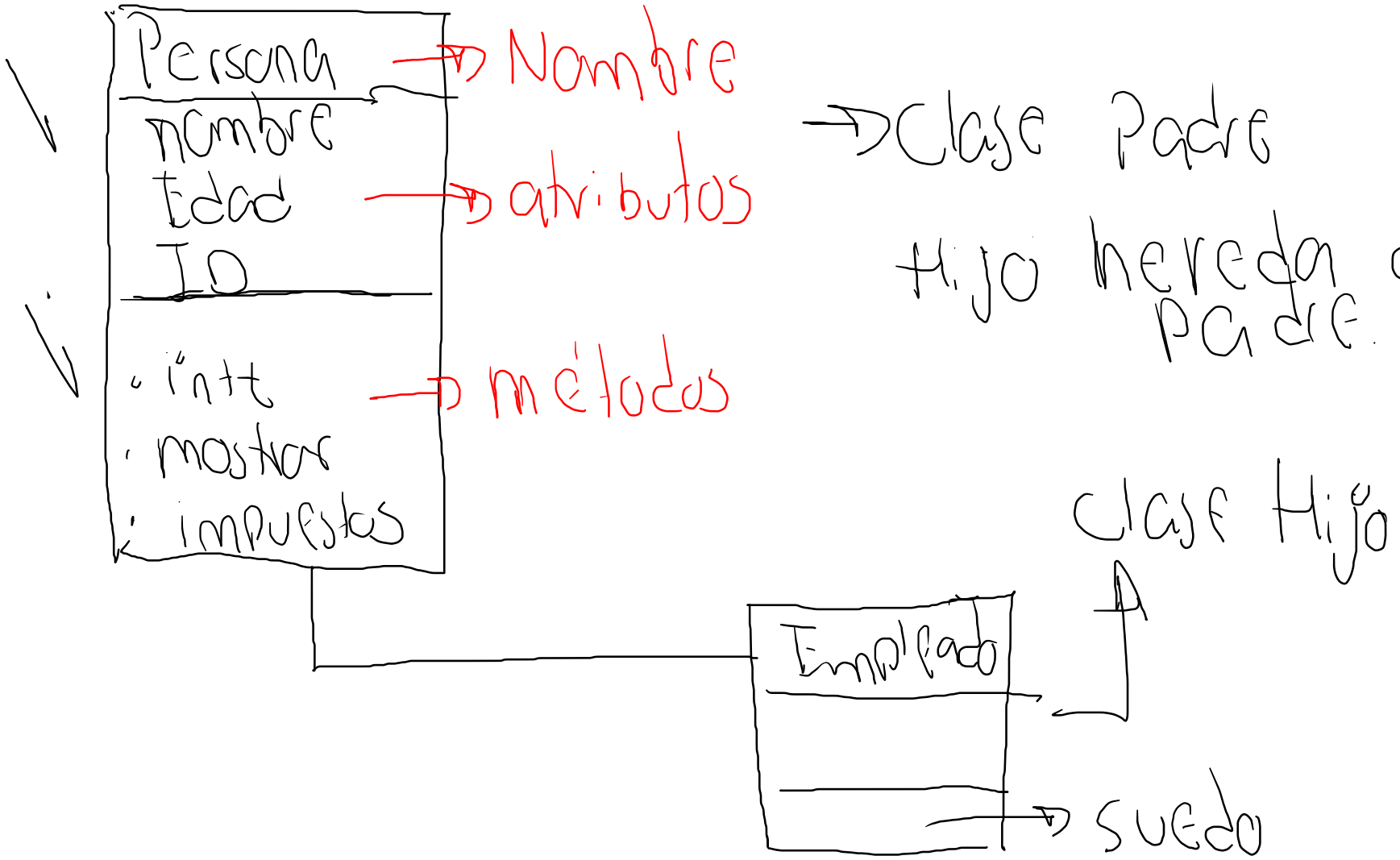
Sistema de Información



Repaso semana Santa

- Lunes → operadores, tipos datos, Algoritmos.
- martes → Estructuras de control: if, else, for, While.
- miercoles → Colecciones: tuplas, listas, Diccionarios
  Funciones: Recursión
- Jueves → (modulos y paquetes) (cadenas).
- Viernes → Objetos

* manejo (I/O) (ENUM)

operadores = + - * / % pow(x,y) = x^y

↓

$x^{**}y = x^y$

Logicos = and, or, not

relaciones = >, <, >=, <= !=

- tipos Datos = int, float, bool, str

Trucos = ingresar datos desde teclado input() str.

print() mostrar datos por Consola.

Variables → asignacion

id = "Valor" → valor, / dato: int, float, bool, str, lista, tupla, diccion.

identificador

operaciones logicas

AND

x_1	x_2	out
T	T	T
T	F	F
F	T	F
F	F	F

OR

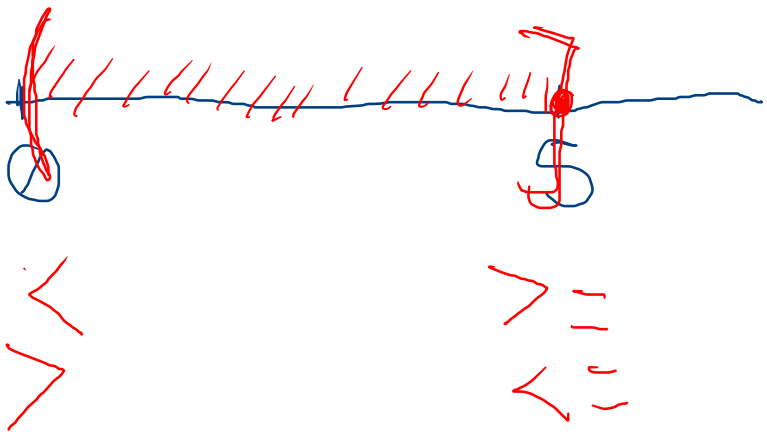
x_1	x_2	out
T	T	T
T	F	T
F	T	T
F	F	F

Intervalos

(0 - 5]

abierto

cerrado



a b c → frecuencia cardiaca

a → edad → int

b → peso → float

c → float

lista

memoria = [0 . . . . . 20]

10 11 12 13 14

Supuestos: 0 → memoria vacia

!= 0 → memoria ocupada

Random → función

$1/2/4$

$\frac{1}{2} : \frac{4}{1} = \frac{1}{2} : \frac{1}{4} : \frac{1}{8}$.

rad	Graus
π	$180°$
x	ângulo

$$x = \frac{\pi \cdot \text{ângulo}}{180}$$

1	3
2	6
3	
4	10
5	15
6	21
7	28
8	36
9	45
10	55

50
51
52
53
54
55

Estructuras de control

if ___

↳ condición True, False

```
if x > 10:
    ≡
```

```
if ___:
else:
```

```
if ___:
elif ___:
else ___:
```

Identificar caracteres

"H" → Consonante Mayuscula.
'o' → Vocal_minuscula.

[a-z] [A-Z] ⇒ Conjunto de Busqueda.

Vocales = [a, e, i, o, u] Vocales_M = ['A', 'E', 'I', 'O', 'U']

[B-C-D-F... z]

ASCII ⇒ Tabla.

Expresion Regular: → Encontrar patrones.

→ patron.

V_m : [aeiou]

V_mi = V_m

num: [0-9]

id = [a-zA-Z] ([a-zA-Z] [num])*

gramatica Regular

cb2

3b1 × No es valido.

Bucles / ciclos

for = para → for i in objeto_iterable:

While = mientras

≡

- range(inicio, terminar)
- range(ini, term, incr)

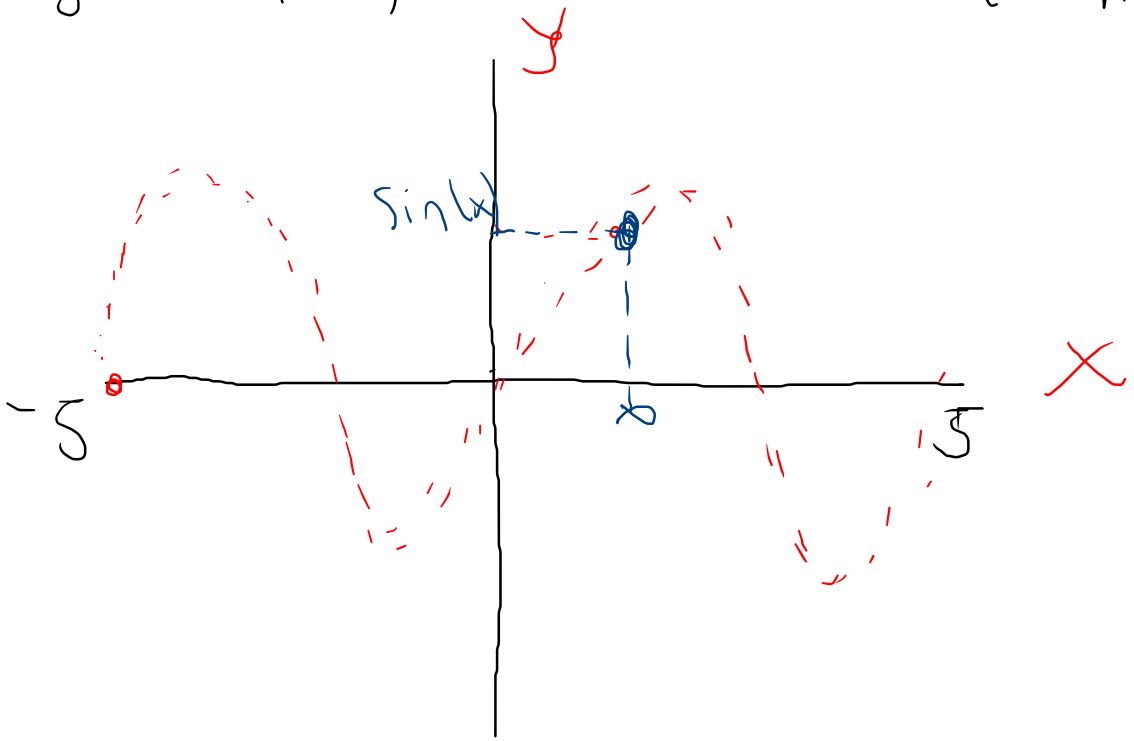
- str
- lista
- tupla
- diccionario

```
for i in l:
    print(i)
```

Grafica de una función trigonometrica.

$X = [-5, 5]$ # Dominio

$y = \sin(x)$ # Co-Dominio (Rango).



la raiz n-esima $\Rightarrow$ $\sqrt[2]{n} = n^{1/2}$ $\sqrt[5]{n} = n^{1/5}$

$n = [2 - 100]$ $\sqrt[3]{n} = n^{1/3}$ $\sqrt[6]{n^3} = n^{3/6} = n^{1/2}$

$\sqrt[4]{n} = n^{1/4}$

tomar numero = 3

$\sqrt[x]{n^3} = n^{3/x}$

$\sqrt[2]{3}$

$\sqrt[3]{3}$

$\sqrt[4]{3}$

$\sqrt[100]{3}$

for i in range(2, 100):

$\sqrt[i]{n^1} = n^{1/i}$

Algoritmo Euclides.

a b

a < b

swap (a, b)

a = b
b = a

aux = a
b = aux
a = b

While

True

While condicion :

```
cont = 0
While cont <= 10:
    print(cont)
    cont += 1
```

iteracio	cont	Print
1	0	0
2	0 + 1	1
3	1 + 1	2
4	2 + 1	3
.	.	
.	.	
.	.	
	10	10

Colecciones

→ Estructuras datos

- Tuplas : son inmutables No se puede cambiar
- Listas : son mutables
- Diccionarios : key:value.

- Guardar datos, Organizar datos, Procesar datos...
- objetos iterables

T = (1, 2, 3, 4, 5) Datos (0 1 2 3 4 Pos)

T[0] → 1 T[0]=4 ✗

L = [1, 2, 3, 4, 5] ⇒ L[0] → 1 L[0]=9 ✓

→ Guardar cualquier tipo de dato

Multiplicación de matriz

M(F,C)

l = [1, 2, 3, 4, 5]

l (1, 5)

l1 = (1, 5)

l2 = (1, 5)

A(m,n)

B(p,q)

A * B => n = p

l = [1, 2, 3, 4, 5]

l2 = [6, 7, 8, 9, 0]

l3 = [ ]

Sale del Range.

```
l1 = [1, 2, 3, 4, 5]      l2 = [6, 7, 8, 9, 0]      l3 = []

for i in l1:
    l3.append(l1[i] * l2[i])
```

2	*	7
3	*	8
4	*	9
4	*	

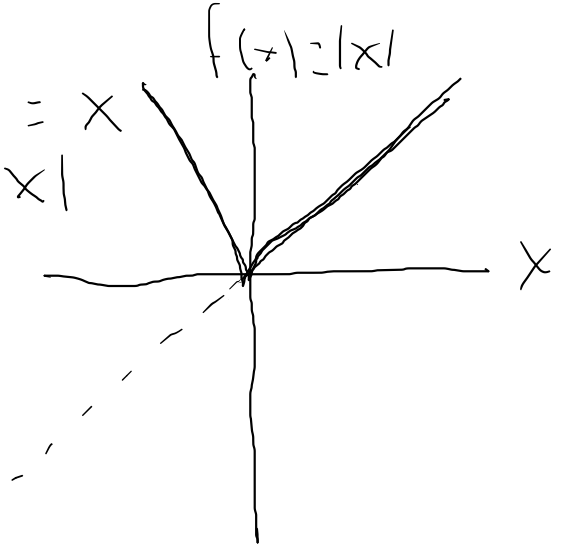
range(len(l1))

0 1 2 3 4 5 → idx

result = [ ] l = [1, 10, 2, 6, 2, 0]

1 - 10 = |-9| = 9
10 - 2 = |8| = 8
2 - 6 = |-4| = 4
6 - 2 = |4| = 4
2 - 0 = |2| = 2

abs(-x) = x
f(x) = |x|



abs(l[i] - l[i+1])

while cont < len(l)

result.append(abs(l[cont] - l[cont+1]))

cont += 1

Diccionario d = { key: value, key: value, ... }

key → 'str'
Int

value → str, int, float, bool, [ ], ( ) ... {}

SPO = { 'Nombre': —
tiempo: —
Genero: —
Artista: — }

→ Gusto musical
: más veces repro...
: más larg
:

SPO = { : 'Canción': [t, G, A, Num] }
:
:
:

Funciones

parametros

```
def nombre(x,y):
    ==
    return (x+y)
```

Manejo de errores

try:

exception → capturar el error

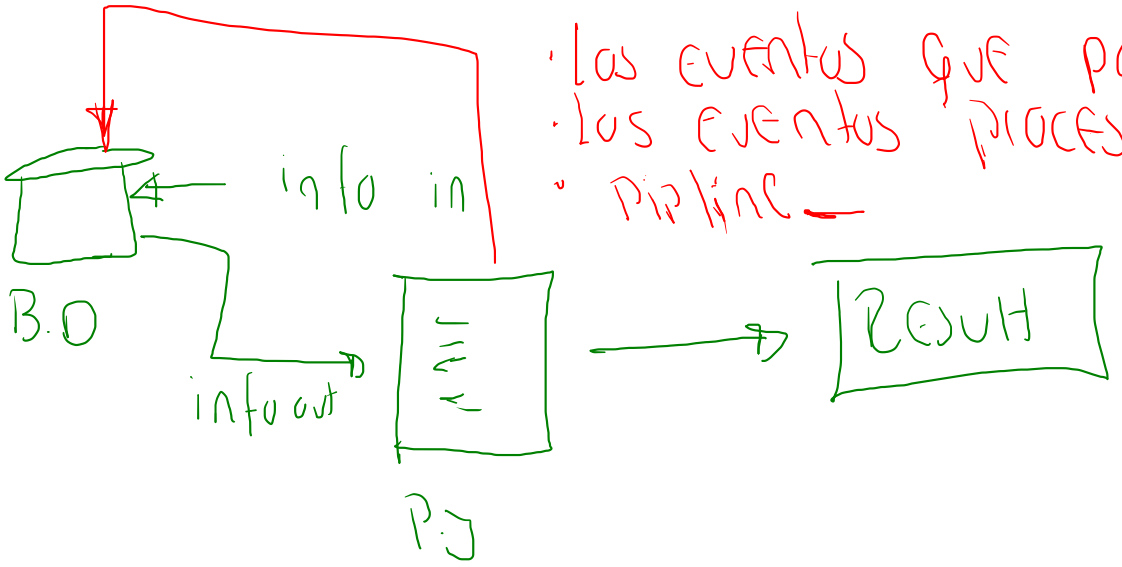
finally:

Lenguaje de Programación → Léxica → $, ~, ^,
Sintáctica
Semántica

Reglas gramaticales

```
fn incrementa(x){
```

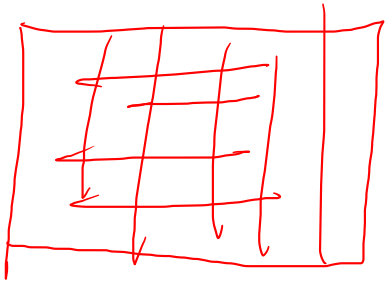
→ significado
- Error tipo
- Div / 0
- Index
- ⋮



- los eventos que pasan en la BD
- los eventos procesamiento
- Pipline

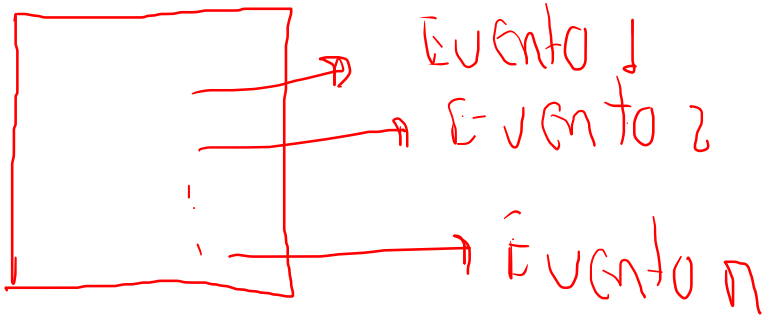
→ Dato malo
→ Dato malo
→
→ warning

Pre-Procesamiento



→ DataSet

logging.py



(mostrando información

P.O.O.

→ Abstracción en objeto

Objeto:
- Nombre
- atributos
- métodos

→

class Person()

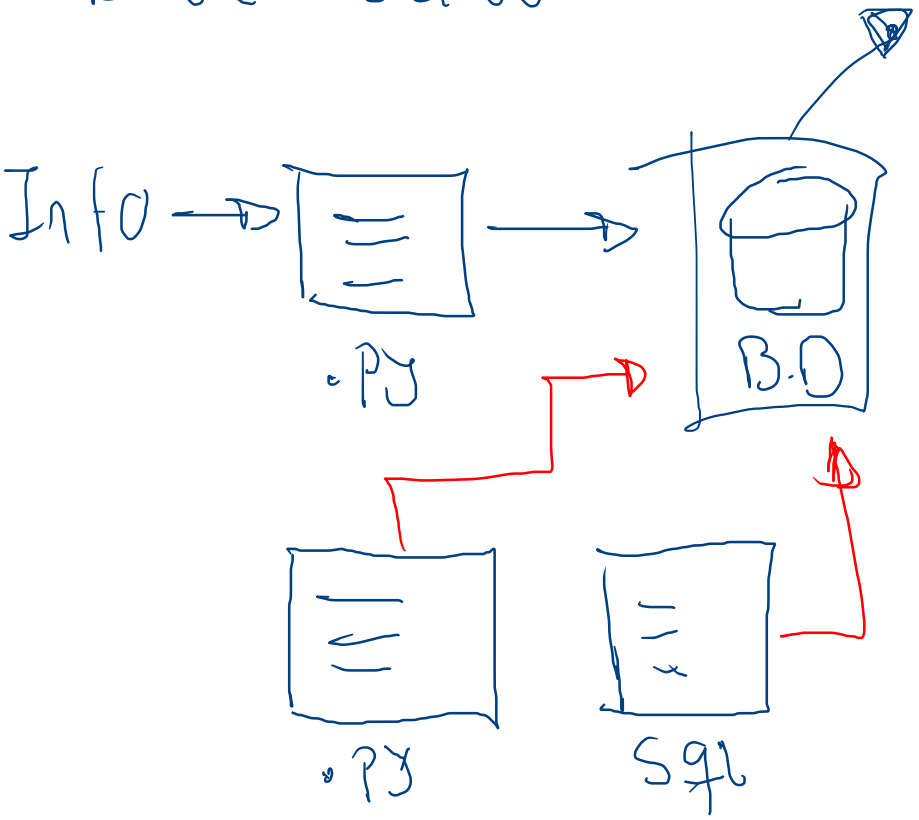
- Constructor __init__
- self.x (atributo)
- def foo():

Persona = Person() (Instancia de objeto).

Base Datos

Servidor Aplicaciones

//:local.host



- Create
- Read
- Update
- Delet

- SQL
- SQLite

- AWS ✓
- Google ✓ ($)
- Azure ✓