FACULTAD DE
INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

Introducción a Python

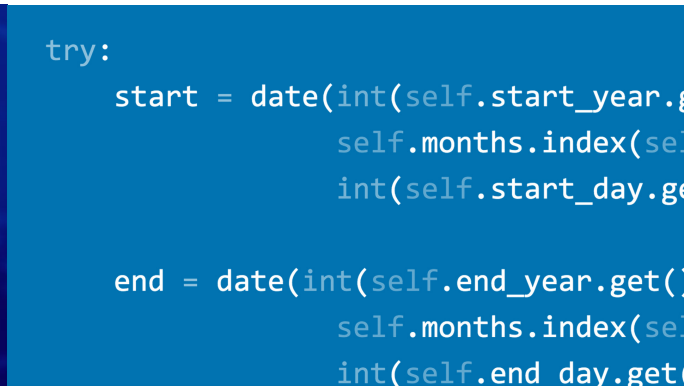
Manejo de errores

Miguel Carrasco
miguel.carrasco@uai.cl


Data
Scientist
Python
MATLAB

Temario

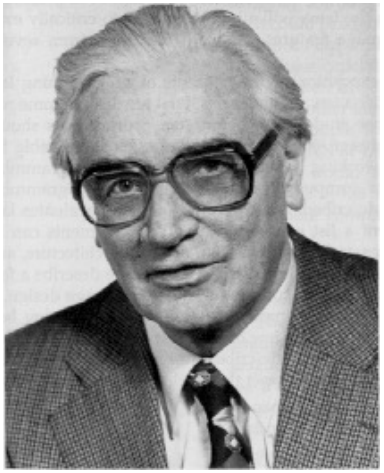
- Números Binarios
- Tipos de errores
  - Lógicos/sintácticos
  - Ejercicios



Historia de los computadores

- Los primeros computadores podían ser programados con secuencias de programas muy simples compuestos por valor binarios {unos y ceros} ¿Pero cómo se ingresaban los datos?

- Peso: 1000kg
- 1 Suma en 0.8seg
- 1 Multiplicación cada 3 seg



Computador Z3. Desarrollado por Konrad Zuse (1941)

más info en : https://www.dw.com/es/konrad-zuse-inventor-de-la-primera-computadora/a-5717799

* Imagenes tomadas desde Wikipedia

Historia de los computadores

- Debido a que los primeros computadores eran muy limitados, sólo podían realizar operaciones básicas (sumas y restas). Cada una de las operaciones era transformada a un *opcode o número específico*.

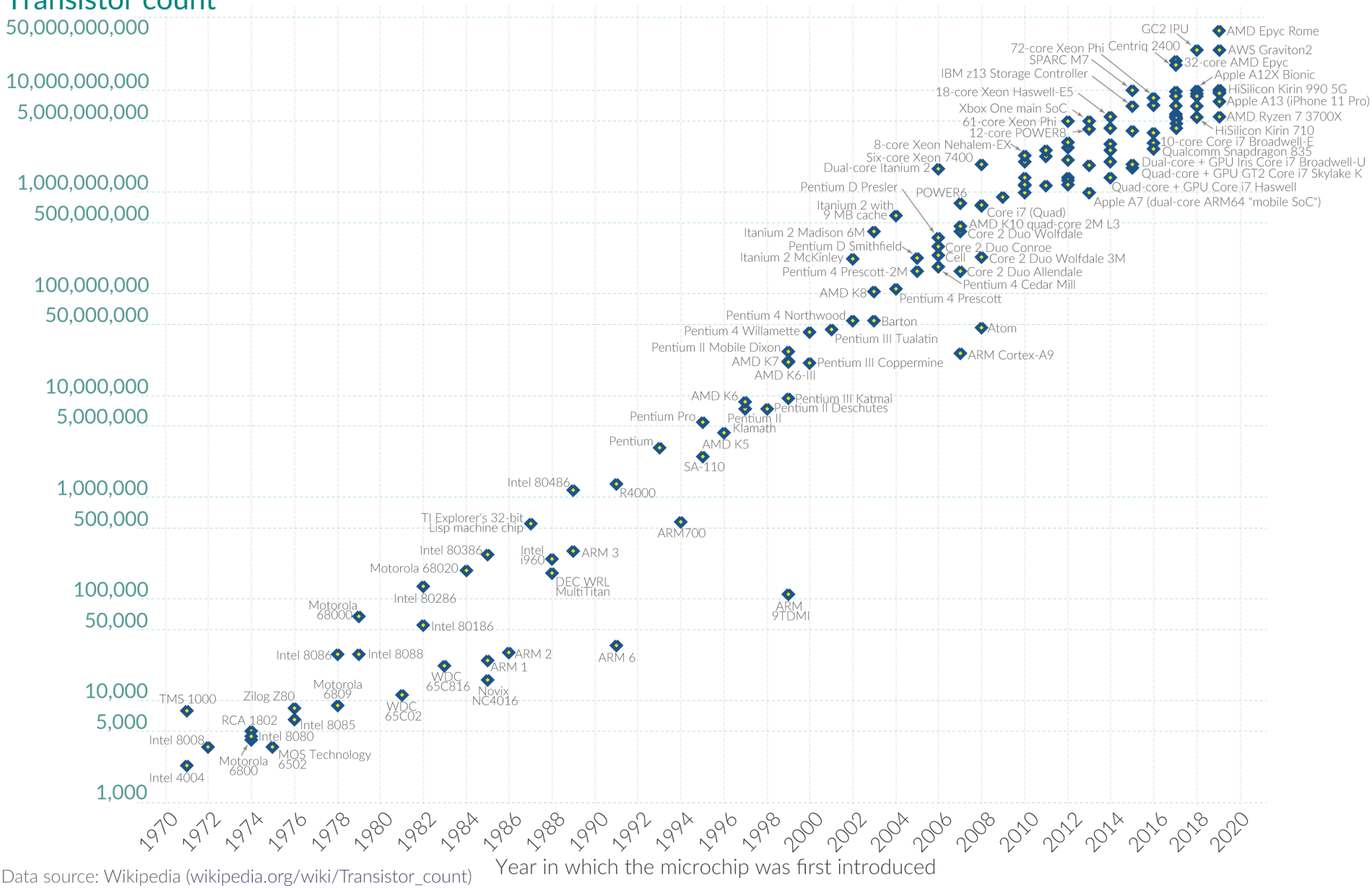
Computador	**ENIAC (1946)**	**AMD Ryzen Threadripper Pro 5995WX (2022/2Q)**
Operaciones	5000 sumas o 300 multiplicaciones / seg	33'935 MOps/Sec (64 núcleos-cores)
Tamaño	167 mts^2	74 mm^2
Consumo	150.000 watts/h	280 watts /h
Interruptores	17.648 (válvulas)	33,200,000,000 (33.4 billones)
Tamaño de transistor	5cm (cada válvula)	7nm cada transistor (*) (*) un nanómetro es la mil millonésima parte de un metro
Peso	27 toneladas	> 200grs
Programación	A través de interruptores	Lenguaje de máquina

(*) un Mops/Sec equivale a 10^6 millones de operaciones matemáticas or segundo. Es una métrica de evaluación

Moore's Law: The number of transistors on microchips doubles every two years

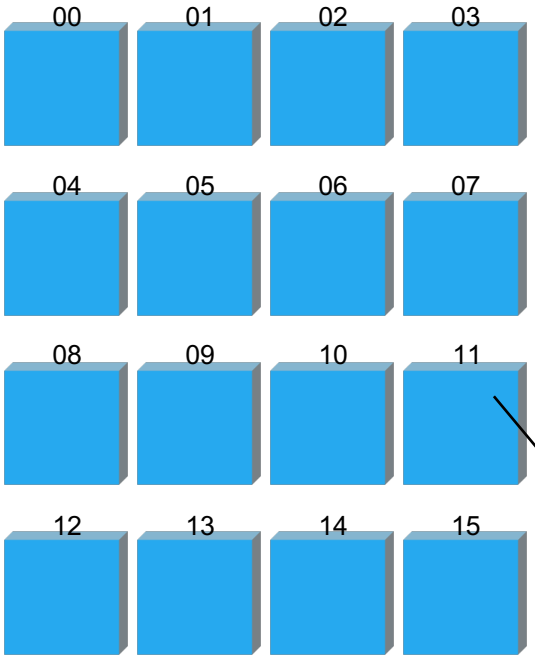
Our World in Data

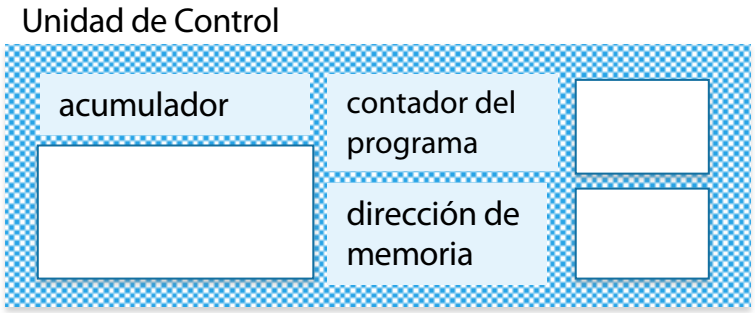
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección

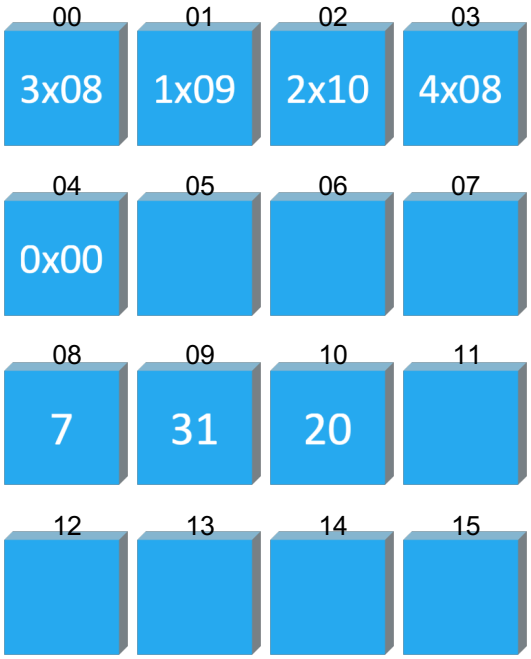


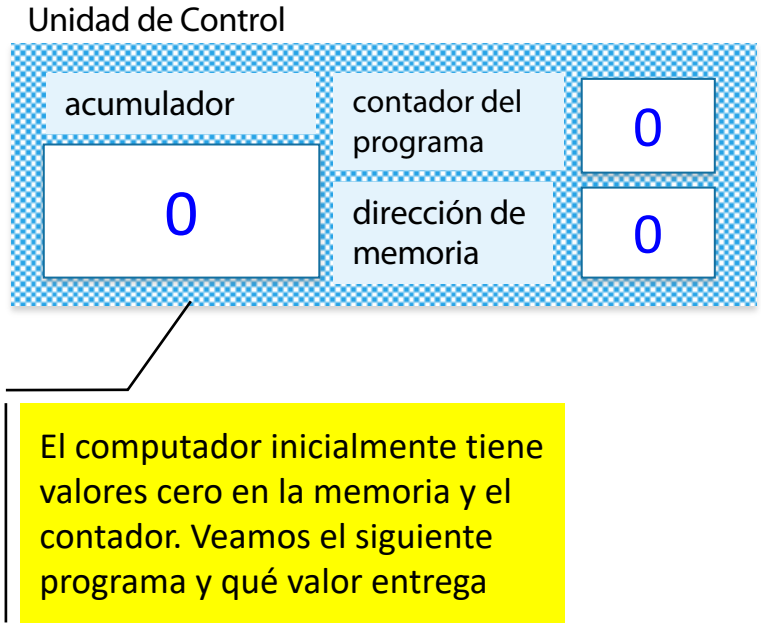


Las instrucciones y los valores se ingresan en la memoria. Luego el computador va recorrediendo una a una la memoria para ejecutar el programa

fuente: https://www.101computing.net/LMC/

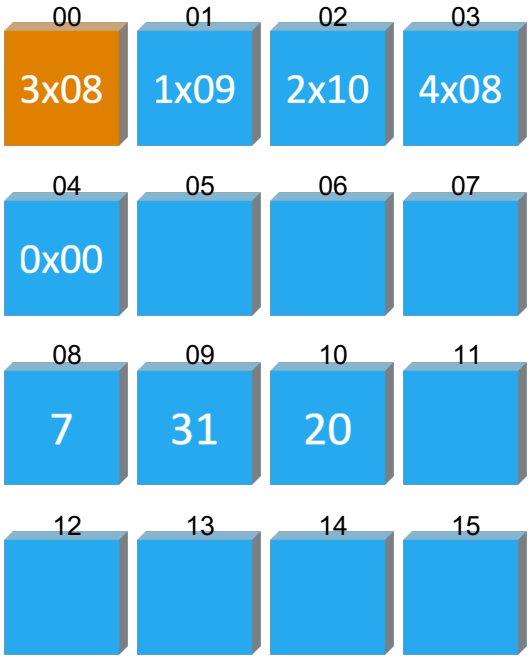
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección

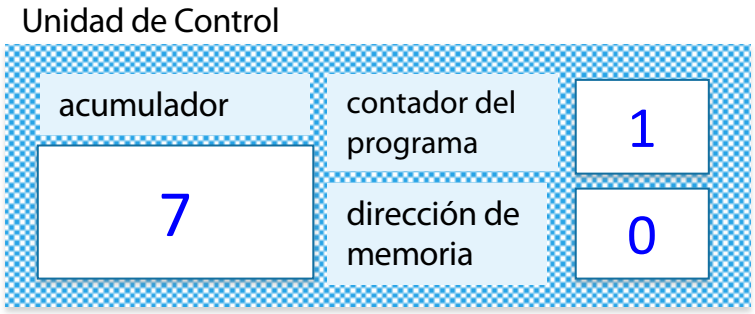




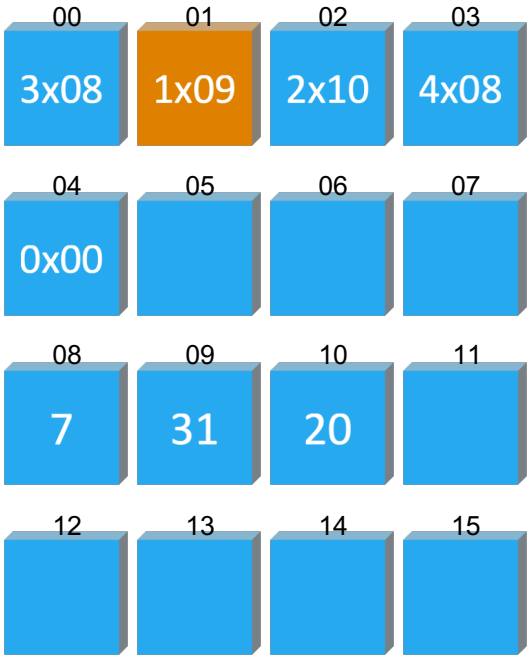
El computador inicialmente tiene valores cero en la memoria y el contador. Veamos el siguiente programa y qué valor entrega

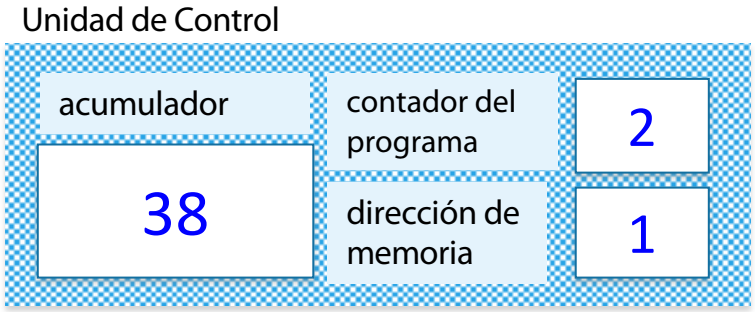
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección



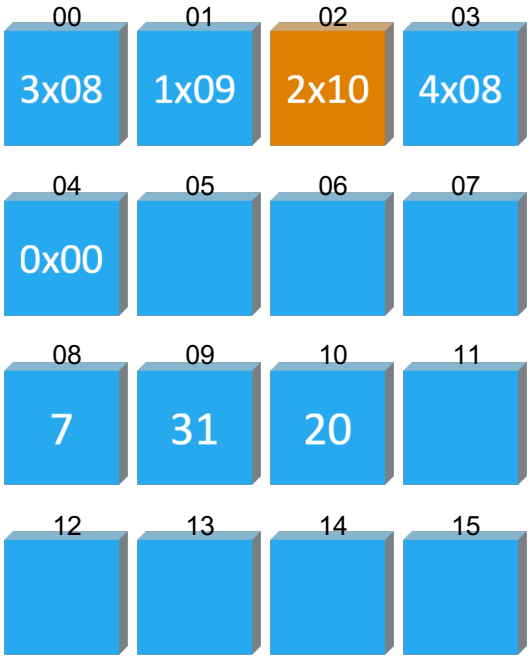


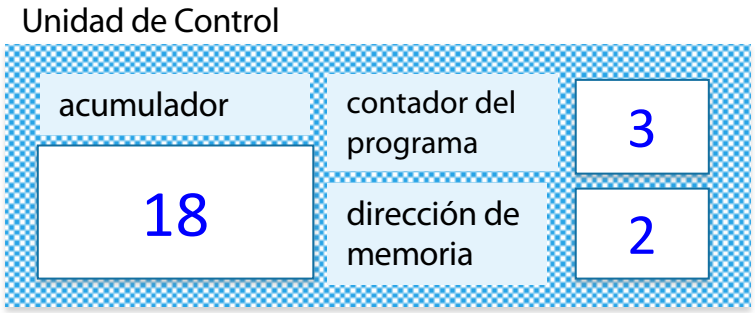
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección



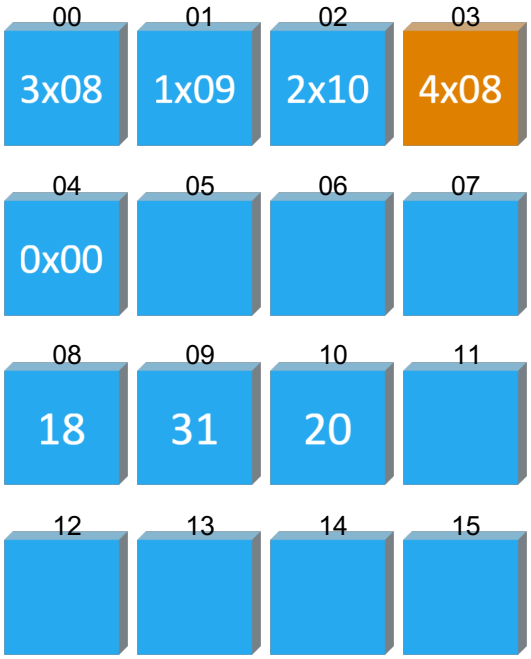


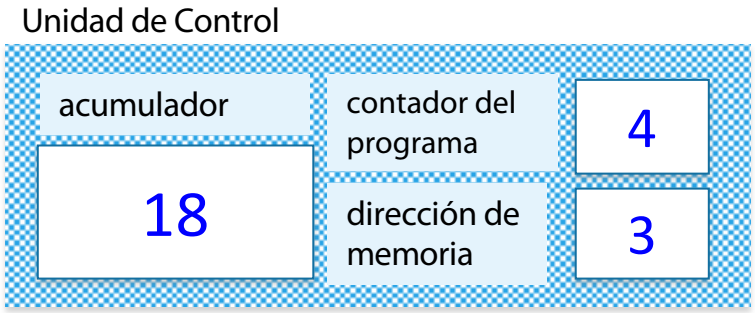
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección



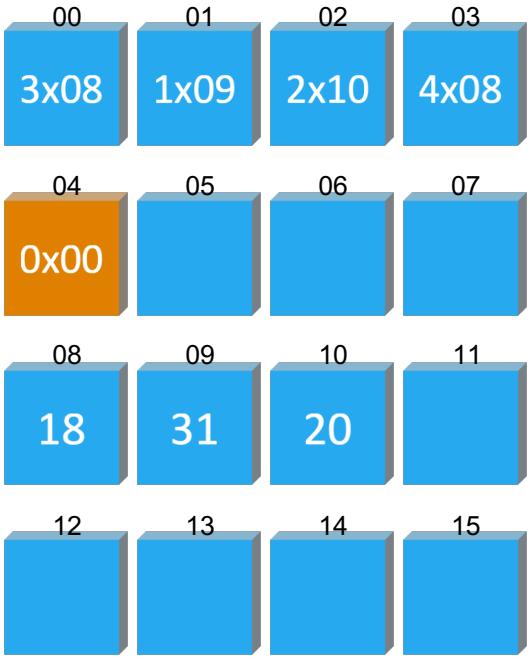


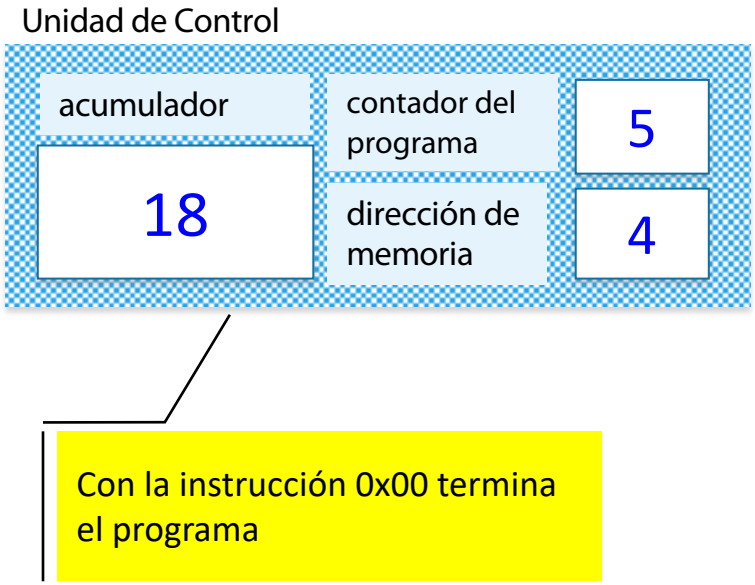
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección





Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección

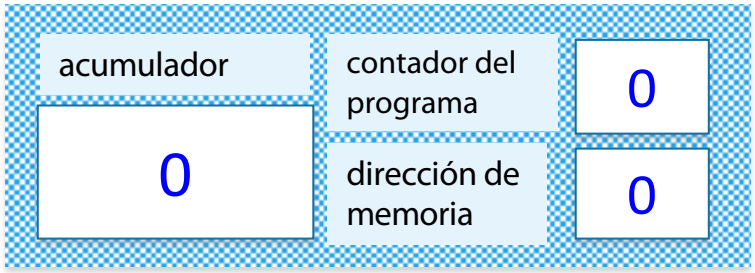




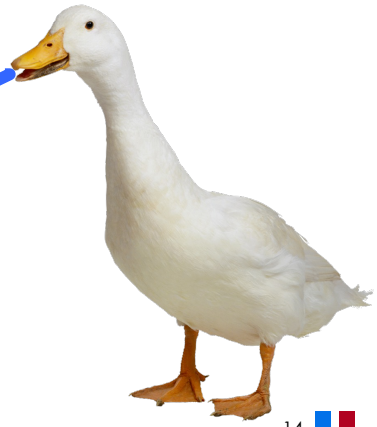
Operación	Opcode	Explicación
Suma (+)	1x	Suma el contenido de la memoria indicada al acumulador
Resta (-)	2x	Resta el contenido de la memoria indicada al acumulador
Carga (load)	3x	Carga el contenido de una dirección en el acumulador
Guarda (store)	4x	Guarda el contenido del acumulador en una dirección
Condición	5x	Si el acumulador es positivo o cero va a una dirección sino continúa

00	01	02	03
3x12	1x08	4x12	3x09
04	**05**	**06**	**07**
2x10	4x09	5x09	0x00
08	**09**	**10**	**11**
3	5	1	
12	**13**	**14**	**15**
3			

Unidad de Control

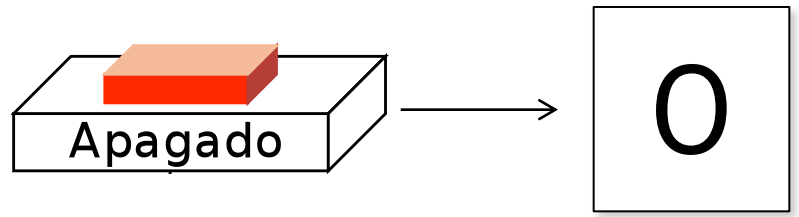


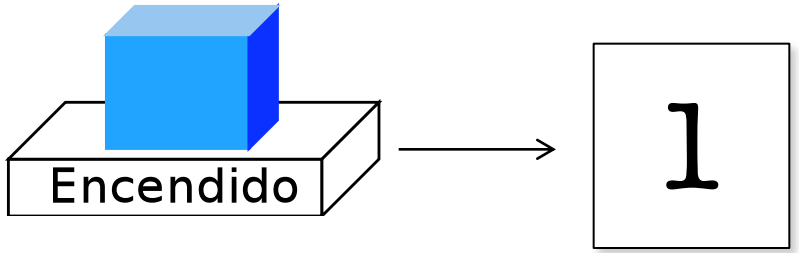
¿Qué valor entrega genera programa?.. Me supera. Donde se toma hora al sicólogo

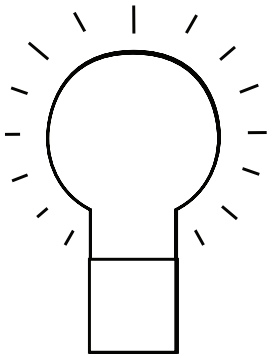


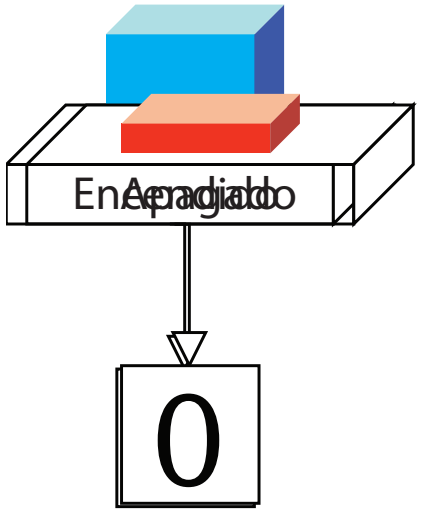
Bits y bytes

- Imagine 1 interruptor de su casa.
  ¿Cuantos estados tiene?





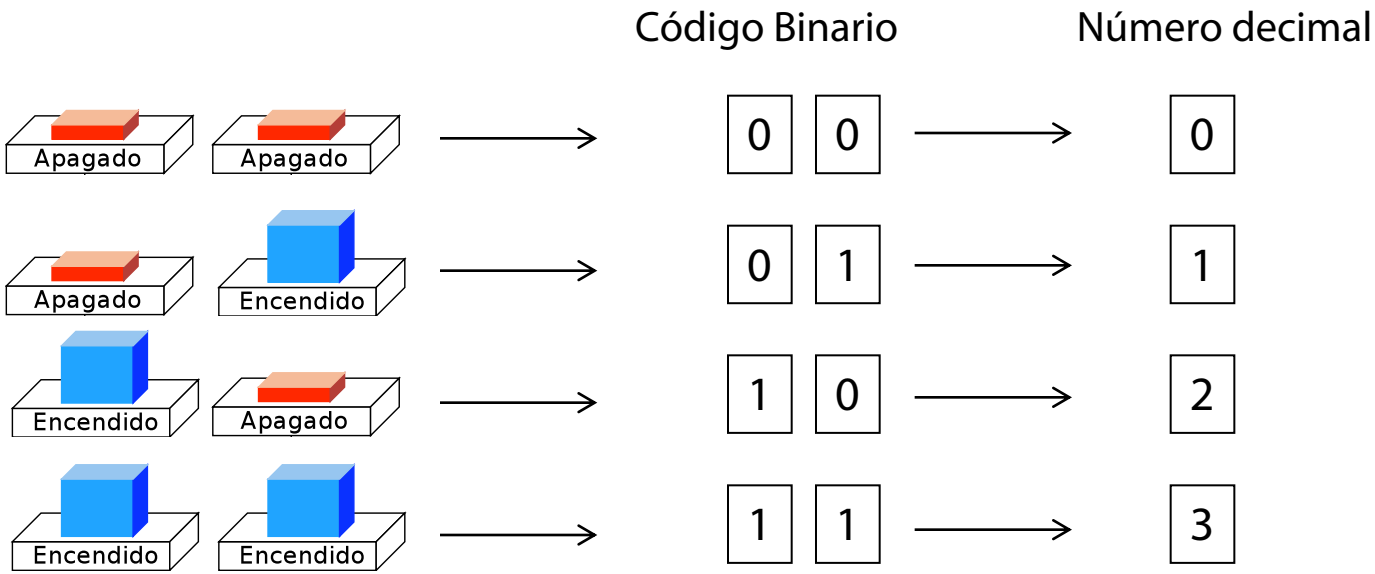




1 interruptor → 2^1 estados

Bits y bytes

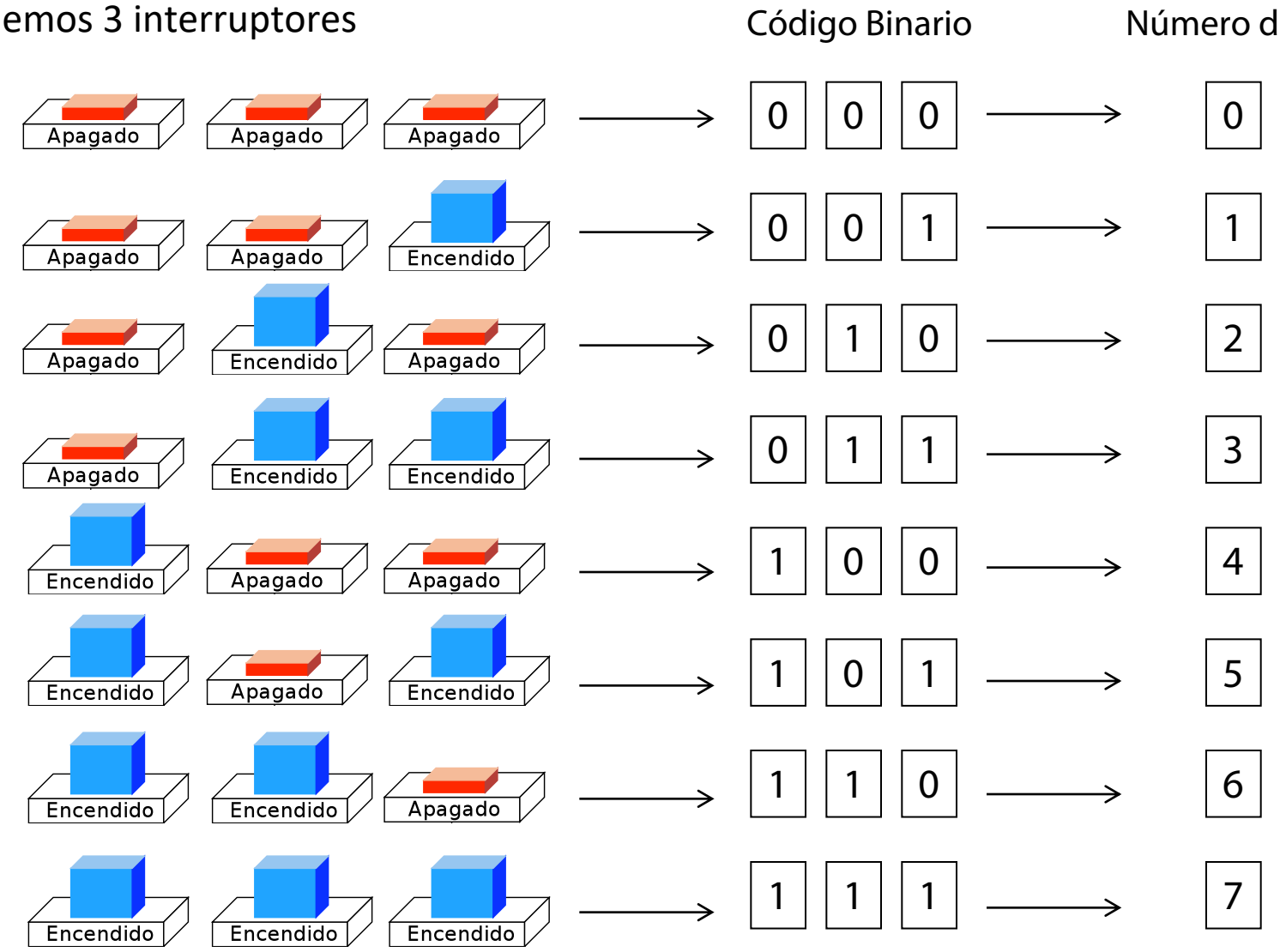
- Suponga que tenemos 2 interruptores



2 interruptores → 2^2 estados

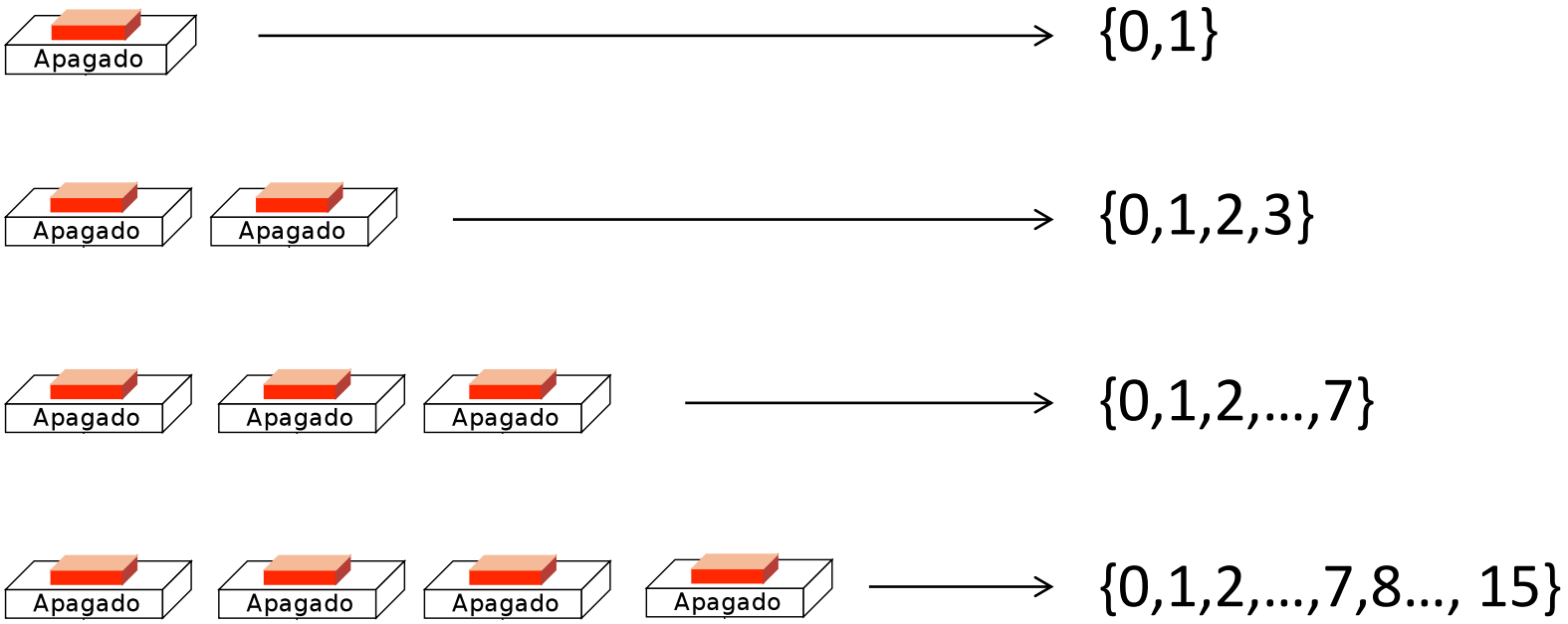
Bits y bytes

- Ahora suponga que tenemos 3 interruptores



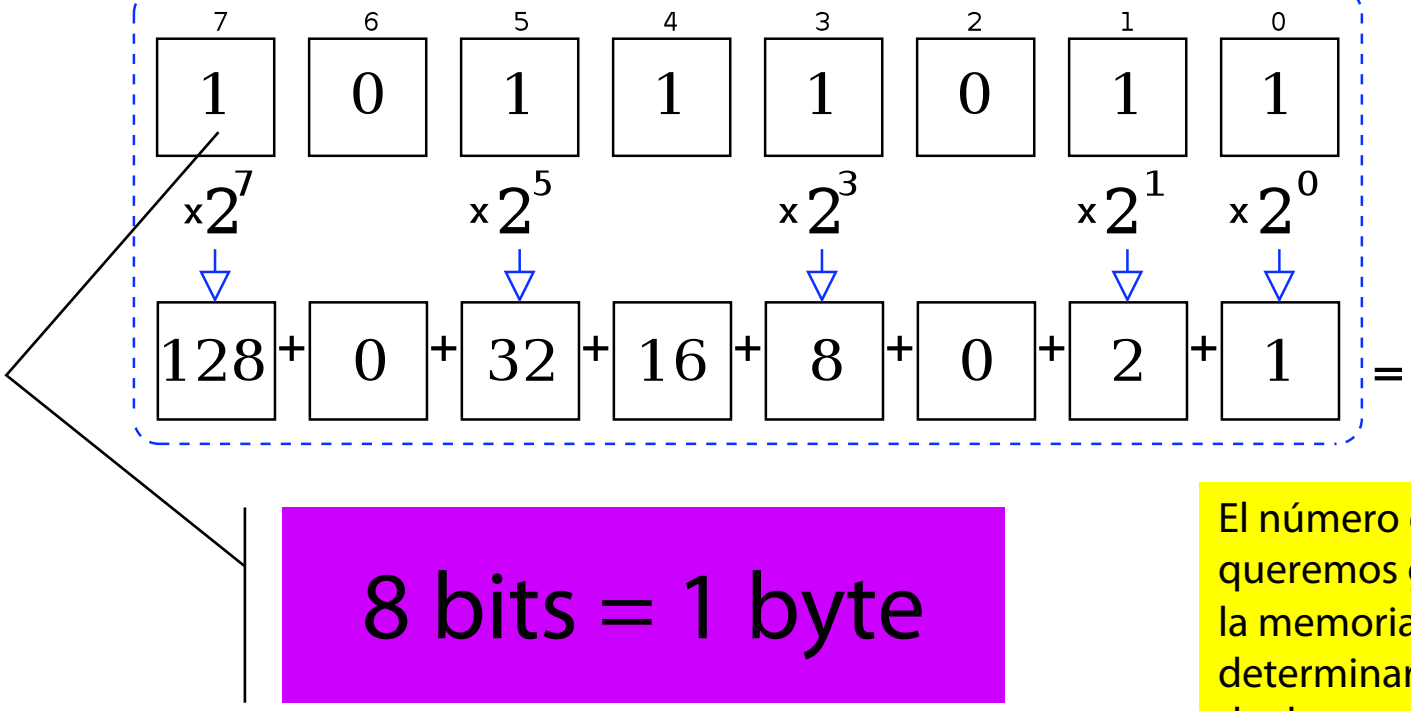
Bits y bytes

- Si queremos guardar el número **8**, cuantos bits necesitamos?



Bits y bytes

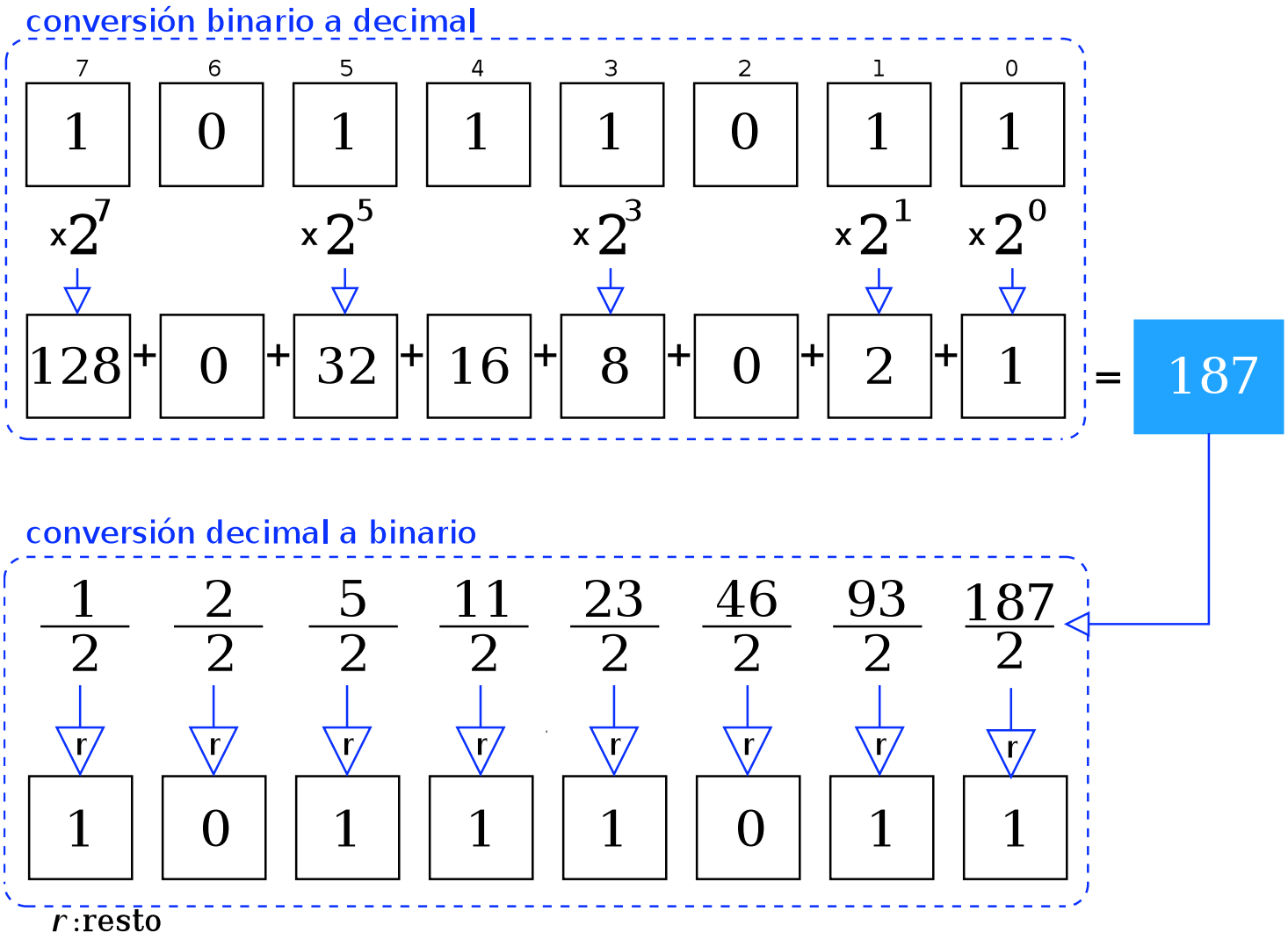
conversión binario a decimal



8 bits = 1 byte

El número que queremos guadar en la memoria determinará el tipo de datos empleado

Bits y bytes

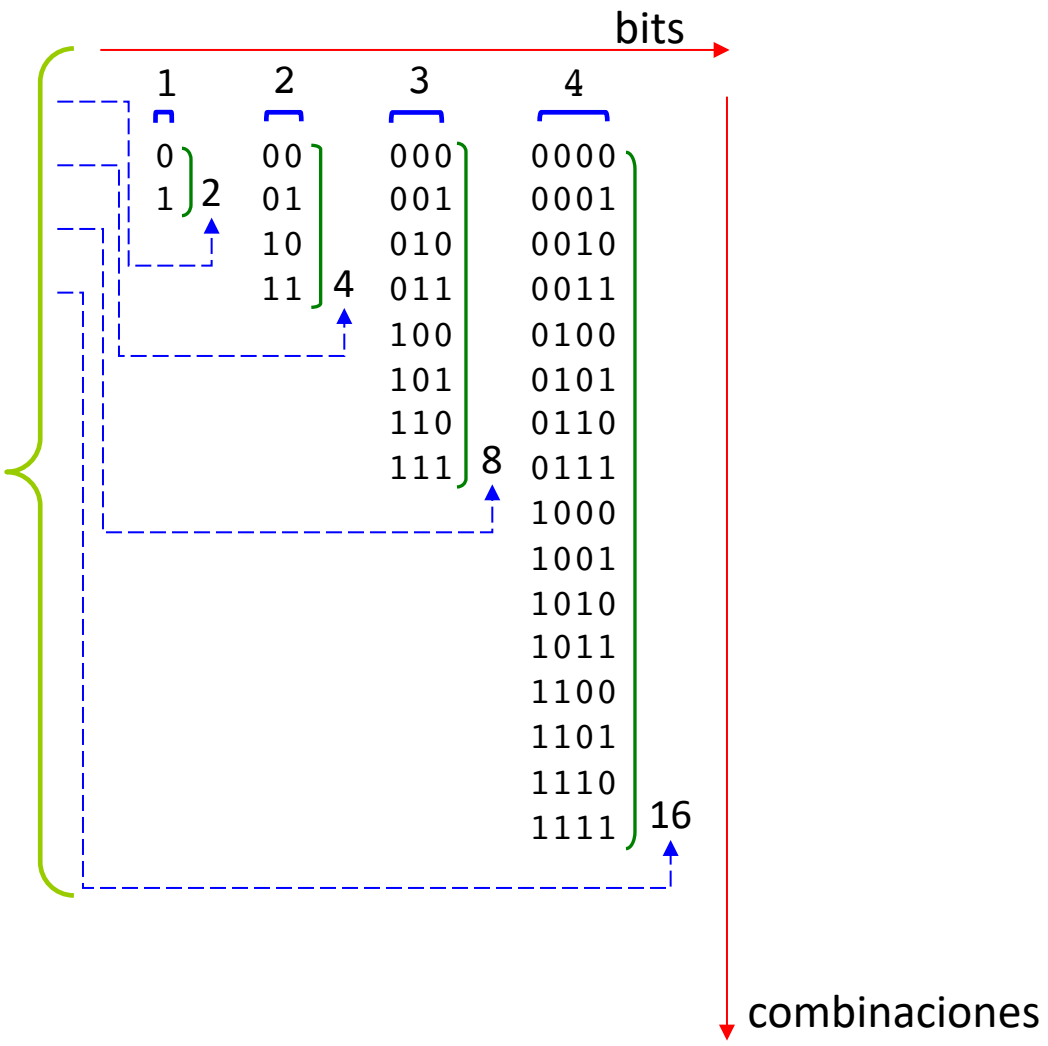

conversión binario a decimal
7 6 5 4 3 2 1 0
1 0 1 1 1 0 1 1
x2^7 x2^5 x2^3 x2^1 x2^0
128 + 0 + 32 + 16 + 8 + 0 + 2 + 1 = 187
conversión decimal a binario
1/2 2/2 5/2 11/2 23/2 46/2 93/2 187/2
r r r r r r r r
1 0 1 1 1 0 1 1
r :resto

Bus de datos

bits	# de combinaciones
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$ (1 kbit)
…	
15	$2^{15} = 32.768$ (32kbit)
16	$2^{16} = 65.536$ (64kbit)
…	
32	4.294.967.296 (4 Gbit)
64	17.5 millones de Terabytes

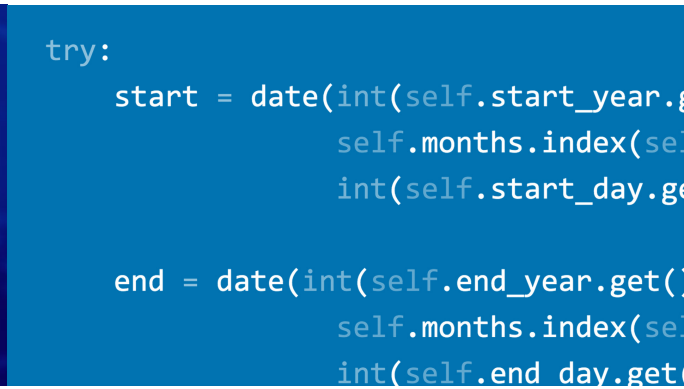
8 bits= 1 byte

16 bits ó 2 kB



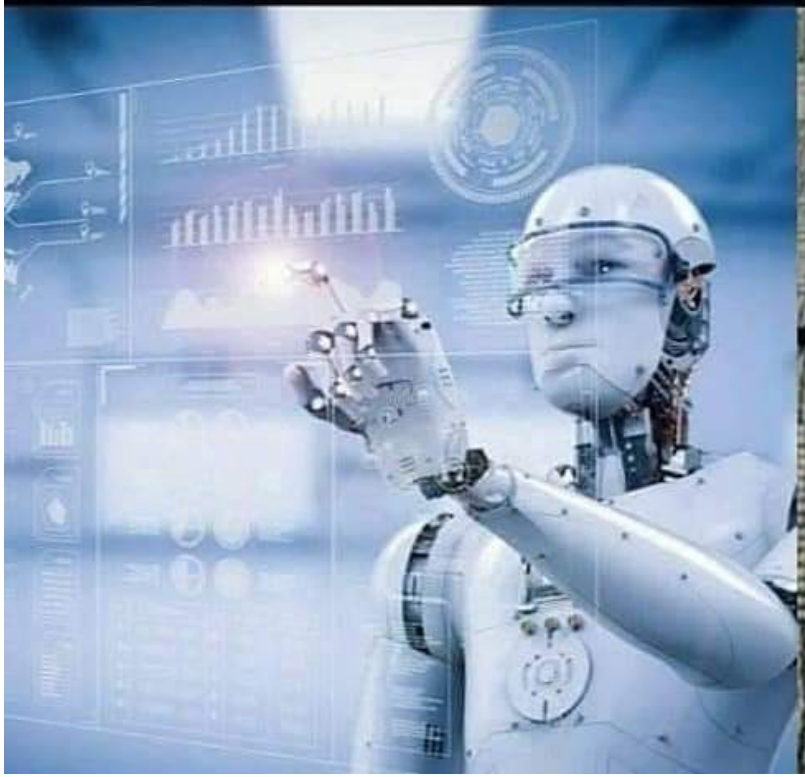
Temario

- Números Binarios
- Tipos de errores
  - Lógicos/sintácticos
  - Ejercicios

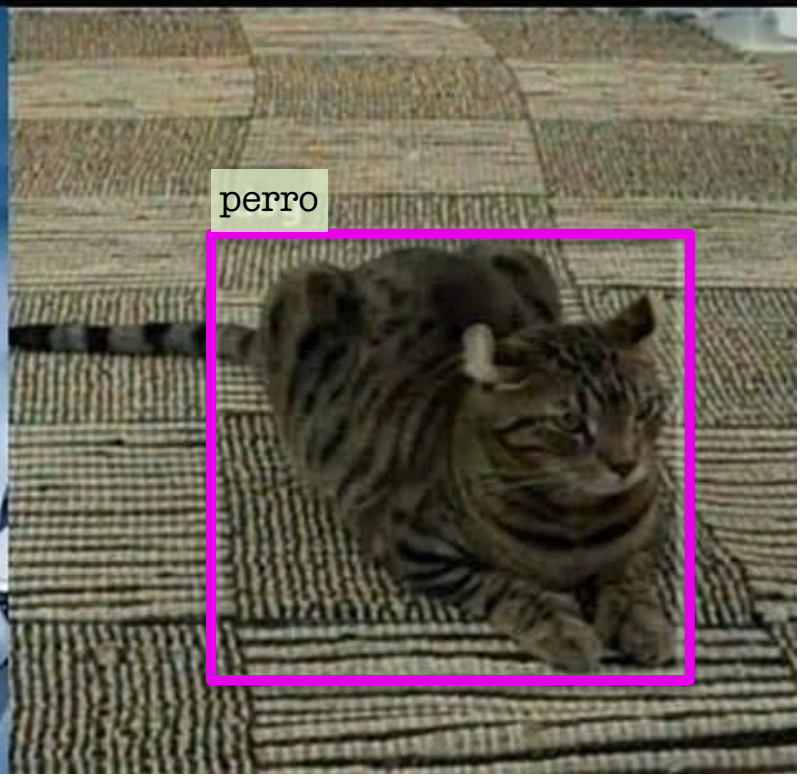


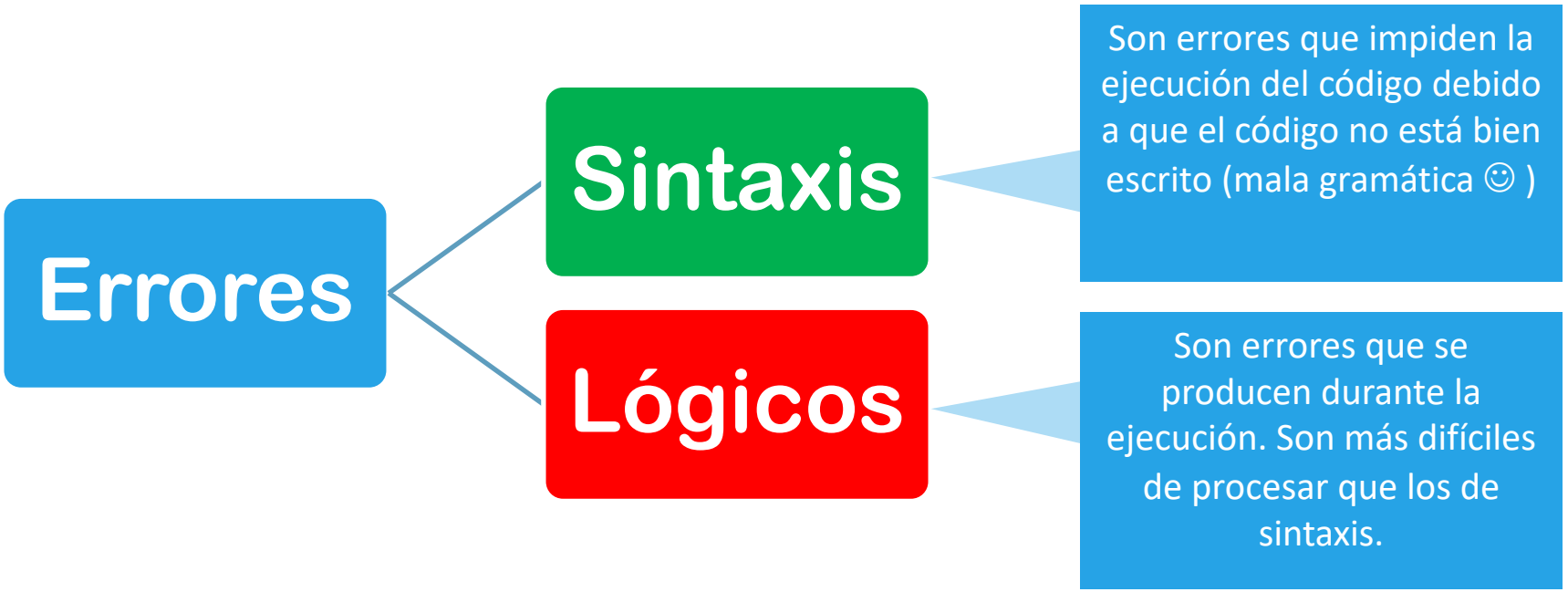
People in 1990: AI will probably be very advanced in 30 years

The AI they expected:



The AI now:




Errores
Sintaxis
Lógicos
Son errores que impiden la ejecución del código debido a que el código no está bien escrito (mala gramática ☺)
Son errores que se producen durante la ejecución. Son más difíciles de procesar que los de sintaxis.

Errores de Sintaxis

- Primero veremos los errores de sintaxis. Algunos ejemplos son:

1. Olvidar escribir las comillas al imprimir un texto
2. Olvidar definir una variable
3. Asignar una variable a un valor
4. Olvidar : al terminar una instrucción de inicio de bloque
5. Olvidar indentar(*)
6. Sobre indentar caminos alternativos de una condición
7. Olvidar un paréntesis
8. Usar el símbolo = en lugar de == para comparar
9. Usar una condición en un else
10. Errores de tipeo
11. Asignación a palabra reservada

(*) la palabra indentación no existe en la RAE. Es un anglisismo

Errores de Sintaxis

Python

```
print(Hola)
```

1. Olvidar escribir las comillas al imprimir un texto

Python

```
#error: olvidar comillas en print()
print(Hola)
```

```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    print (Hola)
NameError: name 'Hola' is not defined
```

Error

Línea donde se produce el error

Mensaje de error

EXPLICACIÓN

Si un parámetro de la función print() no lleva comillas, el sistema interpreta que el texto incluido en los paréntesis es un nombre de una variable. Como esa variable no ha sido definida, el error desplegado tiene relación con ese hecho: la supuesta variable (en este caso Hola) no ha sido definida.

SOLUCIÓN

1. Agregar las comillas si lo que queríamos era imprimir el texto “Hola”
2. Definir la variable Hola, si lo que queríamos era imprimir una variable

Errores de Sintaxis

Python

```
if x>3:
  print("Hola")
```

2. Olvidar definir una variable

Python

```
#error: olvidar definir una variable
if x>3:
  print("Hola")
```

```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    if(x>3):
NameError: name 'x' is not defined
```

Error

EXPLICACIÓN

Cada vez que usas una variable en tu código, debes asegurarte de que ha sido inicializada antes. Es decir, le has asignado un valor antes de imprimirla por pantalla o usarla con algún operador.

SOLUCIÓN

Inicializando la variable x antes de usarla:

```
#error: olvidar definir una variable
x=1  #se agrega esta línea
if(x>3):
  print("Hola")
```

Errores de Sintaxis

Python

```
int(input()) = y
```

3. Asignar una variable a un valor

Python

```
#error: Asignar una variable a un valor
int(input()) = y
```

Error

```
Traceback (most recent call last):
File "main.py", line 2, in <module>
    int(input()) = y
                 ^
SyntaxError: can't assign to function call
```

EXPLICACIÓN

Cada vez que desees asignar un valor a una variable, ya sea en forma directa o como resultado de una función, debes asegurarte que la variable esté al lado izquierdo del signo =, de lo contrario tratará de tomar el valor de la variable y asignarlo al valor o función

SOLUCIÓN

Variable = Valor

```
    #error: Asignar una variable a un valor
    y = int(input())  #arreglo
```

Errores de Sintaxis

Python

```
x = 1
if x>3
  print("Hola")
```

4. Olvidar agregar ":" al terminar una instrucción de inicio de bloque

Python

```
#error: Olvidar los : al terminar una instrucción de inicio de bloque
x = 1
if x>3
  print("Hola")
```

```
 Traceback (most recent call last):
 File "main.py", line 3
    if(x>3)
          ^
SyntaxError: invalid syntax
```

Error

EXPLICACIÓN

En este caso el mensaje de error no es tan explícito (solo te indica que hay un error de sintaxis, pero no te dice cuál). Sin embargo, hay un símbolo ^ que apunta al sector donde se detectó el error. Y puedes ver que está apuntando hacia el final de la instrucción if (al terminar la condición). Eso puede darte una pista de qué está faltando.

SOLUCIÓN

Agregar ":"

Errores de Sintaxis

Python

```
x = 1
if x>3:
print("Hola")
```

5. Olvidar indentar

Python

```
#error: Olvidar indentar
x = 1
if x>3:
print("Hola")
```

Error

```
 Traceback (most recent call last):
 File "main.py", line 4
    print("Hola")
        ^
IndentationError: expected an indented block
```

EXPLICACIÓN

Luego de una instrucción que termina con dos puntos (un if, un else, un while o un for) el sistema espera que el código tenga una o más instrucciones que se ejecutarán dependiendo de si se cumple o no la condición asociada a la instrucción. Esas instrucciones deben estar indentadas. Como el sistema no encuentra instrucciones indentadas, despliega un error.

SOLUCIÓN

Indentar las instrucciones que deben ejecutarse dentro del bloque if/else/while/for.

Errores de Sintaxis

Python

```
x = 1
if x>3:
  print("Hola")
  else:
    print("Chao")
```

6. Sobre indentar caminos alternativos de una condición

Python

```python
#error: sobre indentar
x = 1
if x>3:
  print("Hola")
  else:
    print("Chao")
```

Error

```
 Traceback (most recent call last):
 File "main.py", line 5
    else:
       ^
SyntaxError: invalid syntax
```

EXPLICACIÓN

Como ya mencionamos, las instrucciones dentro de un bloque if deben estar indentadas. Pero el else NO es parte del bloque if, sino que debe estar fuera como un camino alternativo alineado a la condición if.

SOLUCIÓN

Indentar los elif, else alineados a la columna donde se define la condición if

Errores de Sintaxis

Python

```
x = 1
if (x == 3) or (x == 1 or (x == 5):
    print('hola a todos!')
```

7. Olvidar un paréntesis

Python

```python
#error: olvidar un paréntesis
x = 1
if (x == 3) or (x == 1 or (x == 5):
    print('hola a todos!')
```

Error

```
 Traceback (most recent call last):
 File "main.py", line 3
    if (x == 3) or (x == 1 or (x == 5):
                                      ^
SyntaxError: invalid syntax
```

EXPLICACIÓN

Los paréntesis sirven para separar condiciones, u otras instrucciones, de forma de hacer nuestro código más ordenado. Sin embargo, a veces la multitud de paréntesis nos hacen olvidar cerrar alguno de ellos.

SOLUCIÓN

Si te aparece un error de este estilo lo mejor es, si usas paréntesis en esa línea, contar los paréntesis derecho y los izquierdo y asegurarte que estén balanceados

Errores de Sintaxis

Python

```
x = 0
if x = 0:
    print(x)
```

8. Usar el símbolo = en lugar de == para comparar

Python

```python
#error: usar = en vez de ==
x = 0
if x = 0:
    print(x)
```

Error

```
 Traceback (most recent call last):
 File "main.py", line 3
    if (x=0):
         ^
SyntaxError: invalid syntax
```

EXPLICACIÓN

El símbolo = se usa para asignar valor a una variable, por lo que al hacer x=0 lo que hacemos es otorgarle valor 0 a la variable x. Esta asignación no entrega un valor de verdad, por lo que no se puede utilizar en una condición.

SOLUCIÓN

Debes asegurarte que en una condición uses == en lugar de =. Recuerda que los comparadores válidos son: ==, !=, <, <=, >, >=

Errores de Sintaxis

Python

```
x = 0
if x == 0:
    print(x)
else x != 0:
    print(x)
```

9. Usar una condición en la instrucción else

Python

```python
#error: usar una condición en el else
x = 0
if x == 0:
    print(x)
else x != 0:
    print(x)
```

```
 Traceback (most recent call last):
 File "main.py", line 5
    else (x!=0):
         ^
SyntaxError: invalid syntax
```

Error

EXPLICACIÓN

Dentro de un condicional de tipo if…else… el bloque else cubre el caso que es el complemento de la condición definida en el if. Por ende, NO debe llevar nunca una condición específica.

SOLUCIÓN

1. Eliminar la condición del else
2. Cambiar else por elif si realmente deseaba agregar otra condición

Errores de Sintaxis

Python

```
x = 0
if x == 0:
    print = ('El valor de x es:',x)
else:
    print(x)
print('termino del programa')
```

10. Asignar un valor a una palabra reservada

Python

```python
#asignar un valor a una palabra reservada
x = 0
if x == 0:
    print = ('El valor de x es:',x)
else:
    print(x)
print('termino del programa')
```

```
 Traceback (most recent call last):
 File ·main.py", line 8, in <module>
    print('termino del programa')

TypeError: 'tuple' object is not callable
```

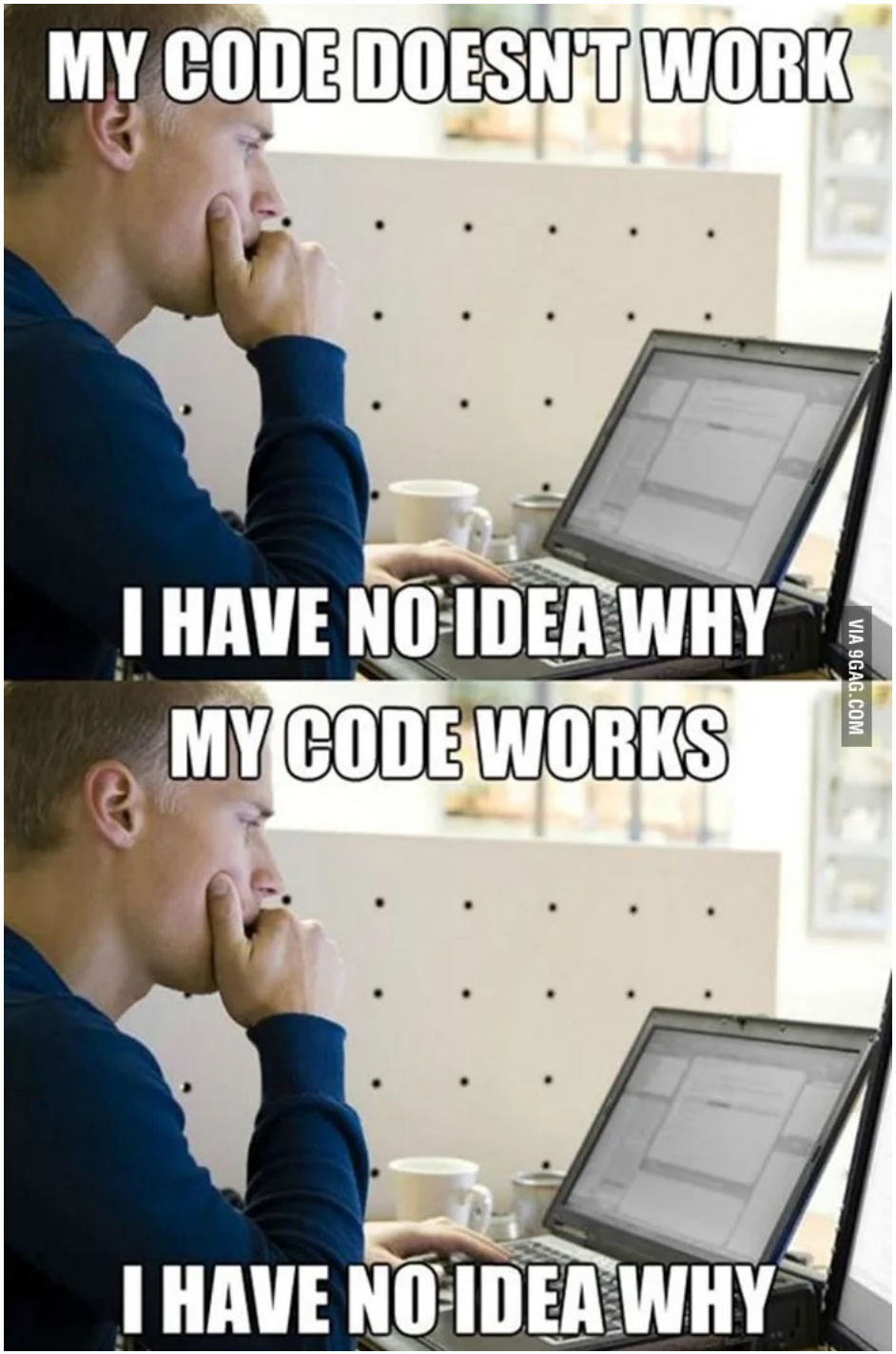
Error

EXPLICACIÓN

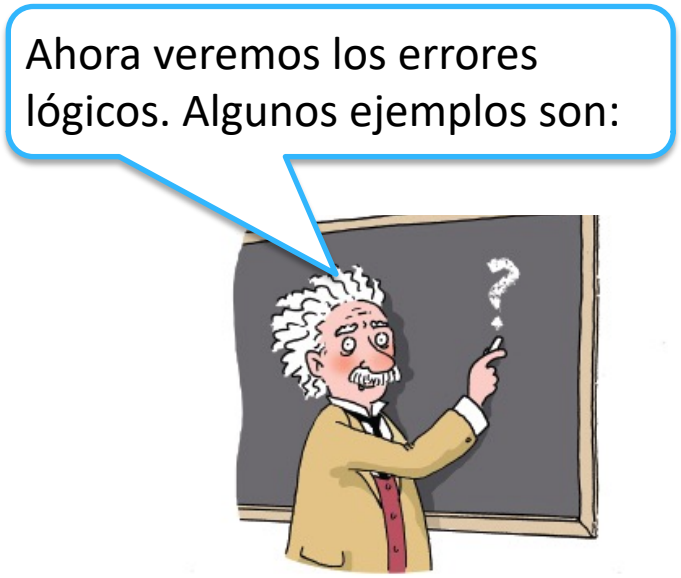
La palabras reservadas no deben ser modificadas (asignarle un valor) ,ya que al hacerlo, cambia su funcionalidad dentro del programa.

SOLUCIÓN

1. No asignar un valor a una palabra reservada


MY CODE DOESN'T WORK
I HAVE NO IDEA WHY
VIA 9GAG.COM
MY CODE WORKS
I HAVE NO IDEA WHY

Errores Lógicos



1. Condición mal definida (en if o en while)
2. Variable mal actualizada
3. Instrucción mal posicionada

Errores Lógicos

Escriba un programa en Python que solicite un número entre 2 y 5 (inclusive). Si el usuario ingresa un número fuera de este rango entonces el programa debe continuar solicitando el número hasta que sea válido.

Python

```
print("Ingresa numero entre 2 y 5, ambos inclusive")
x = int(input())
while x<2 and x>5:
   print("Ingresa numero entre 2 y 5, ambos inclusive")
   x = int(input())
```

1. Condición mal planteada (en if o while)

Escriba un programa en Python que solicite un número entre 2 y 5 (inclusive). Si el usuario ingresa un número fuera de este rango entonces el programa debe continuar solicitando el número hasta que sea válido.

Python

```
#error: condición mal planteada
print("Ingresa numero entre 2 y 5, ambos inclusive")
x = int(input())
while x<2 and x>5:
   print("Ingresa numero entre 2 y 5, ambos inclusive")
   x = int(input())
```

EXPLICACIÓN

En este caso el programador está verificando que el número ingresado por el usuario no sea menor de 2 ni mayor de 5....¡¡al mismo tiempo!! Eso no puede ocurrir, ¿cierto?

SOLUCIÓN

La solución correcta es que verifique si el número ingresado es menor a 2 o mayor a 5:
while(x<2 or x>5):

Errores Lógicos

Escriba un programa en Python que calcula la sumatoria de los N primeros números naturales, usando un ciclo while. N es un número ingresado por el usuario.

Python

```python
print("Ingresa la cantidad de números naturales")
N = int(input())
suma = 0
i = 1
while i<=N:
    suma = i
    i = i+1
print("La suma es igual a: ", suma)
```

2. Variable mal actualizada

Escriba un programa en Python que calcula la sumatoria de los N primeros números naturales, usando un ciclo while. N es un número ingresado por el usuario.

Python

```python
#error: variable mal actualizada
print("Ingresa la cantidad de números naturales")
N = int(input())
suma = 0
i = 1
while i<=N:
    suma = i   #variable mal actualizada
    i = i+1
print("La suma es igual a: ", suma)
```

EXPLICACIÓN

En este caso el programador está asignando el valor de i a la suma en cada iteración del ciclo, haciendo que el valor final de la suma sea el último valor de i, o sea N.

SOLUCIÓN

La solución correcta es ACUMULAR la suma:

suma = suma + i

Errores Lógicos

Escriba un programa en Python que calcula la sumatoria de los N primeros números naturales, usando un ciclo while. N es un número ingresado por el usuario.

Python

```python
print("Ingresa la cantidad de números naturales")
N = int(input())
i = 1
while i<=N:
    suma = 0
    suma = suma + i
    i = i+1
print("La suma es igual a: ", suma)
```

3. Instrucción mal posicionada

Escriba un programa en Python que calcula la sumatoria de los N primeros números naturales, usando un ciclo while. N es un número ingresado por el usuario.

Python

```python
#error: variable mal actualizada
print("Ingresa la cantidad de números naturales")
N = int(input())
i = 1
while i<=N:
    suma = 0        #instrucción mal posicionada
    suma = suma + i
    i = i+1
print("La suma es igual a: ", suma)
```

EXPLICACIÓN

En este caso el programador está inicializando el valor de la suma a 0 en cada iteración del ciclo, lo que hace que suma siempre valga 0 + i al término de ciclo, dando un resultado igual que el ejemplo anterior.

SOLUCIÓN

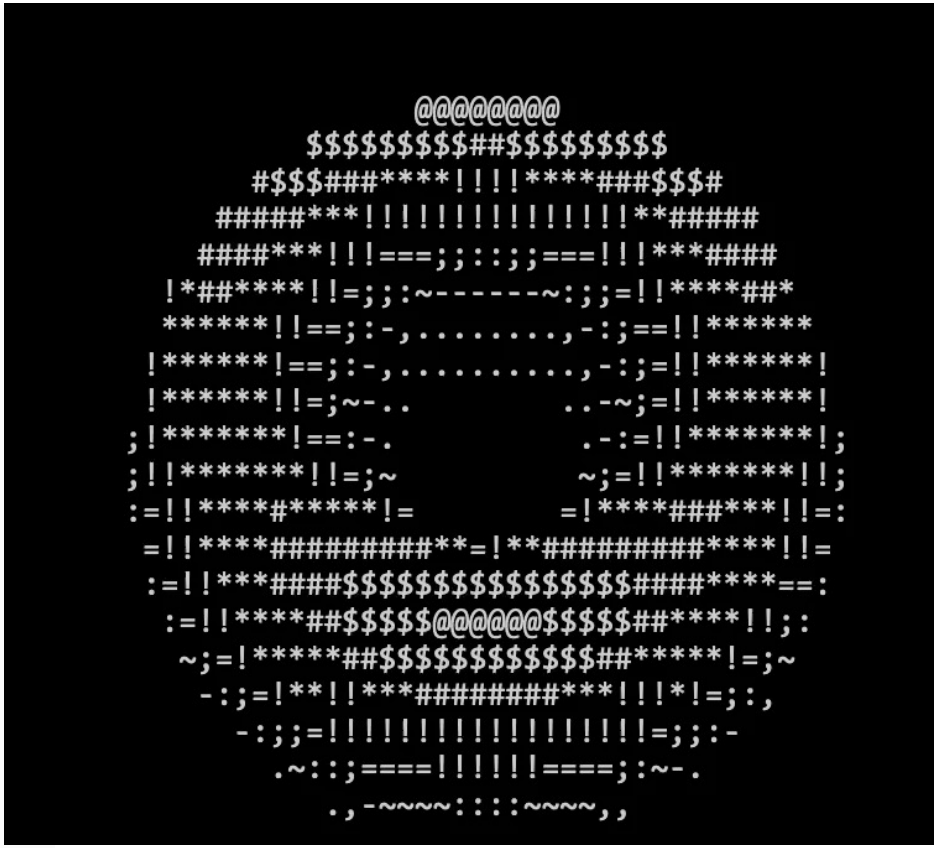
La solución correcta es inicializar las variables de acumulación ANTES del ciclo.

Obfuscated code

código en C

```
              k;double sin()
          ,cos();main(){float A=
        0,B=0,i,j,z[1760];char b[
      1760];printf("\x1b[2J");for(;;
   ){memset(b,32,1760);memset(z,0,7040)
   ;for(j=0;6.28>j;j+=0.07)for(i=0;6.28
  >i;i+=0.02){float c=sin(i),d=cos(j),e=
  sin(A),f=sin(j),g=cos(A),h=d+2,D=1/(c*
  h*e+f*g+5),l=cos      (i),m=cos(B),n=s\
in(B),t=c*h*g-f*        e;int x=40+30*D*
(l*h*m-t*n),y=            12+15*D*(l*h*n
+t*m),o=x+80*y,          N=8*((f*e-c*d*g
 )*m-c*d*e-f*g-l        *d*n);if(22>y&&
 y>0&&x>0&&80>x&&D>z[o]){z[o]=D;;;b[o]=
 ".,-~:;=!*#$@"[N>0?N:0];}}/*#****!!-*/
  printf("\x1b[H");for(k=0;1761>k;k++)
   putchar(k%80?b[k]:10);A+=0.04;B+=
     0.02;}}/*****####*******!!=;:~
       ~::==!!!**********!!!==::-
         .,~~;;;========;;;:~-.
             ..,--------,*/
```

resultado



más info en: https://www.a1k0n.net/2011/07/20/donut-math.html

- Tipos de errores
  - Lógicos/sintácticos
  - Ejercicios

```
try:
    start = date(int(self.start_year.
                     self.months.index(se
                     int(self.start_day.ge

    end = date(int(self.end_year.get(
                     self.months.index(se
                     int(self.end_day.get
```

Ahora tú

Detecta los errores en cada uno de los programas que aparecerán a continuación. Luego menciona cómo se pueden arreglar los errores que hayas encontrado

PROGRAMA 1

Escriba un programa en Python que imprima los primeros 10 números naturales

Python

```
i = 1

while i<=10:
    print(i)
```

Detecta los errores en cada uno de los programas que aparecerán a continuación. Luego menciona cómo se pueden arreglar los errores que hayas encontrado

PROGRAMA 2

Escriba un programa en Python que imprima los números desde N (ingresado por el usuario) hasta 0

Python

```
print("Ingresa un número positivo mayor que 0")
x = int(input())

while x<=0:
    print("Ingresa un número positivo mayor que 0")
    x = int(input())

while x<0:
    print(x)
    x = x-1
```

Detecta los errores en cada uno de los programas que aparecerán a continuación. Luego menciona cómo se pueden arreglar los errores que hayas encontrado

PROGRAMA 3

Escriba un programa en Python que solicite un número par. Si el usuario ingresa un número impar, debe volver a solicitar un número. Luego de que se ha ingresado un número par, debe imprimir si es mayor a 10 o no

Python

```python
print("Ingresa un número par")
n = int(input())

while n%2==1:
    print("Ingresa un número par")

if n>10:
    print("Numero par mayor que 10")
else:
    print("Numero par menor o igual a 10")
```

Detecta los errores en cada uno de los programas que aparecerán a continuación. Luego menciona cómo se pueden arreglar los errores que hayas encontrado

PROGRAMA 4

Escriba un programa en Python que detecta si el usuario está en su infancia (<13 años), adolescencia (entre 13 y 20 años, ambos inclusive) o adultez (>= 21 años)

Python

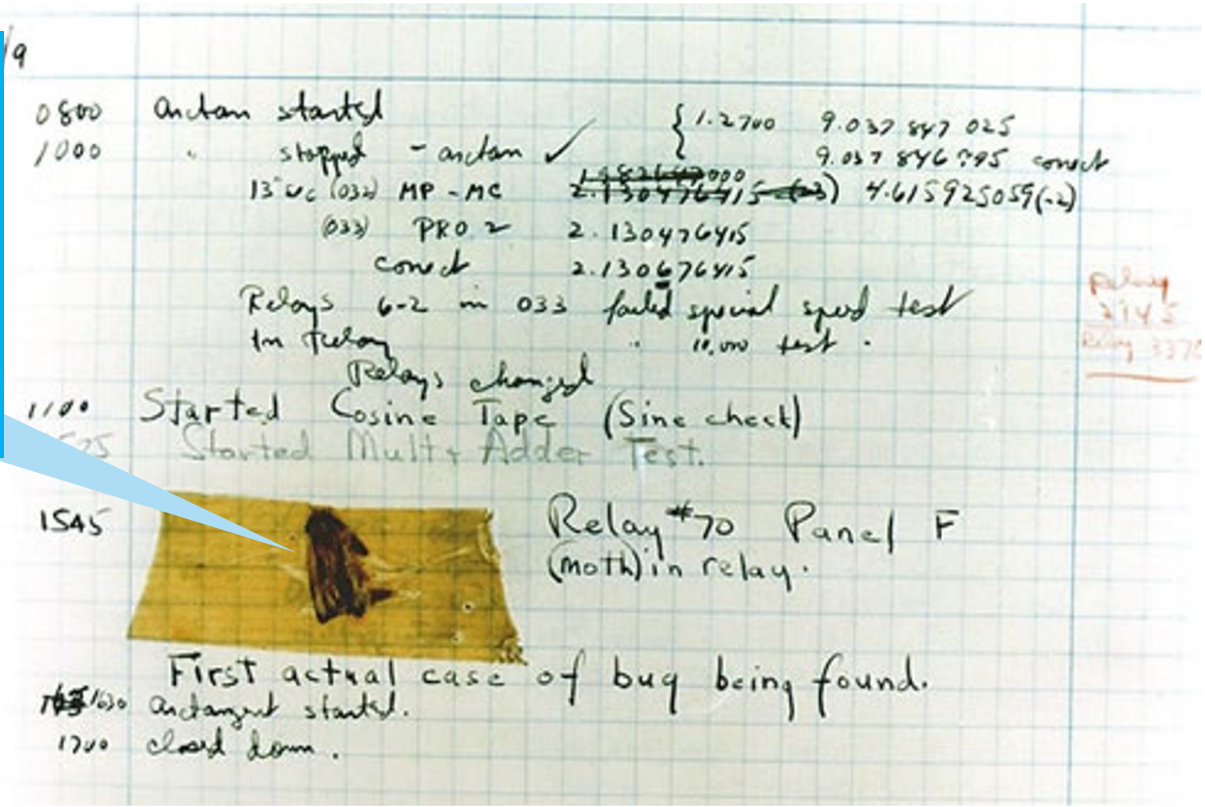
```python
print("Ingresa tu edad")
edad = int(input())

if edad <13:
    print("Estas en tu infancia")

if edad>=13 and edad<=20:
    print("Estas en tu adolescencia")
else:
    print("Estas en tu adultez")
```

Debugging es el arte de encontrar errores en nuestro programa y eliminarlos

Se llama así pues el primer error en un computador se produjo porque una polilla se metió en los relés de un computador Mark II



fuente: https://mujeresconciencia.com/2020/01/02/la-matematica-grace-hopper-y-el-revuelo-de-la-polilla/

Grace Hopper en la estación de control de la UNIVAC I (hacia 1960). Imagen: Wikimedia.

Ayuda en Python

Para encontrar errores hay métodos tanto manuales como dentro de Python.

MANUALMENTE

- Tablas de Ruteo

CON PYTHON

- *Debugging con* print()
- *El método* type()
- *El método* isdigit()
- *El método* isinstance()
- *El método* assert


Hey, are you sleeping?
Yes, now shut up
I know how to fix that bug on line 255

Tabla de Ruteo

Una tabla de ruteo es una manera de analizar el valor de variables que existan en el código de forma que podamos detectar errores sin ejecutar el programa

NOTA:

No es solo para errores, sino que también para poder determinar lo que hace el programa en sí

Ejemplo

Python

```
print("Ingresa un número positivo mayor que 0")
x = int(input())

while x<=0:
    print("Ingresa un número positivo mayor que 0")
    x = int(input())

while x>0:
    print(x)
    x = x-1
```

x	Pantalla	Comentario
	Ingresa un número positivo mayor que 0	
-1		
	Ingresa un número positivo mayor que 0	
7		
	7	
6		

Tabla de ruteo

x	Pantalla	Comentario
	Ingresa un número positivo mayor que 0	
-1		
	Ingresa un número positivo mayor que 0	
7		
	7	
6		
	6	
5		
	5	
4		
	4	
3		
	3	
2		
	2	
1		
	1	
0		

Ejemplo

¿Qué hace el programa?

PROGRAMA

Escriba un programa en Python que imprima los números desde N (ingresado por el usuario) hasta 1

```
print("Ingresa un número positivo mayor que 0")
x = int(input())

while x<=0:
    print("Ingresa un número positivo mayor que 0")
    x = int(input())

while x>0:
    print(x)
    x = x-1
```

Pasos

Escribir en una tabla TODAS las variables que aparecen en el programa

El uso de la columna pantalla y comentarios es opcional

Ir línea por línea en su programa y escribir en su tabla cada vez que exista una ASIGNACIÓN que cambia de valor una variable de su programa. Nuevamente, si usa la columna pantalla, también coloque lo que se muestra

Notas:

Número de valores nuevos en las mismas filas para distintas variables. La idea de una tabla de ruteo es ver si una variable tiene valor o no, y qué valor tiene si es que lo tiene.

Ahora tú

Python

```
print("Ingresa un numero")
N = int(input())
i = 1
while i < N+1:
    x = 2**i
    print(x)
    i = i+1
```

Para el programa anterior:

1. Construye su tabla de ruteo para un dato de entrada igual a 4
2. Explica qué hace este programa, en español, de modo que alguien que no sepa programación entienda. Al final de la explicación, agrega un ejemplo específico. NO deben explicar el programa línea a línea, sino su OBJETIVO

Tabla de ruteo

N	i	x	Pantalla	Comentario
--	--	--	Ingresa un numero	Inicio Programa
4	--	--		Ingreso valor 4
4	1	--		i = 1
4	1	2		x = 2**i
4	1	2	2	Imprimo
4	2	2		i = i + 1
4	2	4		x = 2**i
4	2	4	4	Imprimo
4	3	4		i = i + 1
4	3	8		x = 2**i
4	3	8	8	Imprimo
4	4	8		i = i + 1
4	4	16		x = 2**i
4	4	16	16	Imprimo
4	5	16		Fin Programa

Ayuda en Python

Para encontrar errores hay métodos tanto manuales como dentro de Python.

MANUALMENTE

- Tablas de Ruteo

CON PYTHON

- *Debugging con* print()
- *El método* type()
- *El método* isdigit()
- *El método* isinstance()
- *El método* assert

Ejemplo

¿Qué hace el programa?

PROGRAMA

Este programa imprime las primeras N potencias de 2, una por línea. Por ejemplo, para N= 4 se imprime:

```
print("Ingresa un numero")
N = int(input())
i = 1
while i < N+1:
    x = 2**i
    print(x)
    i = i + 1
```

```
2
4
8
16
```

Debugging: print

Hacer debugging con print() es como hacer la tabla de ruteo, pero dejando que Python nos muestre los valores por pantalla

Instrucciones para debugging

Python

```
print("Ingresa un numero")
N = int(input())
print("N=", N)
i = 1
print("N=", N, "; i=", i)
while i < N+1:
    x = 2**i
    print("N=", N, "; i=", i, "; x=", x)
    print(x)
    i = i + 1
    print("N=", N, "; i=", i, "; x=", x)
```

Debugging: print

```
print("Ingresa un numero")
N = int(input())
print("N=", N)
i = 1
print("N=", N, "; i=", i)

while i < N+1:
    x = 2**i
    print("N=", N, "; i=", i, "; x=", x)
    print(x)
    i = i + 1
    print("N=", N, "; i=", i, "; x=", x)
```

```
Ingresa un numero

4
N = 4
N = 4 ; i = 1
N = 4 ; i = 1 ; x = 2
2
N = 4 ; i = 2 ; x = 2
N = 4 ; i = 2 ; x = 4
4
N = 4 ; i = 3 ; x = 4
N = 4 ; i = 3 ; x = 8
8
N = 4 ; i = 4 ; x = 8
N = 4 ; i = 4 ; x = 16
16
N = 4 ; i = 5 ; x = 16
```

type()

`type()` nos permite ver el tipo de una variable antes de hacer operaciones como comparar, concatenar, asignar, etc

Python

```
print("Ingresa un numero")
N = input()
print("N es de tipo", type(N))

if type(N) == str:
    N = int(N)

i = 1

while i < N+1:
    x=2**i
    print(x)
    i = i + 1
```

isdigit()

`isdigit()` nos permite ver si una cadena de texto tiene solamente números en ella

Python

```
print("Ingresa un numero")
N = input()

while not N.isdigit():
    print("Ingresa un numero")
    N = input()

N = int(N)
i = 1

while i < N+1:
    x=2**i
    print(x)
    i = i + 1
```

isdigit()

```
print("Ingresa un numero")
N = input()

while not N.isdigit():
    print("Ingresa un numero")
    N = input()

N = int(N)
i = 1

while i < N+1:
    x = 2**i
    print(x)
    i = i + 1
```

```
Ingresa un numero
hola
Ingresa un numero
chao
Ingresa un numero
43dfe
Ingresa un numero
4
2
4
8
16
```

isinstance()

`isinstance(variable, tipo)` nos permite evaluar si la variable corresponde un determinado tipo

Python

```python
num = input("Ingresa un numero: ")

while not num.isdigit():
    print("Error: ingresaste una cadena de texto")
    num = input("Ingresa un numero: ")

tipo_float = isinstance(num, float)

if tipo_float==True:
    print("verdadero")
else:
    print("falso")

print(type(num))
```

assert

`assert <condición>, <mensaje de error>` nos permite evaluar si la condición es verdadera. Si es falsa nos arroja un mensaje de error tipo assert

Python

```
num = input("Ingresa un numero: ")

assert num.isdigit(),'Error. Input no es entero'

# si no hay error, entonces el programa continúa su
ejecución después de assert.

# si hay un error, entonces el programa falla, y arroja un
error de tipo assert.

num = int(num)
print(type(num))
```

Ahora tú

Para los siguientes programar escribe la tabla de ruteo y luego escribe en idioma español el OBJETIVO de este programa

Python

```
print("Ingresa un numero")
x = int(input())
elegido = x
i = 0

while i < 4:
    print("Ingresa un numero")
    x = int(input())
    if x<elegido:
       elegido=x
    i = i + 1

print("El numero elegido es: ", elegido)
```

Ahora tú

Para los siguientes programar escribe la tabla de ruteo
Corrige los errores del programa

Python

```
print("Ingresa un numero entero")
x = int(input())

print("Estos son los amigos de tu numero:")
i = 1

while i < x+1:
    if x%i==0:
        print(i)
```