## *Primer Parcial Fundamentos de Informatica(PARTE TEORICA)*

*Introducción a Python*

## *Tema 1: Sintaxis Python; IDE (Entornos Jupyter like, VCode); Scripting;* Extensiones de ejecutables; Manejo de archivos

Software vs hardware:

* HARDWARE: parte fisica de la computadora
* SOFTWARE: phyton

→ software: interfaz entre la parte física y la maquina. la maquina habla en código binario para interpretar eso usamos IDE (entorno interactivo de desarrollo)

* ejemplo: VC (visual code), entorno interactivo donde vamos a estar desarrollando el programa

**Tipos de datos:**

1. números (float) → int
2. strings
3. booleano
4. array
5. dict

Len(string) String.upper() String.lower() String.count(“hola”)

String.replace(lo que quiero reemplazar, lo que quiero poner)

saludo = "Hola mundo"

saludo[0:3] me da lo que está entre eso (0,1,2) 'Hol'

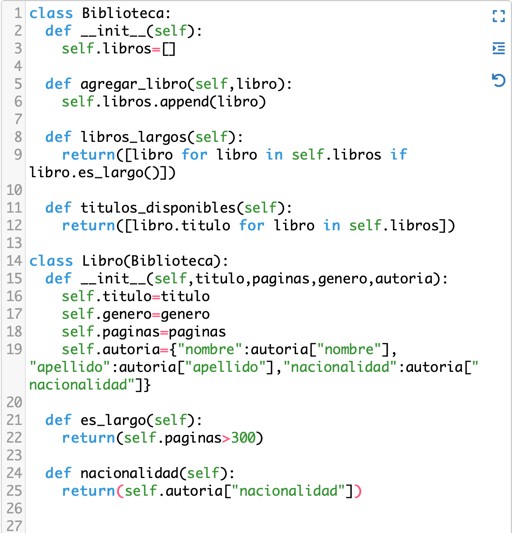
saludo[:3]me da **hasta** el que pongo 'Hol'

saludo[3:] me da lo que esta **después** de lo que pongo 'a mundo'

saludo[3] me da **exactamente** esa 'a'

Listas []

self.append() agrega algo de una lista self.remove()saca algo de una lista

self.index() accedo a la posición de lo que pongo adentro

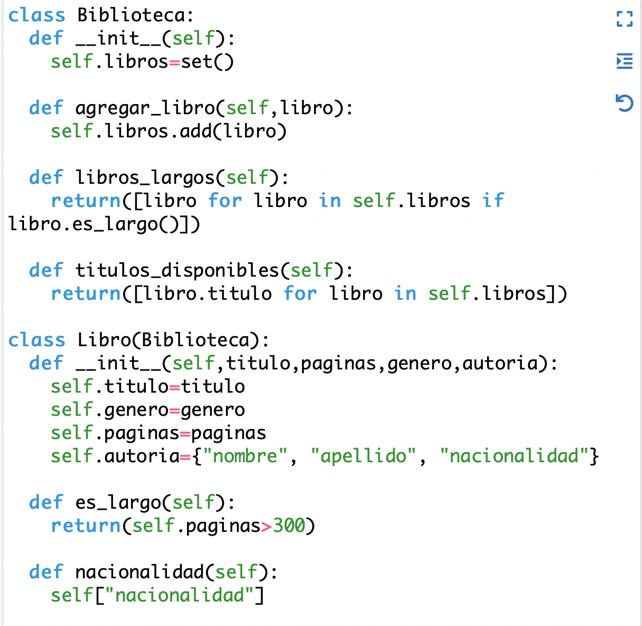
Diccionarios{}

Diccionario={“clave”: “valor de la clave”,”clave2”:”valor clave 2”}

Diccionario.keys() accedo a todas las claves del diccionario

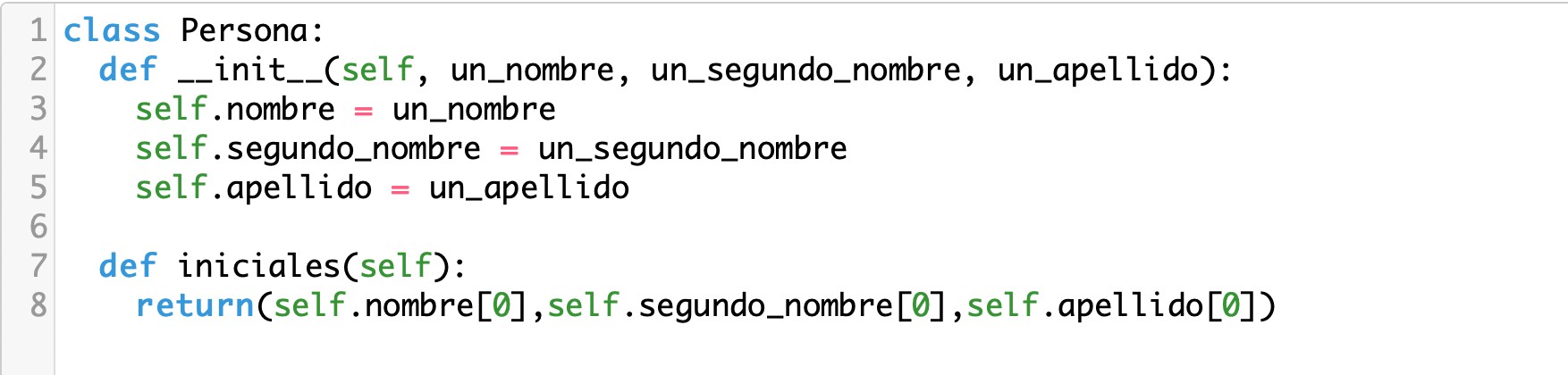
Diccionario[“clave”] me va a devolver el **valor** de esa clave

Diccionario[“clave”]=”valor” asi le doy el valor a esa clave

*Sets es como una lista pero NO tiene elementos repetidos*

*Self.add() Self.remove()*

*Tuplas NO las puedo modificar*

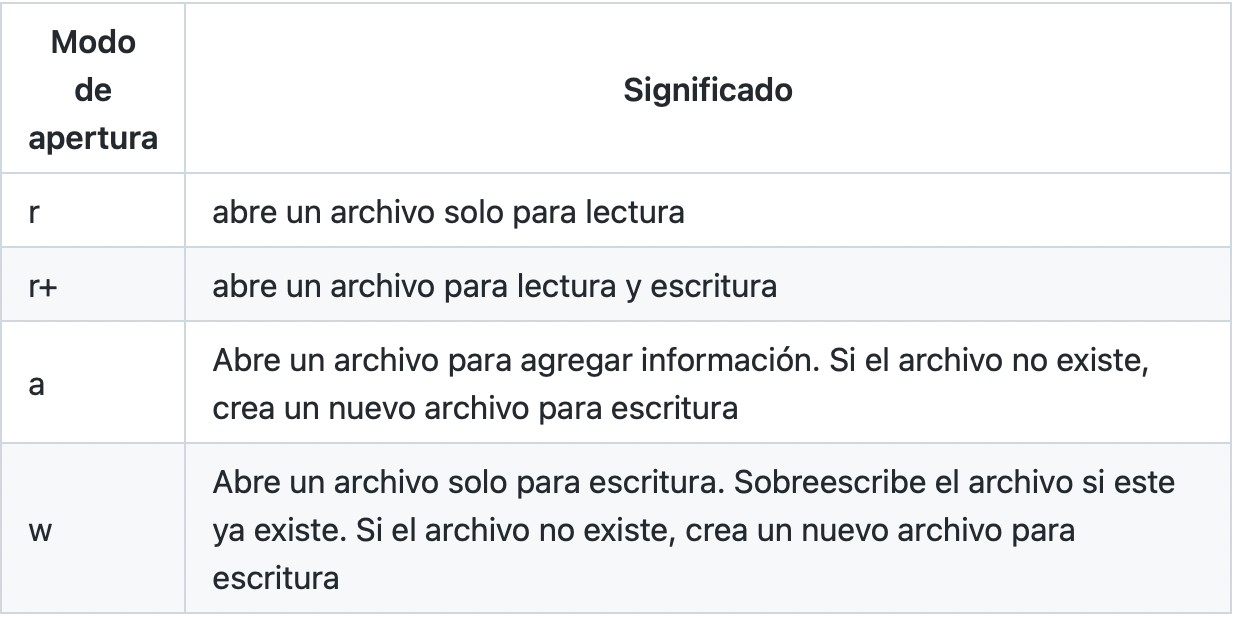


Archivo binario cualquier tipo de archivo que NO es txt With open (archivo,modo) as file:

Ejemplo de with open:

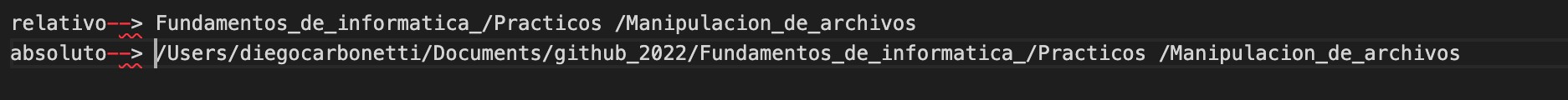
with open(path\_al\_archivo, modo) as arch: arch.write("Este es el contenido del archivo")

Archivo ruta dodne se encuentra mi archivo Modo como Python va a abrir mi archivo



Rutas absolutas es el recorrido de directorios y carpetas que recorro para llegar a mi archivo

Ejemplo: "/home/Facultad/Fundamentos/Manipulación\_de\_archivos.md" Relativa poner la última partecita



MODOS DE LECTURAS:

.read() lee todo el archivo

.readline() lee una linea

.readlines() lee líneas del archivo y las devuelve como lista

ATAJOS:

Command+J me abre la terminal

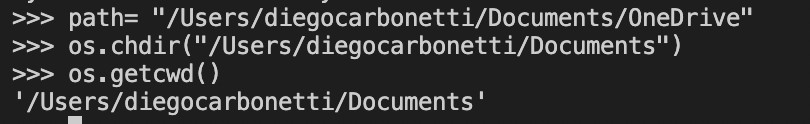
**cd** change directory

**cd.**  me dice donde esto parado

**cd..**  me voy una carpeta para atrás

**ls** → me lista los archivos que tengo en la carpeta

**pwd** (en Python)→ working directory (carpeta donde estoy parada)



BIBLIOTECAS:

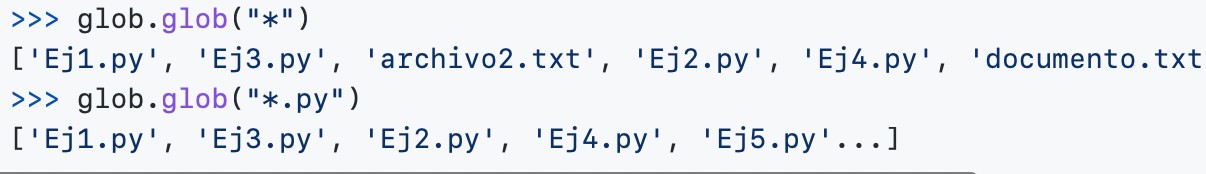
**Import os**

* os.chdir(path) change directory
* os.listdir() me dice todos los archivos que tenga en la carpeta
* os.mkdir(argumento) toma como argumento un nombre crea una carpeta
* os.path.join(“capreta1”, “carpeta2”,”archivo”)me genera la ruta solo escribiendo los nombres

**Import glob**

Glob.glob() listo archivos específicos

\* cualquiera Ejemplo:



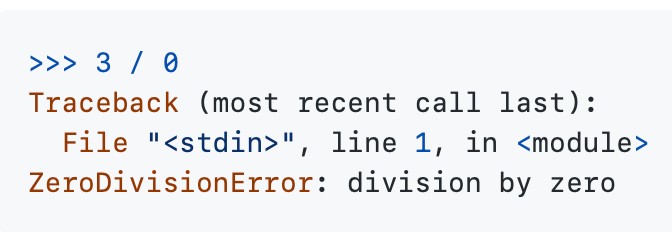
**Ejercicio:**crear un string que arme la carpeta "datos\_personales" en users y que guarde alli un archivo con tu nombre apellido y edad.

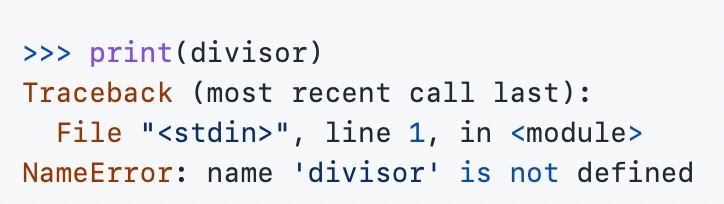
1. os.getcwd() asi veo si estoy en users (si no estoy hago un os.chdir("/Users/diegocarbonetti/Documents/datos\_personales")
2. os.mkdir("datos\_personales") asi me crea la carpeta
3. os.chdir("/Users/diegocarbonetti/Documents/datos\_personales) asi me ubico donde quiero que se guarde el archivo
4. lo nombro como path2=os.getcwd()
5. with open ("datos\_mios.txt","w") as mi\_archivo: mi\_archivo.write("los datos que quiero")
6. le agrego al path mi nuevo archvio asi se ubica ahi dentro path2= os.getcwd() + "//datos\_mios.txt")

## *Tema 2: Manejo de excepciones; Expresiones Regulares; Uso de biblioteca RE* Manejo de Excepciones Python:

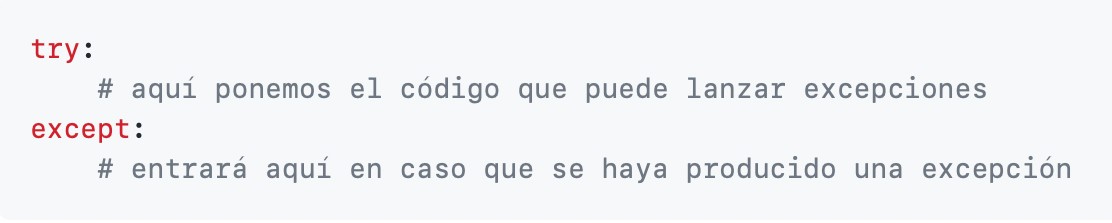
***Anticiparnos a los errores***

* 1. Errores de sintaxis tipo de error que tiene que ver con “escribí mal”
  2. *TypeError*



* 1. *ZeroDivisionError dividir por cero*
  2. *NameError no definí algo y la maquina no entiende lo que le paso*

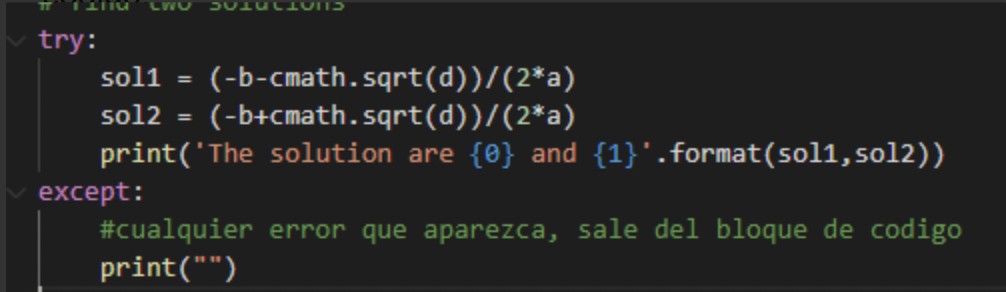
*FORMAS DE AGARRAR EL ERROR*

1. *Try-Except el try* ***intenta agarrar*** *el error. Exept dice que si falla ejecute esto. Entonces agarro el error pero sigue ejecutando y programando cosas(sino en Python no deja seguir*

*ejecutando se corta todo)*

*Por ejemplo no puedo dividir por cero y si ponen como a=0 se cortaría*

*todo entonces cuando a=0 va a ejecutarse el print(“ ”)*



1. *If-raise:* ***creo*** *una excepción personalizada cuando pasa algo determinado. (levanto un error)*

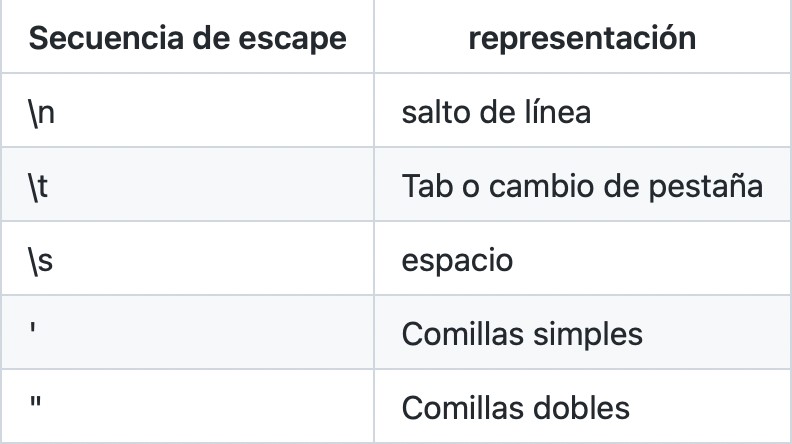
*Ejemplo: quiero saber si lo que le paso a la comptadora es un string Def es\_string(algo):*

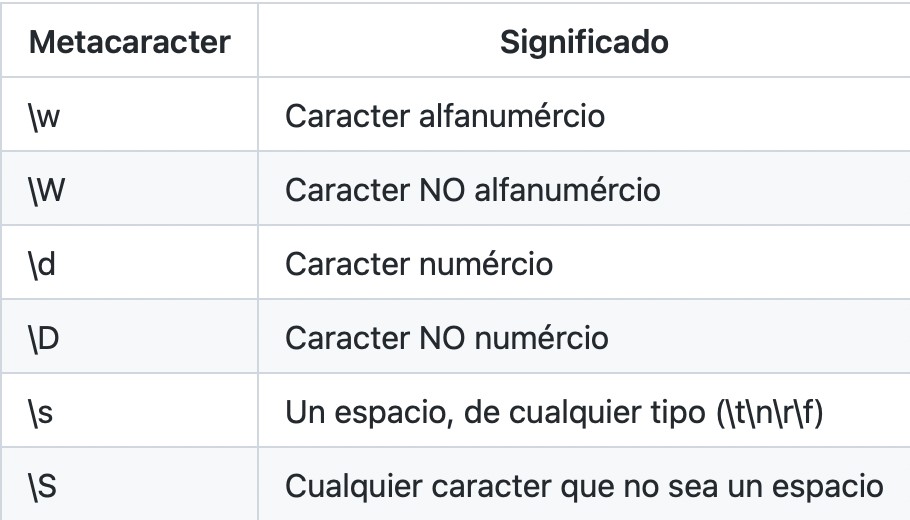
*If trype(algo)!= str:*

*Raise TypeError(“esto no es un string*

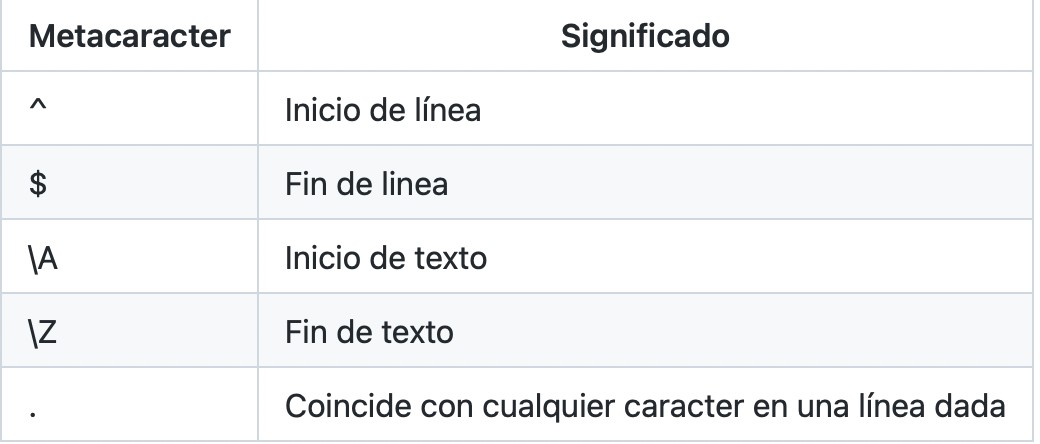
## *Expresiones Regulares*

* + Permite detectar formatos de texto especiales
* Secuencias de escape: carácter especial que determina tipo de formato

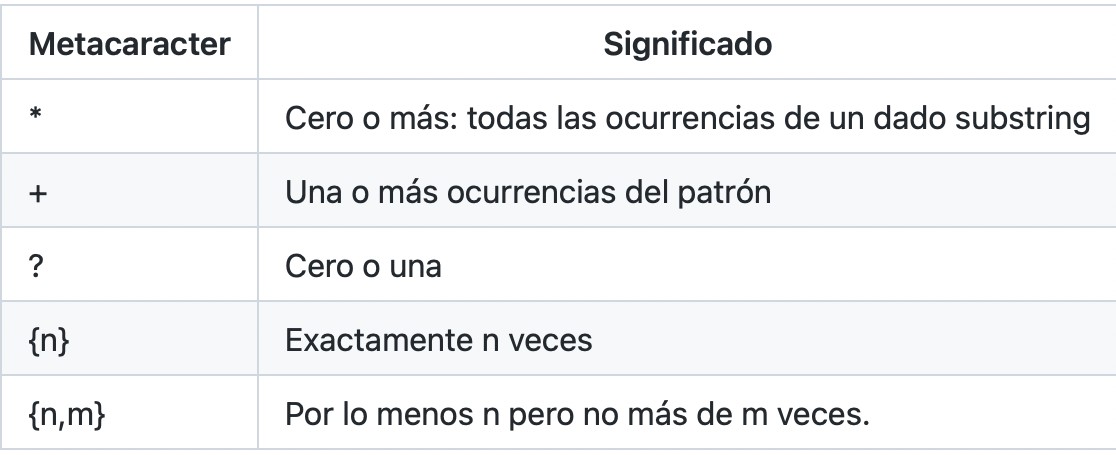


* Expresiones regulares conjunto de caracteres que determina **fragmento especifico** del texto
  + Puedo buscar cosas especificas dentro del texto
* Meta caracteres  carácter especial con significado especial para expresiones regulares.

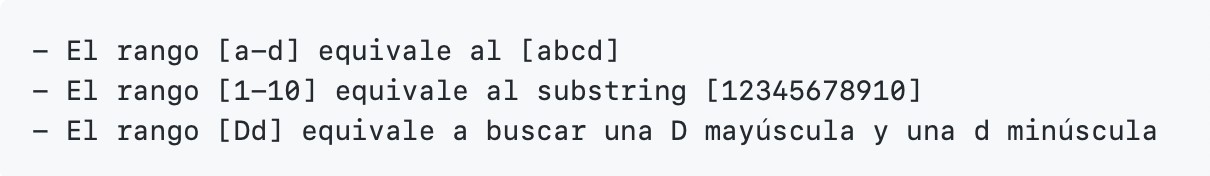
1. Delimitadores  delimita **donde** quiero buscar el patrón.

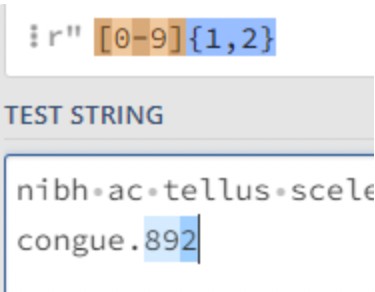


1. Cuantificadores cuantas veces quiero que se repita esa búsqueda



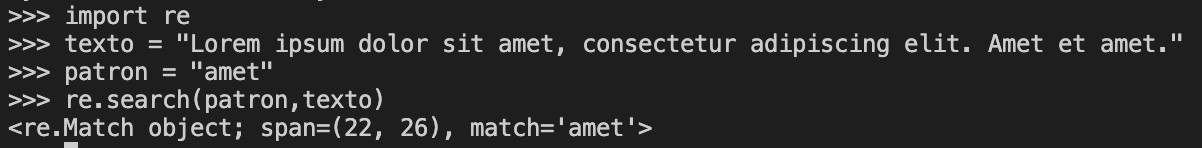
Rangos:



Pongo entre corchetes el rango y después que busque números de mínimo 1 y máximo 2 caracteres.

Listar caracteres que **NO** deben aparecer( ^). Si

pongo rango [^a-d] me da cualquier cosa que **NO** sea abcd. BIBLIOTECA RE:

1. Match sirve para el **comienzo de la sentencia**
2. Searchsirve para **todo** el texto (muestra SOLO la primer aparición y en que linea esta). **AL MENOS UNA VEZ**
3. Search.group() me trae el objeto en si (solo la palabra)



1. Findall separa y devuelve apariciones en forma de lista. **TODOS** coinciden



Ejemplo: obtener lo que se encuentre entre las palabras “ipsum” y “sit”

1. Group(1)Obtener el **substring** que este dentro de las palabras Group(0) nos dio la coincidencia

Group(1) nos dio lo que esta dentro de eso (un substring)

Span() me devuelve en que posiciones

1. sub reemplazo las

ocurrencias del patrón por otro patrón en un string

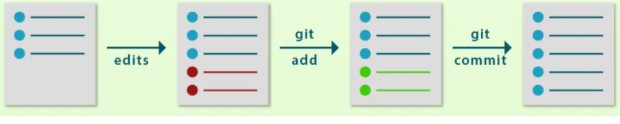


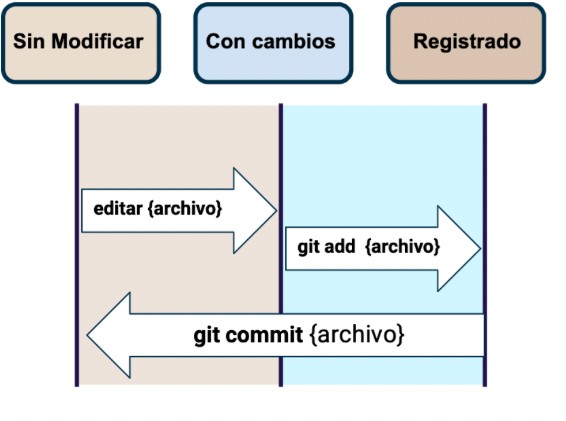
## *Tema 3: Control de versiones (GitHub)*

Control de versiones guarda copias de tu código (distintas versiones) y podes ir para atrás para ver de dónde está el error.

File Clone Repository URL pongo el code de github me clona el repo de la persona y puedo verlos dentro de git.

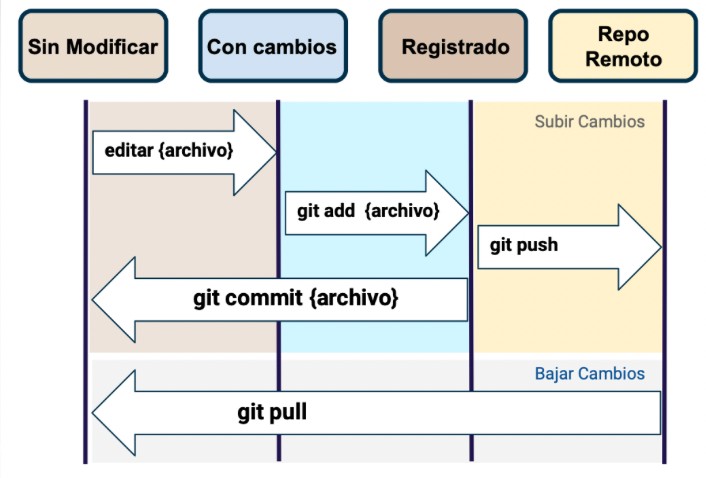
Sistema de control de versiones herramienta que realiza seguimiento de los cambios en un documento (crea distintas versiones de mi archivo)



1. edito mi archivo
2. git add agrego lo que edite (mi archivo genero cambios) DENTRO DE MI COMPUTADORA
3. git commit aca queda regsitrado mi cambio. Todos mis commits forman un repositorio (historial de cambios que hice en el archivo)

**recién cuando hago el push me conecta con git hub**

git local mi computadora ve los cambios

git remoto tengo que subir mis cambios (commits) y compartirlos (git push)

## *¿Cómo hago eso?*

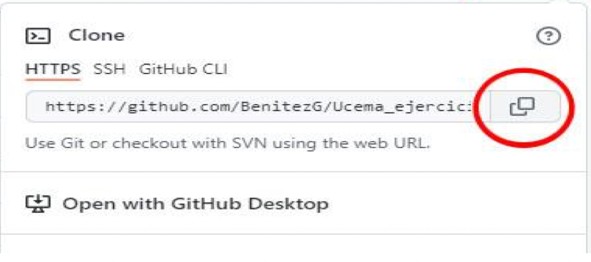
* + Git push **envía** mis commits que genere localmente (lo pueden ver)
  + Git pull me descargo los **NUEVOS**

cambios del repositorio

* + Git clone me descarga **TODO**

Como voy a buscar el archivo para el parcial:

1. Creo carpeta “Parcial”
2. Git pull “https del repo que nos pasen”
3. Dentro de la carpeta UN archivo por ejercicio
4. Termino el parcial hago git push

Como abrir un repositorio:

1. Clono el http
2. Abro carpeta donde quiero que este mi repo clonado (github\_2022)
3. Hago **git clone** “https del repo”

**Ahí se generó la carpeta y me tengo que mover a esa carpeta para empezar a trabajar ahí dentro.**

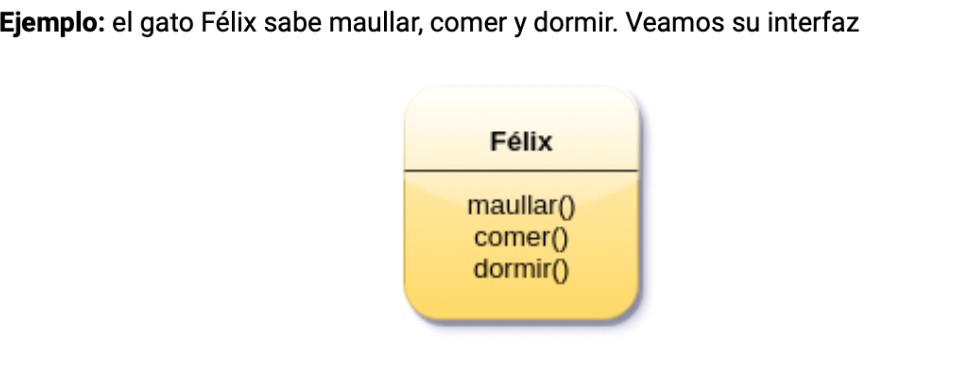
**Como agrego cosas:**

1. **Git add \* (agregame todo lo que no tenga cambios)**
2. **Git commit -m “mensaje de lo que hice”**
3. **Git push (hice cambios localmente (vscode) y los envío para que los vean)**
4. **Git pull (para descargarme los cambios que hice desde la nube)**

*POO (programación orientada a objetos)*

## *1. Objetos, clases, instancias, atributos, métodos. Estado e interfaz. Herencia y* polimorfismo.

Objetos ente computacional que me comunico mediante **mensajes. Le puedo enviar mensajes que forman la interfaz (el conjunto de mensajes que entiende este objeto).**



**OBJETOS DE UNA MISMA CLASE DEBERIAN COMPARTIR LA INTERFAZ (Feliz es un gato,**

**Mirta es otro gato ambos deberían compartir la interfaz).**

**Interfaz parcial** **entienden algunos mensajes iguales dos clases distintas**

* Polimorfismo **dos o más clases** son **polimórficas** si comparten la interfaz, es decir, **pueden responder a un mismo mensaje** enviado desde una **tercera** clase **aunque el resultado sea distinto.**

**Que dos clases sean iguales NO implica que sean polimórficas, son polimórficas cuando EXISTE una tercer clase que les manda un mensaje y ambas lo entienden aunque la respuesta sea distinta**

# PARA QUE EXISTA POLIMORFISMO TIENE QUE EXISTIR OTRA CLASE QUE LE MANDE LOS MENSAJES MAS ALLA DE QUE COMPARTAN METODOS SI NO EXISTE CLASE MADRE NO SON POLIMORFICAS!!!!

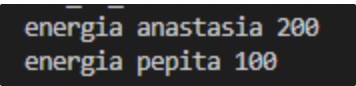
 **la tercera clase NO necesita un init (si no tiene atributos no instancio)**

* + **Cada objeto es una INSTANCIA de la clase porque tienen una IDENTIDAD.**
  + **Instancia** **crear al objeto (darle vida a algo abstracto)**

**Ejemplo: en la clase Golondrina (modelo abstracto) xq es una descripción general del objeto (sé que tienen distintas funciones las describo en general NO a una particular). Pepita es una INSTANCIA (le doy vida), llame a la clase y le di los valores para nacer.**

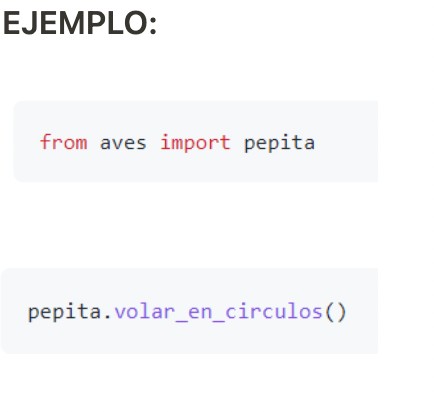
**Pepita= Golondrina(100) Ejemplo:**

**Distinta identidad** ya que pepita y anastasia tienen estados distintos.

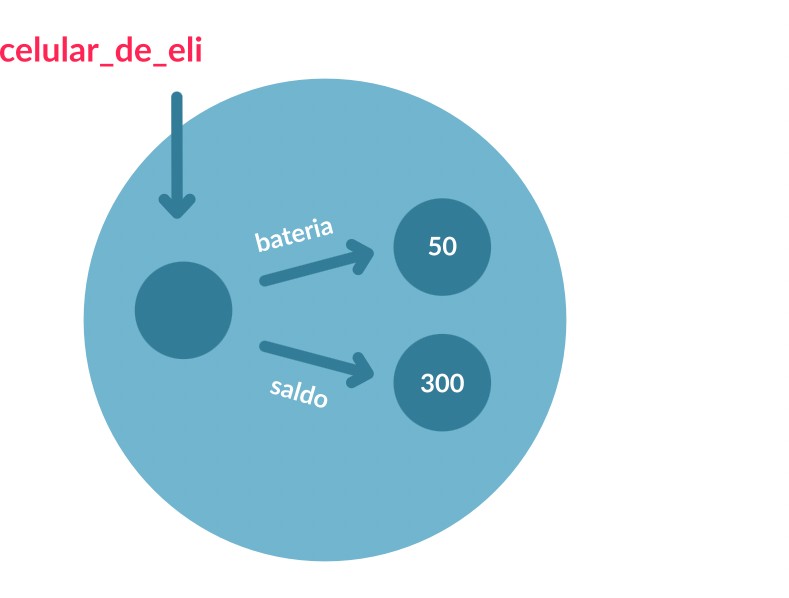


* + Si comparo dos instancias de la misma clase (pepita y juanita) que tienen el mismo estado inicial (100) si hago print(pepita==juanita) me da False,

porque no son la misma variable (no se almacena de igual manera en la memoria). Si imprimo estos objetos print(juanita), print(pepita) me da un número de objetos distintos (como el dni).



Estado del objeto **conjunto de atributos** del objeto **estado estático.**

* + **El estado estático es el init**  **CONSTRUCTOR DEL OBJETO DENTRO DEL ESTADO PUEDO TENER VARIOS**

**ATRIBUTOS(lo que valen los estados(50,300 en este celu especifico) PERO EL ESTADO ES UNO (batería y saldo para todos los celus)**

* Los estados van cambiando a medida que lo hago hacer cosas **COMPORTAMIENTO DEL OBJETO.**
* Estado dinámicocontempla **valores** concretos de cada atributo.
* **El estado dinámico son las instancias** **le doy entidad al objeto**

Ambiente lugar donde viven los objetos.

* + En un ambiente pueden estar varios objetos (pepita y anastasia)

Mensajes cosas que le puedo pedir al objeto que haga  **ES UN METODO**

## *objeto.mensaje(argumentos)*

llamo al objeto (pepita en este caso)

devuelve NONE ejecuto la acción pero no dice nada entonces sé que **TIENE EL ATRIBUTO** xq entendió el mensaje

# SI EL OBJETO CONOCE EL MENSAJE QUE LE ENVIAMOS NO TIRA ERROR PERO NO SIEMPRE TIENEN QUE RESPODNERTE ALGO.

**Tiro ERROR** entonces sabemos que este mensaje no lo entiende



Métodos **NO** se crean funciones sino que decimos **METODOS** ya que están **dentro de la clase** y siempre tienen como **primer parámetro self**.

* + lo que ejecuta el objeto por el mensaje que yo le envié

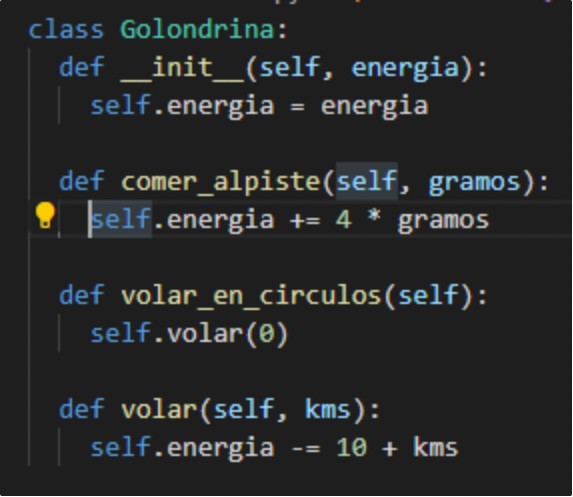
**CLASES DE OBJETOS** dan vida a los objetos y si los objetos son de la **misma clase** se comportaran de igual forma (entienden mismos mensajes)

Clases dan vida a los objetos y si los objetos son de la **misma clase** se comportan de igual forma.

# LAS CLASES TIENEN QUE SER DEFINIDAS EN SINGULAR YA QUE REPRESENTAN UNA COSA

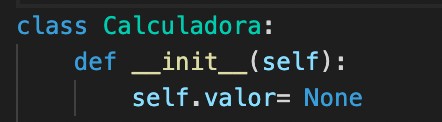
Self instancia de la clase (el objeto especifico que quiero analizar)

# COMO SE ESCRIBE UNA CLASE:



1. Nombre de la clase con mayúscula “Golondrina”.
2. Los mensajes que reconoce cada objeto **METODO**
3. **init**  **NO** es un método **ES CONSTRUCCION** (con esto doy forma al objeto) lo que está dentro del init forma el **ESTADO** del objeto.

**OJO PUEDO TENER UN INIT SIN ATRBUTOS (ponerle None)**



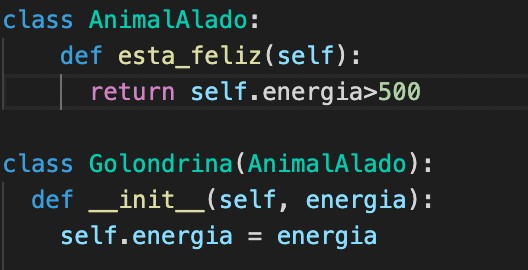
# COMO CREO UN OBJETO:

Le pongo nombre y le asigno el estado inicial

Roberta tiene **dos atributos** dentro de su estado.

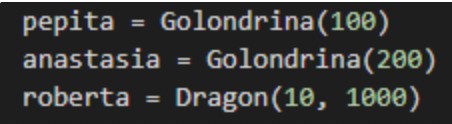
## *Herencia*

* Puedo crear una clase (clase madre) donde salen otras clases que tienen

**MISMO** método. (AnimalesAlados)

* Clase madre “hereda” cosas a las clases hijas (golondrinas y dragón)
* Las clases madres **NO** se **instancian** son **ABSTRACTAS** (no les doy vida nombrándola como a la clase Golondrina con pepita)
* Las clases que heredan los métodos hacen LO MISMO (no es como en polimorfismo que el

mensaje era distinto solo que entendían el concepto en herencia ambos hacen lo mismo)



Metodo getter permite acceder a los estados del objeto



Metodo setter modifica el atributo

**CUANDO CREE UN INIT EN LA CLASE MADRE NO PUEDO VOLVER A CREAR UNA EN LA CLASE HIJA TENGO QUE USAR UN SETTER**

