tarea en Python, con Python

Gonzalo Fernández -- Desarrollador Full-Stack



# AGENDA

- 
1. Tareas de Programación
  2. Qué entendemos por "complejidad"
  3. Midiendo la complejidad de una tarea
  4. Implementación



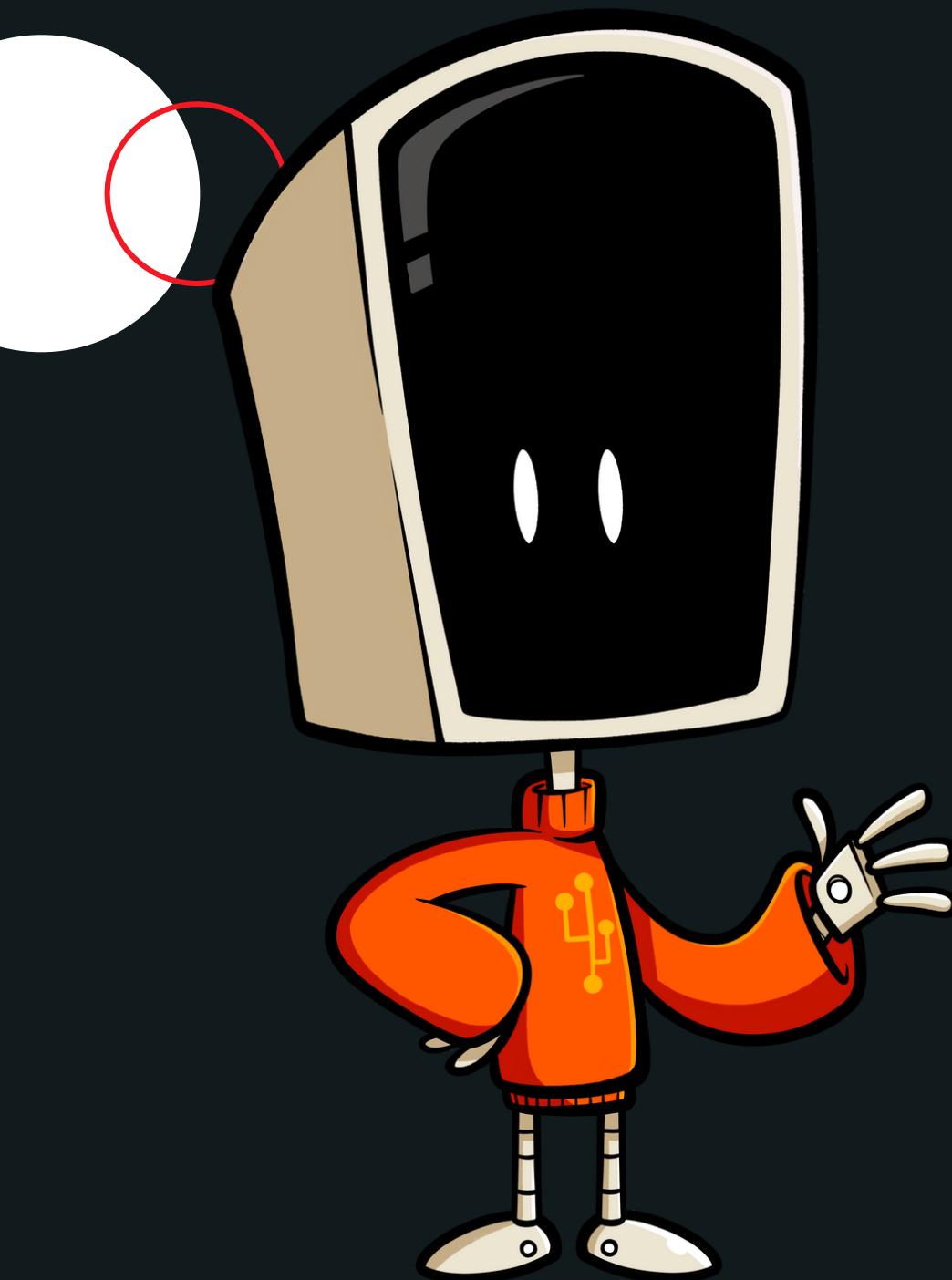
# GONZALO FERNÁNDEZ

## DESARROLLADOR FULL-STACK

Aspirante a trompetista de jazz, y futuro ingeniero civil en informática de la UTFSM

Amante del desarrollo de software y su unión con la docencia. Todo el aporte que se pueda realizar con software, genera un interés en mí para investigar

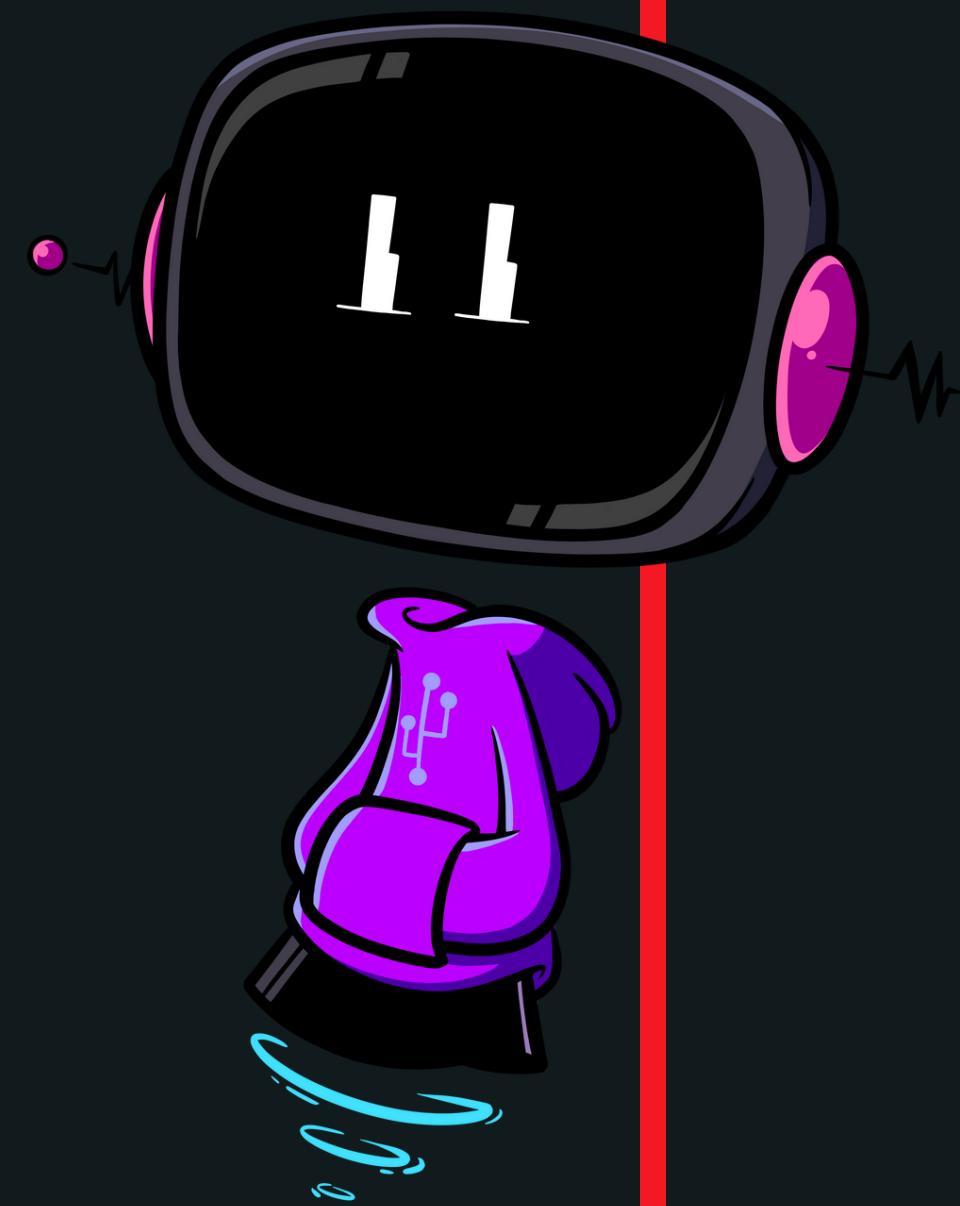
# TALLERES ÓCI>LABS.



Talleres orientados a la enseñanza de la programación para estudiantes desde 7mo Básico a 4to Medio.

Existen 3 Niveles:

- **Básico** – introducción al pensamiento computacional a través de Scratch.
- **Intermedio** – introducción a la programación a través de C++.
- **Avanzado** – conceptos avanzados de programación y competencia OCI.



# TALLERES >LABS.



Profesor Federico  
Meza



Profesora Andrea  
Vásquez



Profesor Pedro  
Godoy

# TALLERES ÓC>LABS.



Anastasiia Fedorova



Mª Paz Morales



Hugo Leyton



Gabriel Carmona

# TALLERES ÓCI>LABS.

Veamos algunas cifras de los talleres desde 2019

**3.292**

ESTUDIANTES HAN  
PASADO POR  
NUESTROS TALLERES

**87**

TUTORES Y TUTORAS  
NOS HAN APOYADO

**13**

ESTUDIANTES HAN  
CLASIFICADO A LA  
COMPETENCIA DE LA OCI

**4**

AÑOS APOYANDO EN  
LA ENSEÑANZA DE LA  
PROGRAMACIÓN

# Programación IWI-131

---



¡Y muchos más!

# Clases de Programación

Veamos que ocurre en 1 semana de clases,  
equivalente a 1 UVA (Unidad Virtual de Aprendizaje)



**TRABAJO  
PREVIO**

(En Aula) Cápsulas de video y Evaluación Formativa

**SESIÓN 1**

**70 m** Aclaración de Dudas y Ejercicios Habilitadores:  

- Parsons
- Completar Código
- Corregir Código
- Ruteo

Control y Ejercicios Introductorios

**SESIÓN 2**

**70 m** Ejercicios Supervisados y Andamiaje.

**AYUDANTÍA**

**70 m** Apoyo a la Tarea y Ejercicios

Tarea Semanal (90 m.)

Ejercicio de Corrección Automática (60 m.)

Consulta a Profesores en horario de atención - Consultas y revisión de Foro Piazza



# TAREAS DE PROGRAMACIÓN

¿QUÉ SON Y POR QUÉ LAS USAMOS?



Los derechos de autor de las obras musicales se cobran a quienes hacen uso lucrativo de estas creaciones. Por ejemplo, a quienes organizan conciertos, a las emisoras de radio, etc. Los dineros recolectados son distribuidos a los compositores y las compositoras de las obras que fueron utilizadas, de acuerdo con un criterio de proporcionalidad. Primero, se obtiene un valor por obra, dividiendo el monto recaudado en una utilización particular (por ejemplo, un concierto) entre la cantidad de obras que fueron interpretadas. Después, el monto asignado a cada obra se reparte equitativamente entre las compositoras y los compositores de la obra. Para apoyar el proceso de distribución de los dineros, se cuenta con varios archivos, cuyo formato se describe a continuación.

El archivo de compositores contiene compositores(as), junto con un número único que les identifica, denominado a nivel internacional como número IPI (*Interested Party Information*). Los campos se separan por un punto y coma. Un extracto de este archivo se muestra a continuación:

compositores.csv

```
00234720294;GROHL DAVID
00403809281;SHIFLETT CHRISTOPHER A
00181675549;MENDEL NATE
00344920759;HAWKINS OLIVER TAYLOR
00226132995;GONZALEZ RIOS JORGE
00245234584;LINDL ROMERO ROBERTO
00244594943;HENRIQUEZ PETTINELLI ALVARO
00201037340;MORISSETTE ALANIS
...
```

En un curso introductorio de programación, las tareas son una herramienta que cumple múltiples propósitos de cara al proceso educativo.

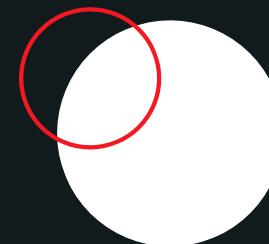
”

**Las tareas son parte  
importante del proceso de  
enseñanza y aprendizaje**

ALLISON BOYE

PH.D. TEACHING, LEARNING, AND PROFESSIONAL DEVELOPMENT CENTER  
TEXAS TECH UNIVERSITY

“



# ¿Para qué sirven?



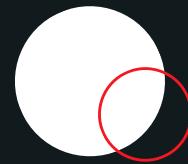
## MEDIR

Miden el aprendizaje del estudiante a lo largo del curso, de modo tal de permitir al docente tomar medidas durante el semestre lectivo



## INCENTIVAR

Brindan un acercamiento efectivo sobre como lo que el estudiante está aprendiendo, tiene un uso real



## APLICAR

Instan al estudiante a aplicar los conocimientos adquiridos a lo largo del curso, obteniendo una experiencia educativa menos expositiva y más práctica



## RETROALIMENTAR

Una vez que su tarea es evaluada, el estudiante obtiene una retroalimentación con la cual puede aprender mediante el entendimiento de sus errores

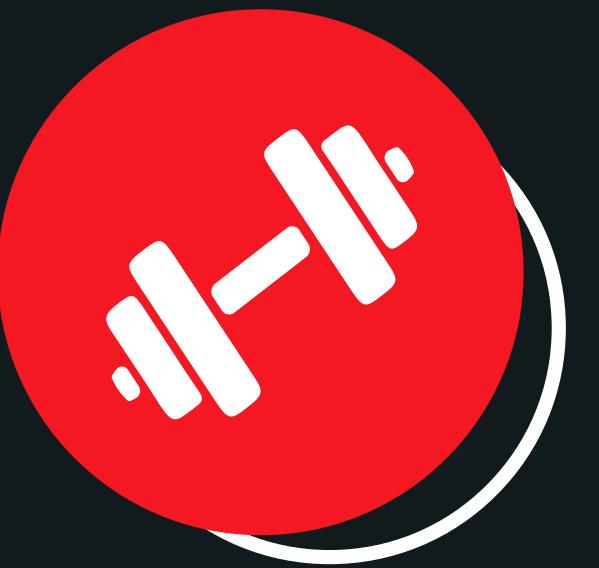
# ¿QUÉ SE DEBE CUMPLIR?



Aplicación o  
problema **real**



Ser  
**interesantes**



Nivel de dificultad  
**adequado**

# ¿Qué ocurre cuando no se cumple?



DESINTERÉS



FRUSTACIÓN

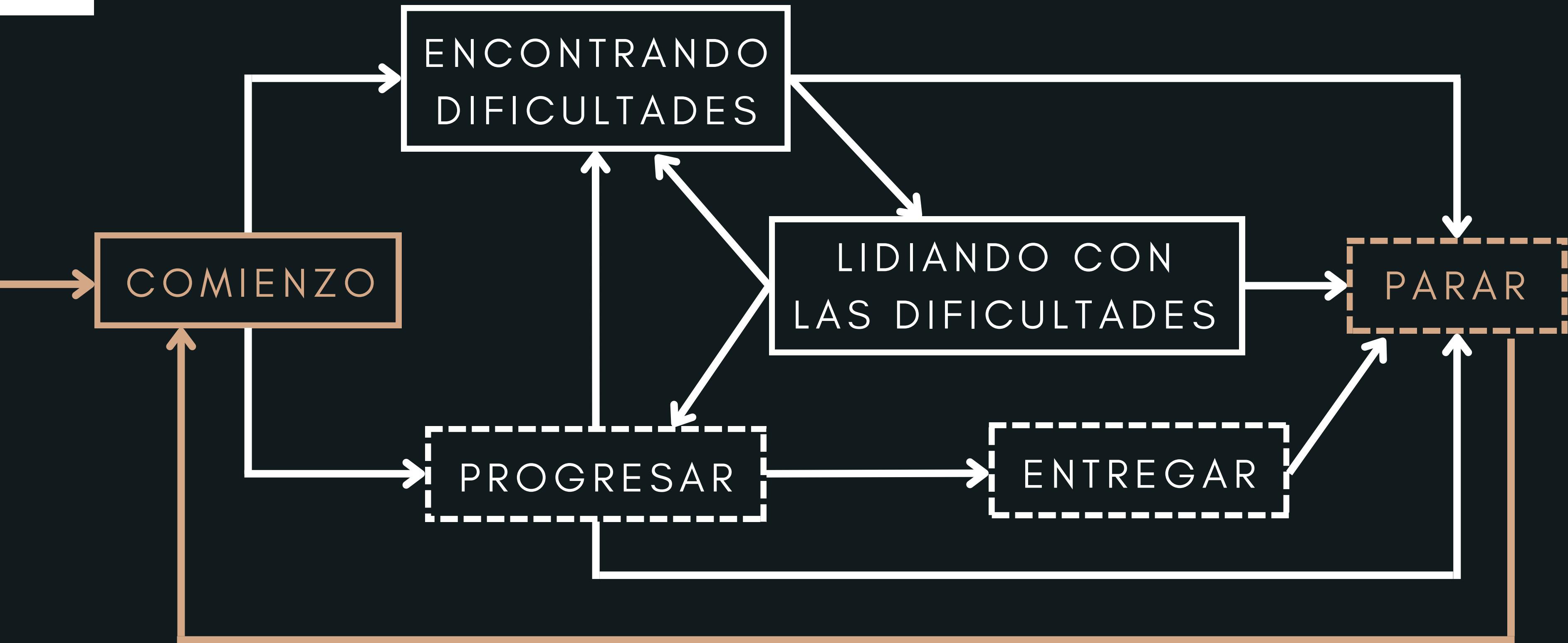


FAULTAS EN  
EVALUACIONES



COMPLICACIÓN  
EN LA RÚBRICA

# Experiencia Emocional al desarrollar una Tarea



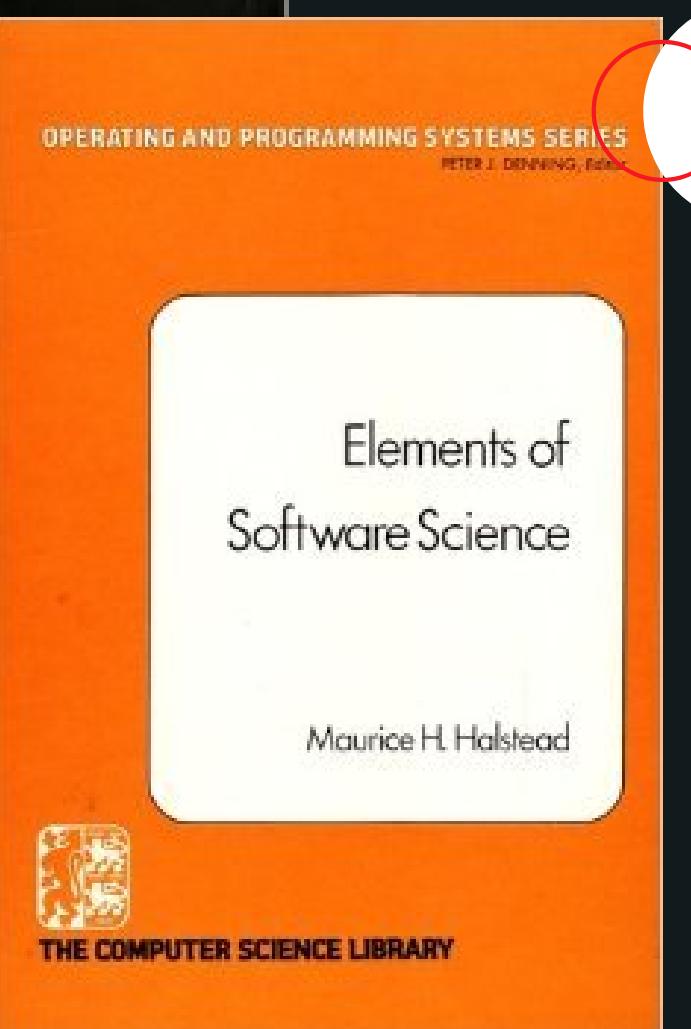


¡ES POR ESTO QUE ES IMPORTANTE  
HACER TAREAS DE CALIDAD!

# COMPLEJIDAD DE UNA TAREA

¿CÓMO PODEMOS MEDIR LA CALIDAD DE  
UNA TAREA?





# MAURICE H. HALSTEAD

## PROFESOR E INVESTIGADOR

Profesor de la Universidad de Purdue y creador de los conceptos de la ciencia del software, que condujo al desarrollo de las métricas del software.

### ELEMENTS OF SOFTWARE SCIENCE

Publicado en 1977, este libro contiene el primer resumen sistemático de una rama de la ciencia experimental y teórica que se ocupa del análisis y medición de la creación de programas computacionales.

# COMPLEJIDAD COGNITIVA DE UN CÓDIGO

---

## ASPECTOS GENERALES

- $\eta_1$  = número de operadores distintos que aparecen en la implementación
- $\eta_2$  = número de operandos distintos que aparecen en la implementación
- $N_1$  = número total de usos de todos los operadores que aparecen en la implementación
- $N_2$  = número total de usos de todos los operandos que aparecen en la implementación
- $\eta = \eta_1 + \eta_2 \rightarrow$  vocabulario de la implementación.
- $N = N_1 + N_2 \rightarrow$  largo de la implementación.

# VEAMOS UN EJEMPLO

```
def GCD(a, b):  
    if(b == 0):  
        return abs(a)  
    else:  
        return GCD(b, a % b)
```

# VEAMOS UN EJEMPLO

```
def GCD(a, b):  
    if(b == 0):  
        return abs(a)  
    else:  
        return GCD(b, a % b)
```



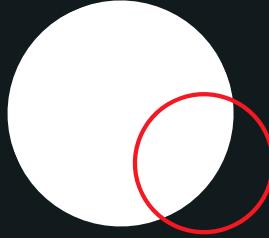
η1: 2  
η2: 3  
N1: 2  
N2: 4  
vocabulario (η): 5  
largo (N): 6

# VEAMOS UN EJEMPLO

```
def GCD(a, b):  
    if(b == 0):  
        return abs(a)  
    else:  
        return GCD(b, a % b)
```



η1: 2 = y, %  
η2: 3 a, b y 0  
N1: 2 1 uso de = + 1 uso de %  
N2: 4 2 usos de b + 1 uso de a + 1 uso de 0  
vocabulario (η): 5 η1 + η2 = 1 + 3  
largo (N): 6 N1 + N2 = 2 + 4



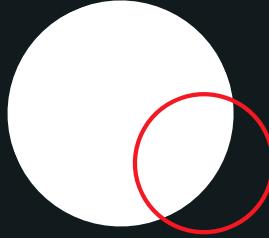
# VOLUMEN DE UN PROGRAMA

¿Cómo podemos tener una medida de tamaño para un programa, que nos dé información independiente del lenguaje?

VEAMOS UN EJEMPLO

```
a = 1 + 2
```

```
b = 2 - 1
```



# VOLUMEN DE UN PROGRAMA

¿Cómo podemos tener una medida de tamaño para un programa, que nos dé información independiente del lenguaje?

VEAMOS UN EJEMPLO

```
a = 1 + 2  
b = 2 - 1
```



η1: 2  
η2: 2  
N1: 2  
N2: 4  
vocabulario ( $\eta$ ): 4  
largo ( $N$ ): 6

# VOLUMEN DE UN PROGRAMA

¿Cómo podemos tener una medida de tamaño para un programa, que nos dé información independiente del lenguaje?

VEAMOS UN EJEMPLO

$$\begin{aligned} a &= 1 + 2 \\ b &= 2 - 1 \end{aligned}$$

$\eta_1: 2$   
 $\eta_2: 2$   
 $N_1: 2$   
 $N_2: 4$   
vocabulario ( $\eta$ ): 4  
largo ( $N$ ): 6

4 elementos únicos (2 operadores y 2 operandos) → podemos representarlos con 2 bits:

00: +  
01: -  
10: 1  
11: 2

¡Podemos calcular la cantidad de elementos diferentes de un algoritmo, sin preocuparnos del lenguaje de programación!

# VOLUMEN DE UN PROGRAMA

¿Cómo podemos tener una medida de tamaño para un programa, que nos dé información independiente del lenguaje?

VEAMOS UN EJEMPLO

$$\begin{aligned} a &= 1 + 2 \\ b &= 2 - 1 \end{aligned}$$

$\eta_1: 2$   
 $\eta_2: 2$   
 $N_1: 2$   
 $N_2: 4$   
vocabulario ( $\eta$ ): 4  
largo ( $N$ ): 6

00: +  
01: -  
10: 1  
11: 2

Para obtener el volumen  $V$  de la implementación de un algoritmo, sólo hace falta determinar cuantas veces se utilizan estos elementos:  
 $V = N \log_2(\eta)$

# Volumen en Distintos Lenguajes

```
a = 1 + 2  
b = 2 - 1
```

```
a = 1 +++++ 2  
b = 2 ----- 1
```

# Volumen en Distintos Lenguajes

```
a = 1 + 2  
b = 2 - 1
```

```
a = 1 +++++ 2  
b = 2 ----- 1
```

00: +

01: -

10: 1

11: 2

00: +++++

01: -----

10: 1

11: 2

# Volumen en Distintos Lenguajes

a = 1 + 2  
b = 2 - 1

00: + η1: 2  
01: - η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario (η): 4  
largo (N): 6

a = 1 +++++ 2  
b = 2 ----- 1

00: +++++ η1: 2  
01: ----- η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario (η): 4  
largo (N): 6

# Volumen en Distintos Lenguajes

```
a = 1 + 2  
b = 2 - 1
```

00: + η1: 2  
01: - η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario ( $\eta$ ): 4  
largo (N): 6

```
a = 1 +++++ 2  
b = 2 ----- 1
```

00: +++++ η1: 2  
01: ----- η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario ( $\eta$ ): 4  
largo (N): 6

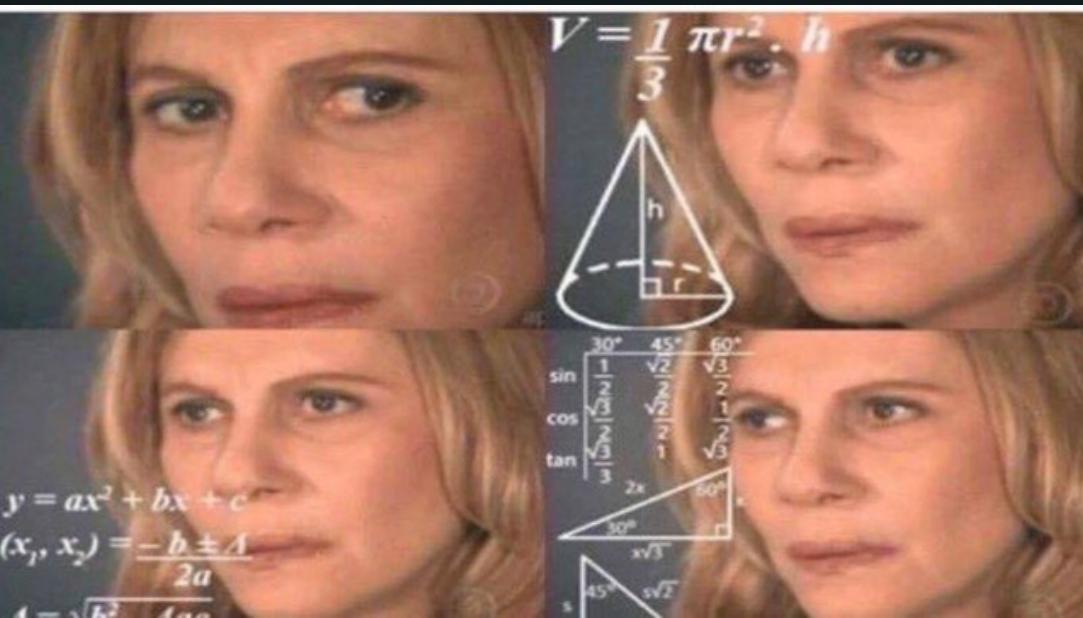
$$V = N \log_2(\eta) = 12$$



$$V = N \log_2(\eta) = 12$$

# Volumen en Distintos Lenguajes

$$\begin{aligned}a &= 1 + 2 \\b &= 2 - 1\end{aligned}$$



$$\begin{aligned}a &= 1 +++++ 2 \\b &= 2 ----- 1\end{aligned}$$

00: + η1: 2  
01: - η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario (η): 4  
largo (N): 6

00: +++++ η1: 2  
01: ----- η2: 2  
10: 1 N1: 2  
11: 2 N2: 4  
vocabulario (η): 4  
largo (N): 6

$$V = N \log_2(\eta) = 12$$



$$V = N \log_2(\eta) = 12$$

# VOLUMEN POTENCIAL

Existen lenguajes de programación que ya implementan algoritmos en sus librerías:

```
resultado = superAlgoritmo(valor1, valor2)
```

# VOLUMEN POTENCIAL

Existen lenguajes de programación que ya implementan algoritmos en sus librerías:

```
resultado = superAlgoritmo(valor1, valor2)
```

En este caso, el nombre de la función sería un operador, y el operador de asignación para utilizar su resultado sería otro:

$\eta 1^*$ : 2, donde \* significa "potencial"

# VOLUMEN POTENCIAL

Existen lenguajes de programación que ya implementan algoritmos en sus librerías:

```
resultado = superAlgoritmo(valor1, valor2)
```

En este caso, el nombre de la función sería un operador, y el operador de asignación para utilizar su resultado sería otro:

$\eta 1^*$ : 2, donde \* significa "potencial"

Además, como no necesitamos repetir ni los operadores ni los operandos, también tenemos que:

$$N1^* = \eta 1^* \text{ y } N2^* = \eta 2^*$$

# VOLUMEN POTENCIAL

Existen lenguajes de programación que ya implementan algoritmos en sus librerías:

```
resultado = superAlgoritmo(valor1, valor2)
```

En este caso, el nombre de la función sería un operador, y el operador de asignación para utilizar su resultado sería otro:

$\eta^*$ : 2, donde \* significa "potencial"

Además, como no necesitamos repetir ni los operadores ni los operandos, también tenemos que:

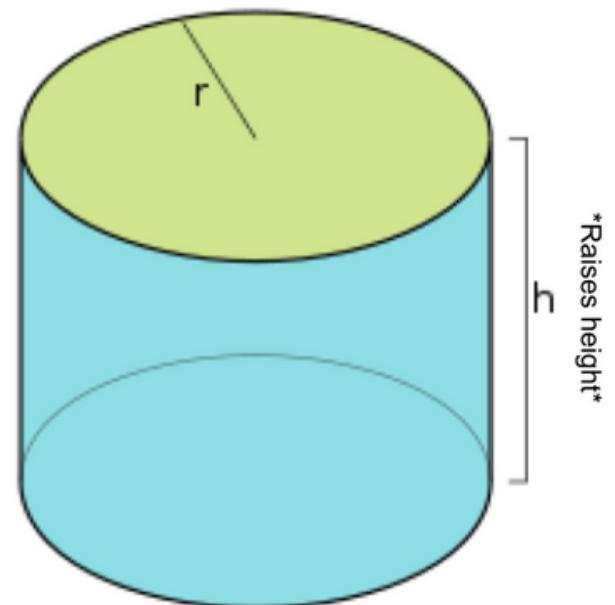
$$N^* = \eta^* \text{ y } N^* = \eta^*$$

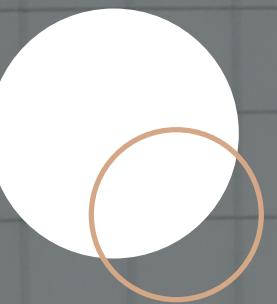
Entonces, podemos calcular el volumen potencial como:

$$V^* = N^* \log_2(\eta^*) = (\eta^* + \eta^*) \log_2 (2 \eta^* + \eta^*) = (2 + \eta^*) \log_2 (2 + \eta^*)$$

Friend: can you turn the volume up?

Me: sure





# RESUMIENDO

- $\eta_1$  = número de operadores distintos que aparecen en la implementación
- $\eta_2$  = número de operandos distintos que aparecen en la implementación
- $N_1$  = número total de usos de todos los operadores que aparecen en la implementación
- $N_2$  = número total de usos de todos los operandos que aparecen en la implementación
- El vocabulario se calcula como  $\eta = \eta_1 + \eta_2$
- El volumen se calcula como  $N \log_2(\eta)$
- El volumen potencial se calcula como  $(2 + \eta_2) \log_2 (2 + \eta_2)$

# Esfuerzo de Realizar un Programa

1

## CREACIÓN DEL CÓDIGO

Realizar **N selecciones** de un vocabulario de  $\eta$  **elementos**.

# Esfuerzo de Realizar un Programa

1

## CREACIÓN DEL CÓDIGO

Realizar **N selecciones** de un vocabulario de  $\eta$  **elementos**.

2

## BÚSQUEDA BINARIA

La acción de realizar N selecciones de un vocabulario de  $\eta$  elementos se traduce a realizar N búsquedas binarias en un conjunto ordenado de  $\eta$  elementos → cada selección realiza  **$\log_2(\eta)$  comparaciones**.

# Esfuerzo de Realizar un Programa

1

## CREACIÓN DEL CÓDIGO

Realizar **N selecciones** de un vocabulario de  $\eta$  **elementos**.

2

## BÚSQUEDA BINARIA

La acción de realizar N selecciones de un vocabulario de  $\eta$  elementos se traduce a realizar N búsquedas binarias en un conjunto ordenado de  $\eta$  elementos → cada selección realiza **log2( $\eta$ ) comparaciones**.

3

## VOLUMEN

Escribir un código equivale a realizar  $N \log_2(\eta)$  procesos cognitivos.

Ya que  $V = N \log_2(\eta)$ , el **volumen** de un programa es **equivalente a la cantidad de comparaciones mentales** requeridas para generar un programa.

# Esfuerzo de Realizar un Programa

1

## CREACIÓN DEL CÓDIGO

Realizar **N selecciones** de un vocabulario de  $\eta$  **elementos**.

2

## BÚSQUEDA BINARIA

La acción de realizar  $N$  selecciones de un vocabulario de  $\eta$  elementos se traduce a realizar  $N$  búsquedas binarias en un conjunto ordenado de  $\eta$  elementos → cada selección realiza **log<sub>2</sub>(η) comparaciones**.

3

## VOLUMEN

Escribir un código equivale a realizar  $N \log_2(\eta)$  procesos cognitivos.

Ya que  $V = N \log_2(\eta)$ , el **volumen** de un programa es **equivalente a la cantidad de comparaciones mentales** requeridas para generar un programa.

4

## DISCRIMINACIONES COGNITIVAS

El número promedio de discriminaciones que realizaríamos cognitivamente, se expresa como la razón entre el volumen de un programa y el volumen potencial:  $V^* / V$

# Esfuerzo de Realizar un Programa

1

## CREACIÓN DEL CÓDIGO

Realizar **N selecciones** de un vocabulario de  $\eta$  elementos.

2

## BÚSQUEDA BINARIA

La acción de realizar  $N$  selecciones de un vocabulario de  $\eta$  elementos se traduce a realizar  $N$  búsquedas binarias en un conjunto ordenado de  $\eta$  elementos → cada selección realiza  $\log_2(\eta)$  comparaciones.

3

## VOLUMEN

Escribir un código equivale a realizar  $N \log_2(\eta)$  procesos cognitivos. Ya que  $V = N \log_2(\eta)$ , el **volumen** de un programa es **equivalente a la cantidad de comparaciones mentales** requeridas para generar un programa.

4

## DISCRIMINACIONES COGNITIVAS

El número promedio de discriminaciones que realizaríamos cognitivamente, se expresa como la razón entre el volumen de un programa y el volumen potencial:  $V^* / V$

5

## ESFUERZO

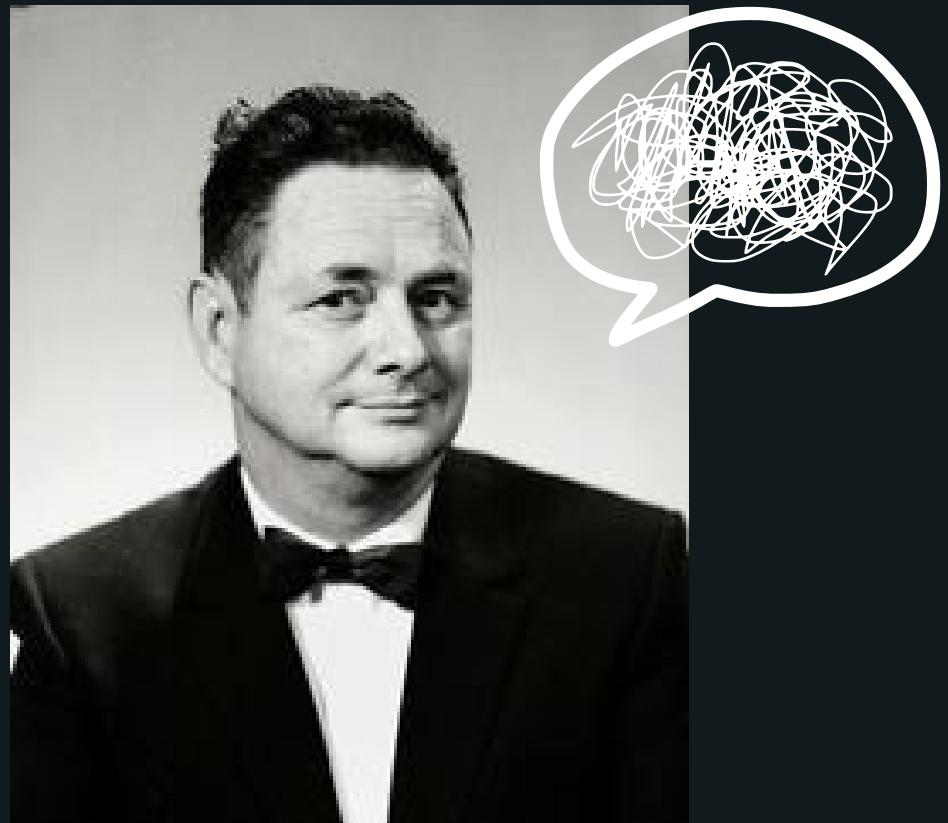
El esfuerzo mental de desarrollar un programa se calcula como la multiplicación entre la cantidad de comparaciones, y la cantidad de discriminaciones:  
 $E = V (V / V^*) = V^2 / V^*$



# IMPLEMENTACIÓN

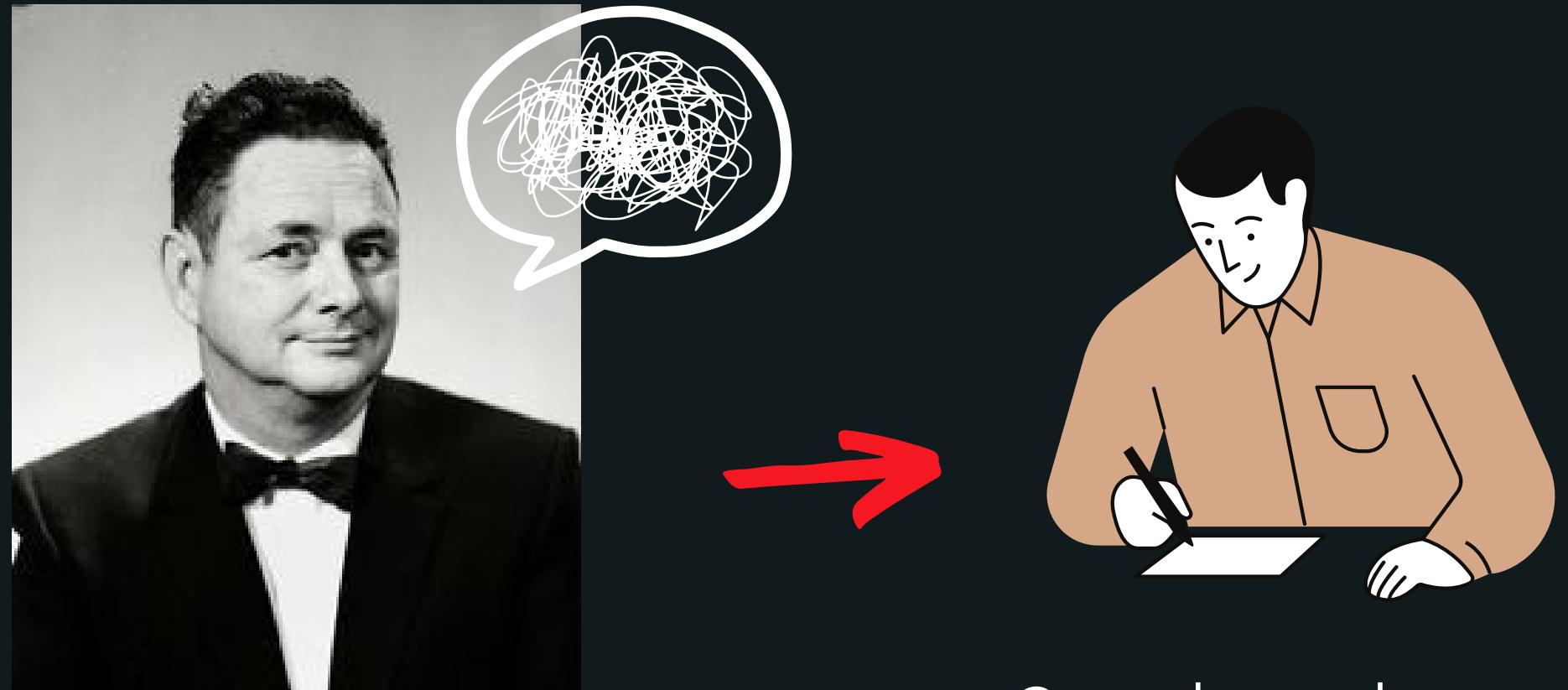
¿CÓMO APLICAMOS ESTAS MÉTRICAS?

# Aplicación de las Métricas



Maurice Halstead y sus  
métricas...

# Aplicación de las Métricas



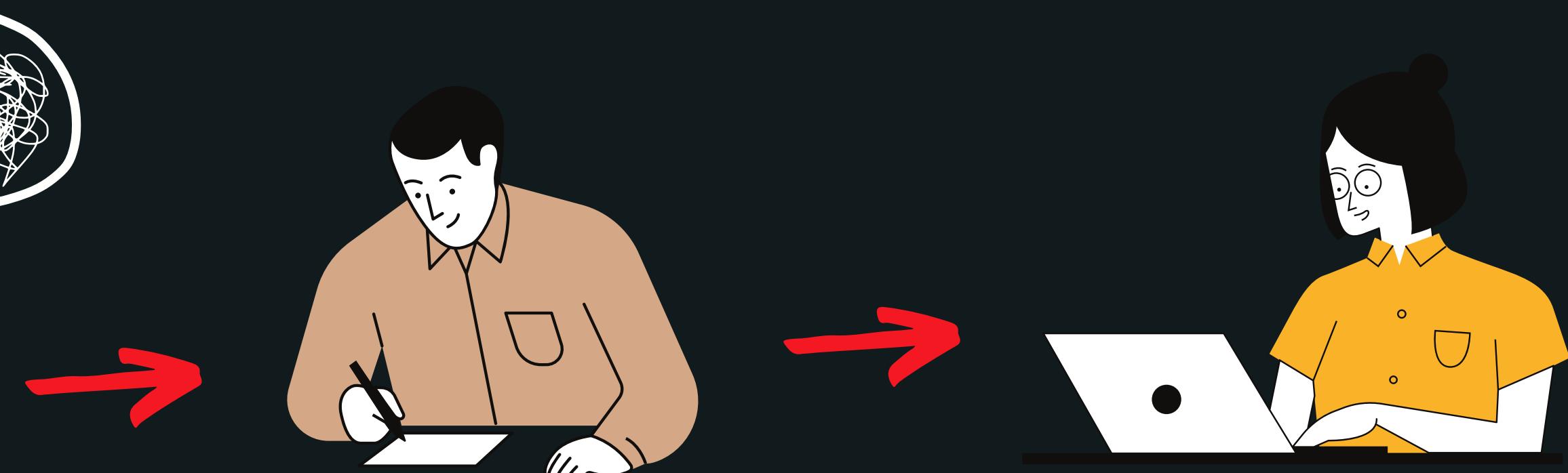
Maurice Halstead y sus  
métricas...

Cuando un docente  
redacte una tarea,  
también la debe  
desarrollar

# Aplicación de las Métricas



Maurice Halstead y sus  
métricas...



Cuando un docente  
redacte una tarea,  
también la debe  
desarrollar

Luego, debe utilizar las  
métricas para estimar el  
esfuerzo que le llevaría al  
estudiante realizarla.

# EXPERIMENTO



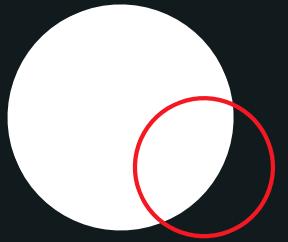
Para cada tarea de este semestre de programación, se tomó 1 entrega cuya nota sea 100, y se calcularon sus métricas de Halstead utilizando Radon.



Radon es una biblioteca de Python que trae implementado el cálculo de métricas de Halstead para un código dado, y también métricas puras, métricas de complejidad ciclomática , y métricas sobre mantenibilidad.

Radon se puede utilizar tanto pragmáticamente (es decir, siendo importado en nuestro código), como mediante su consola de comandos:

```
PS C:\Users\evdto\Desktop\Desarrollo\talks\Pycon 2022 - Evaluando la Complejidad de una tarea en Python\tasks> radon hal .\t1.py
operadores {'Mult', 'FloorDiv', 'Div', 'Mod', 'Sub', 'Add'}
operандos {(None, 'vSinImpuestos'), (None, <ast.BinOp object at 0x00000148F91C0790>), (None, <ast.BinOp object at 0x00000148F918ED90>), (None, 'presupuestoRestante'), (None, <ast.BinOp object at 0x00000148F918EE50>), (None, <ast.BinOp object at 0x00000148F91B3A90>), (None, <ast.BinOp object at 0x00000148F918EE20>), (None, <ast.BinOp object at 0x00000148F91CB9A0>), (None, <ast.BinOp object at 0x00000148F91CBA00>), (None, 'precioEcologico1'), (None, <ast.BinOp object at 0x00000148F91C0A00>), (None, 'billete1'), (None, <ast.BinOp object at 0x00000148F91BCE50>), (None, <ast.BinOp object at 0x00000148F91CB9D0>), (None, <ast.BinOp object at 0x00000148F91C0A30>), (None, <ast.BinOp object at 0x00000148F91BCE20>), (None, <ast.BinOp object at 0x00000148F91BCE80>), (None, 'precioEcologico2'), (None, <ast.BinOp object at 0x00000148F91C0AC0>), (None, 'precioTotal'), (None, 100), (None, <a, <ast.BinOp object at 0x00000148F91CBC40>), (None, 'cantUni'), (None, <ast.BinOp object at 0x00000148F91C0100>), (None, <ast.BinOp object at 0x00000148F91CBC70>), (None, <ast.Call object at 0x00000148F91C1B80>), (None, <ast.BinOp object at 0x00000148F91C0130>), (None, <ast.BinOp object at 0x00000148F91C1C10>), (None, 20), (None, 'moneda'), (None, <ast.BinOp object at 0x00000148F91C1C40>), (None, 'extraccion'), (None, 'n1'), (None, 'precioTotalp1'), (None, <ast.BinOp object at 0x00000148F91C0340>), (None, 19), (None, 'numeroInvertido'), (None, 1), (None, <ast.BinOp object at 0x00000148F91CB490>), (None, 'precio1eraNecesidad'), (None, <ast.BinOp object at 0x00000148F91CB4C0>), (None, 'n2'), (None, 10), (None, 'billete2'), (None, 'presupuestoRandom'), (None, 980), (None, <ast.BinOp object at 0x00000148F9181160>), (None, 'precio')}
.\t1.py:
h1: 6
h2: 50
N1: 53
N2: 106
vocabulary: 56
length: 159
calculated_length: 297.7025844930631
volume: 923.3694326071592
difficulty: 6.36
effort: 5872.629591381533
time: 326.2571995211963
bugs: 0.30778981086905305
```

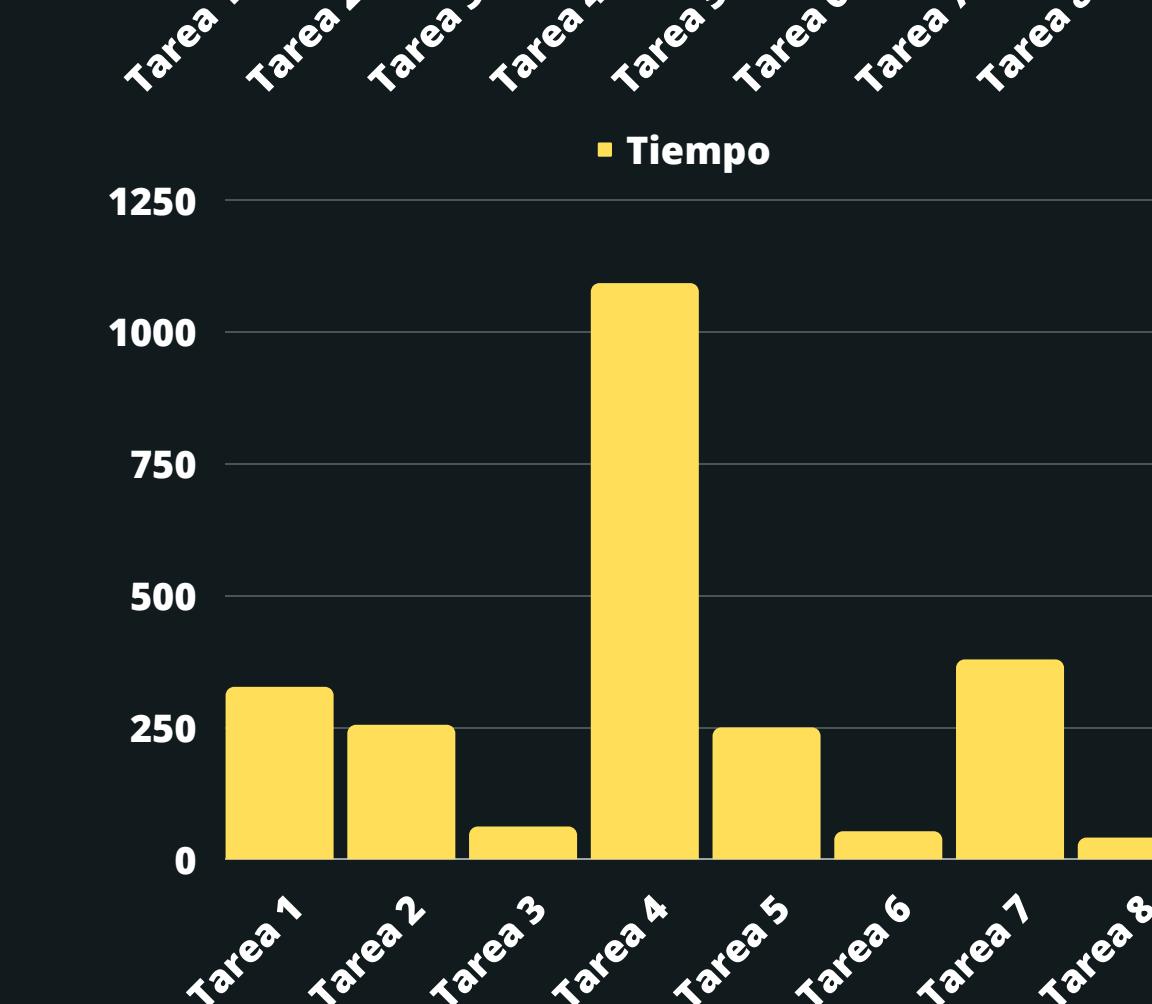
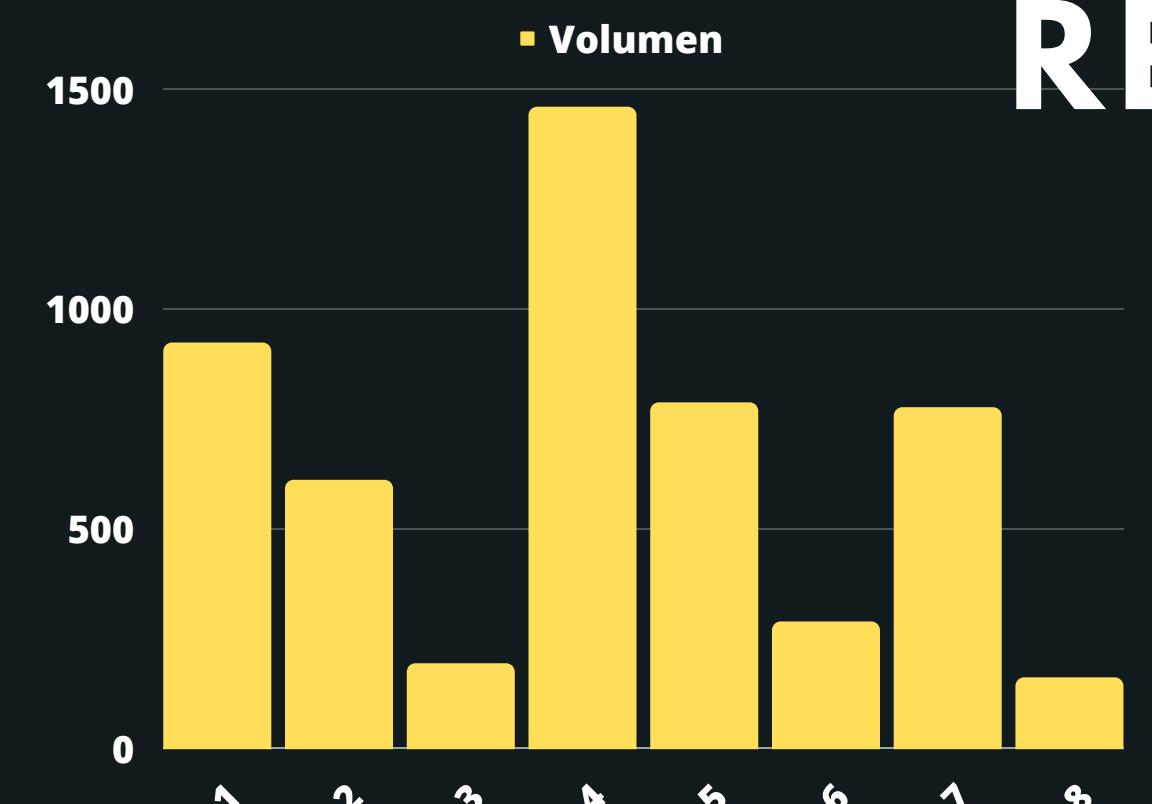
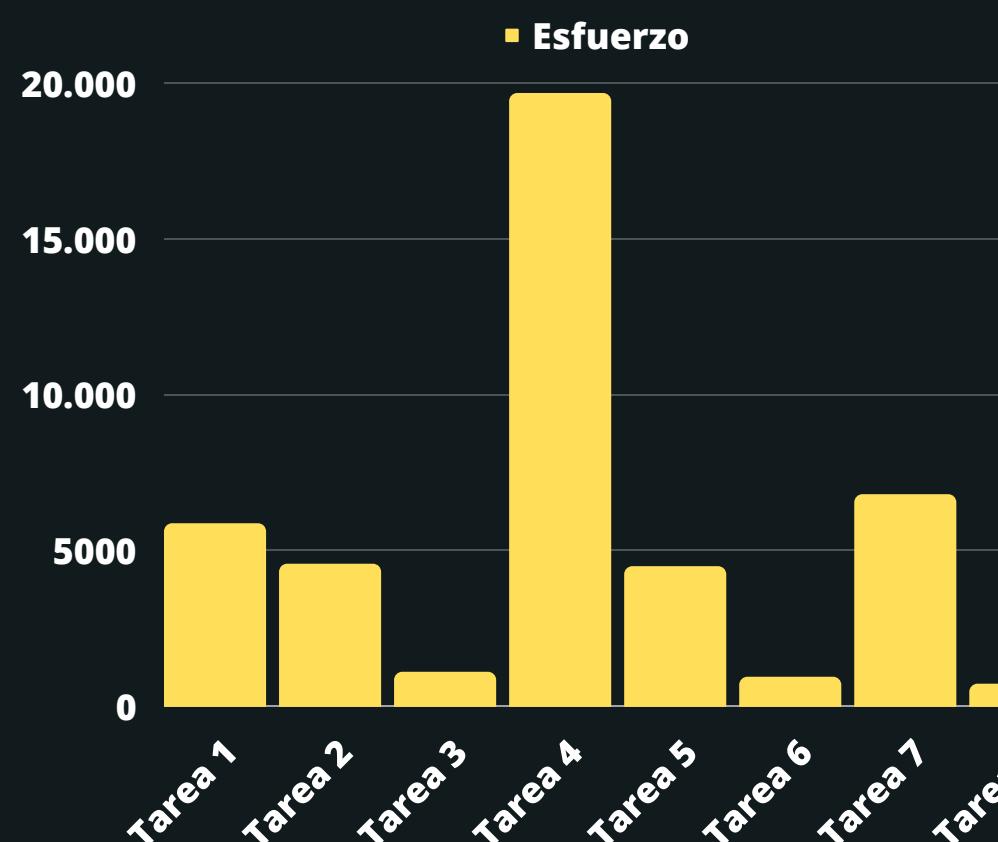
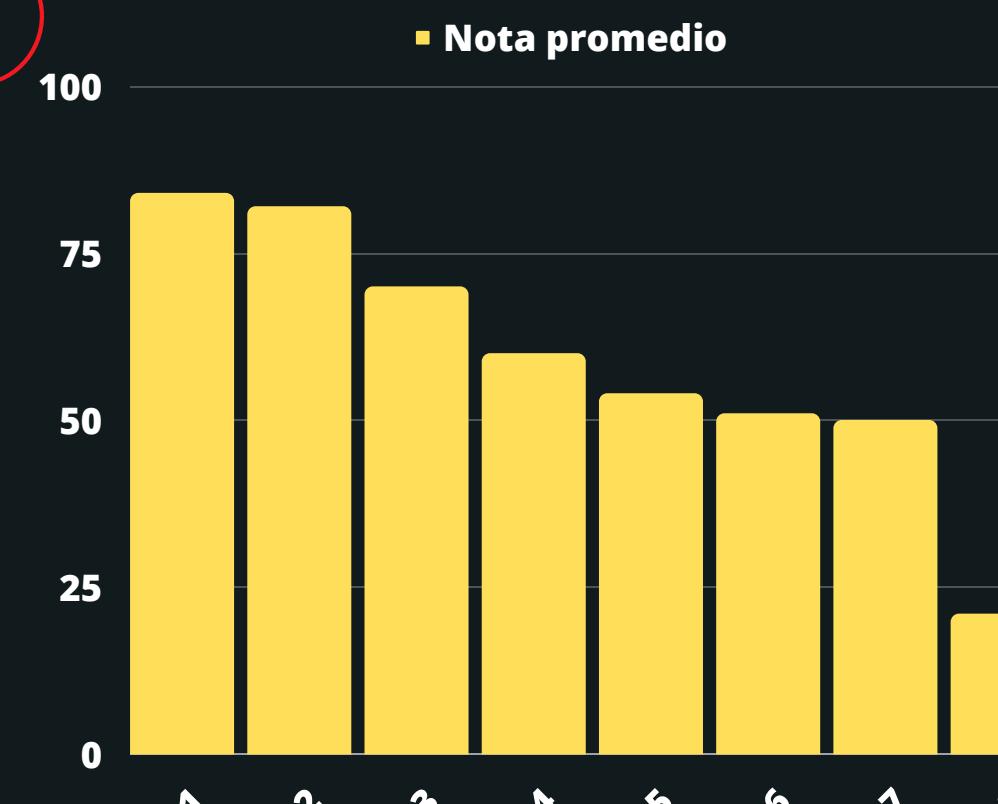


# RESULTADOS

Veamos los resultados de las métricas aplicadas en las tareas de programación de este semestre.

tarea	promedio	η1	η2	N1	N2	vocabulario	largo	volumen	esfuerzo	tiempo
t1	84	6	50	53	106	56	159	923	5872	326
t2	82	8	39	37	73	47	110	611	4574	254
t3	70	8	19	14	27	27	41	194	1108	61
t4	60	10	59	80	159	69	239	1459	19672	1092
t5	54	8	61	42	87	69	129	787	4495	249
t6	51	5	29	19	38	34	57	289	949	52
t7	50	11	54	43	86	65	129	776	6804	378
t8	21	7	18	12	23	25	35	162	726	40

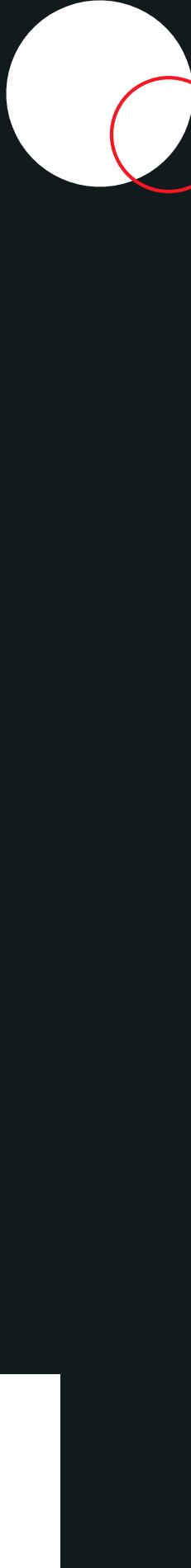
# RESULTADOS



Como consideración, el promedio de notas se ve afectado por diversos motivos, como la menor entrega de tareas a medida que avanzaba el semestre.

Para cada tarea, tanto el volumen, esfuerzo y tiempo estimado de programar se mantienen proporcionales.

Para la tarea 4, la más difícil según las métricas de Halstead, se puede suponer una desmotivación o frustración en base a enfrentarse a esa tarea.



# TRABAJO FUTURO

## ENCUESTAS A ESTUDIANTES

Se planea realizar encuestas a los estudiantes, para relacionar tanto su experiencia emocional como su percepción misma de la tarea, para analizarlas en conjunto con las métricas de Halstead.



# SOBRE EL TRABAJO DE HALSTEAD Y SUS APLICACIONES

Elements of Software Science



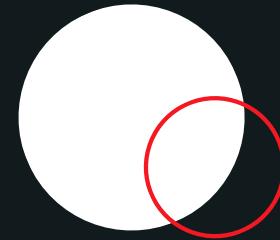
Peter G. Hamer and Gillian D. Frewin

Standard Telecommunication Laboratories Limited  
London Road, Harlow, Essex, England

## Abstract

Karl Popper has described the scientific method as "the method of bold conjectures and ingenious and severe attempts to refute them". Software Science has made "bold conjectures" in postulating specific relationships between various 'metrics' of software code and in ascribing psychological interpretations to some of these metrics.

This paper describes tests made on the validity of the relationships and interpretations which form the foundations of Software Science. The results indicate that the majority of them represent neither natural laws nor useful engineering approximations.



# SOBRE ISHVEL

## TRABAJO DE MEMORIA

Un framework para la elaboración de tareas para cursos introductorios de programación.

# INVITACIÓN

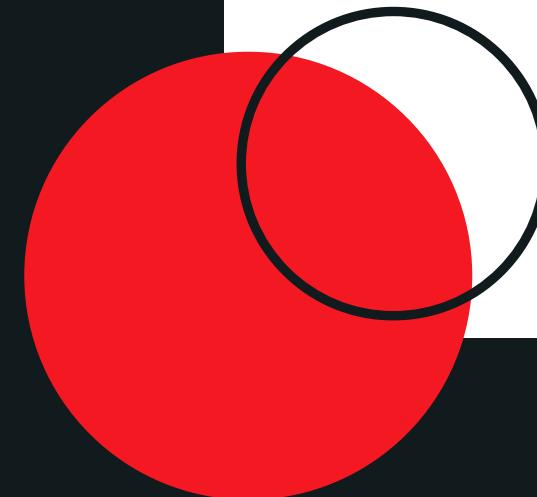
JS CONF 2023

Les extiendo la invitación a verme en la charla "Potenciando Javascript con WebAssembly: Un caso aplicado a la educación", de la Javascript Conf Chile 2023, donde veremos la culminación de todo este trabajo, y cómo implementar el cálculo de estas métricas desde un navegador que ejecuta Javascript.

The promotional card for JSConf Chile 2023 features a dark background with a yellow JSConf logo icon in the top left. To its right, the text "JSConf Chile 2023" is displayed. Below this, there is a circular portrait of a young man with dark hair and glasses, wearing headphones and holding a brass instrument. The portrait is set against a yellow gradient background. To the right of the portrait, the name "Gonzalo Fernández" is written in bold white text, followed by "Senior Full-Stack Developer" and "@Walmart". A thin vertical map of Chile is visible on the far right edge of the card.

# Evaluando la complejidad de una tarea en Python, con Python

Gonzalo Fernández -- Desarrollador Full-Stack



Respositorio con el código utilizado en el análisis de la complejidad de las tareas.