<https://www.youtube.com/watch?v=nKPbfIU442g>

PYTHON

* Puede ser considerado el lenguaje #1 de propósito general
* Es un lenguaje de alto nivel es decir que se parece mucho al lenguaje natural
* Es de tipado dinámico es decir que el mismo lenguaje identifica el tipo de dato y lo define
* Es un lenguaje orientado a objetos
* Es un lenguaje interpretado
* Cuenta con una gran biblioteca estándar <https://docs.python.org>
* Se puede mesclar con otros lenguajes de programación
* Comentarios con **#**
* Se separa palabras con **snake\_case** ej. nombre\_completo
* Para clases notación pascal (HolaMundo)
* Variables métodos y funciones snake\_case
* Validar que tipo de dato es se usa **type() => print(type(variable))**

23:05

**Tipos de Datos simples**

* Texto-string: Se usan las comillas
* Para una línea de texto

“string”

‘string’

* Para varias líneas de texto

“””string”””  
‘’’string’’’

* Números
* Enteros INT
* Flotantes FLOAT
* Boolean: Primera con MAYUSCULA – True ó False

**Tipos de Datos compuestos**

* Son datos que tienen adentro mas datos
* Lista = [dato1, dato2, dato3] – Se puede modificar
* Tupla =(dato1,dato2, dato3) – No se puede modificar
* Crear tupla con tupla**= tuple**([“dato1”,”dato2”])
* Crear tupla con tupla=“dato1”,”dato2”
* Las tuplas se usan para lectura de datos fijos que no se van a modificar
* Crear tupla de un solo dato tupla= “dato1”**,**
* Conjunto ={“dato1”, “dato2”, “dato3”}
* Conjunto =set([“dato1”, “dato2”, “dato3”])

– Se puede reconstruir pero no se puede cambiar cada elemento

* no se puede acceder al índice
* no permite repetir valores
* Se puede meter un conjunto dentro de otro conjunto   
  cinjunto1 = **frozenset**([“dato1”, “dato2”)]  
  conjunto2 = {conjunto1, “dato3”}
* **Conjunto.issubset(conjunto2) ó conjunto <= conjunto2:** permite saber si conjunto es un subconjunto de conjunto 2
* **Conjunto.issuperset(conjunto2) ó conjunto >= conjunto2:** permite saber si conjunto es un super conjunto de conjunto 2
* **Conjunto.isdisjoint(conjunto2):** Permite validar si 2 conjuntos son totalmente diferentes, con que exista 1 solo valor = el valore de respuesta seria False
* diccionario {

‘nombre’: “Victor”,

‘apellido’: “Garcia”,

‘altura’: 1.85

}

* En vez de tener un índice tiene una clave o key y un valor o value
* **Diccionario= dict**(nombre= “victor”, apellido= “Garcia”)
* **Diccionario. Dict.fromkeys([“key1”, “key2”, “key3”], valor\_por\_defecto):** permite crear diccionarios solo con claves y con valores vacíos
* El valor por defecto puede usarse o no dependiendo la necesidad, si no se una se asigna a la key el valor none

**Variables**

* Son datos que se almacenan en la memoria de nuestro programa
* Las variables se declaran se definen y se pueden modificarteponiendo **del variable**
* Se borra la variable an

**Variables.desempaquetado:** Se refiere a asignar los valores de una secuencia a variables individuales mediante una instrucción

* Las variables de la secuencia de ubican al lado izquierdo
* Los valores de cada variable se ubican al lado derecho de la secuencia.
* Funciona para tuplas, listas y conjuntos
* Secuencia con valores mi\_lista = [1, 2, 3]
* Secuencia asignación de variables a, b, c = mi\_lista
* print(a) # Imprime 1
* print(b) # Imprime 2

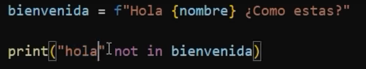
print(c) # Imprime 3

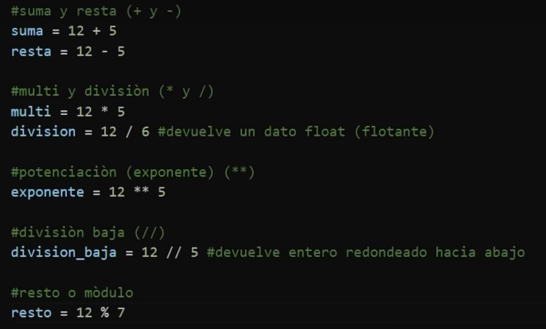
**Concatenar**

* Se concatena strings usando **+**
* Se concatena convirtiendo a string cualquier dato usando **F“string {variable} string”**

**Operadores de pertenencia**

* Devuelve **True o False** con el uso de **in** ó **not in**



**Operadores aritméticos**

**Condicionales**

* Son fracciones de código que devuelven true o false y dependiendo el resultado se ejecute una u otra fracción de código
* **If, elif, else**

**Operadores lógicos**

* And (**&**), or(**|**), **not**

**Métodos**

* **dir :** Devuelve la lista de atributos validos para un objeto sea cadena, numero, lista
* Son funciones aplicadas a objetos
* Nombre\_de\_la\_funcion (parametros)
* Cuando es un método se ejecuta como nombre\_variable.metodo()
* Todos los métodos son funciones pero no todas las funciones son métodos

**Métodos de cadenas**

* string.**upper() :** Convierte el string en mayúscula
* string.**lower() :** convierte todo a minúscula
* string.**capitalize():** Primera letra en mayúscula y el resto en minúscula
* string.**find(busqueda):** Busca por posiciones de letras la letra que ingresemos como parámetro  
  - Cuando no encuentra la letra devuelve -1
* string.**index(búsqueda):** Busca por posiciones de letras igual que el find pero cuando no encuentra nada lanza una excepción
* string.**isnumeric():** Validan dentro del strign si el valor ingresado es numerico
* string.**isalpha():** Valida si el valor el alfanumerico pero no contempla caracteres especiales incluido el espacio
* string.**count(búsqueda):** Busca el numero de coincidencias de una buscada como parámetro en un string palabras o letras
* **len(**string**): es una función** cuenta la cantidad de caracteres en una cadena
* string.**startswith(búsqueda):** Devuelve true o false si la cadena inicia con la búsqueda ingresada como parámetro
* string.**endswith(búsqueda):** Devuelve true o false si la cadena finaliza con la búsqueda ingresada como parámetro
* string.**replace(parámetro antiguo, parámetro nuevo):** si encuentra una coincidencia en el parámetro 1 lo reemplaza por el valor ingresado en el parámetro 2
* string.**Split(): S**epara la cadena que le pasamos como una lista separando el string por palabras

**Métodos de listas**

* Se puede crear una lista vacia con la función **list()**
* Se puede crear una lista definiendo **variable = []**
* **Len(lista):** es una función, nos devuelve la cantidad de elementos que tiene la lista
* Lista.**append(“Nuevo elemento”):** Método que agrega un elemento al final de la lista
* Lista.**insert(posición, “nuevo elemento”):** inserta nuevo elemento en la posición que le indiquemos
* Lista.**extend([nueva lista]):** Agrega una lista a otra lista
* Lista.**pop(indice):** Elimina el elemento que se ingrese como indice en el parametro, si se ingrea como parametro (-1) elimina el ultimo elemento
* Lista.**remove(“elemento”):** Elimina el elemento que se indique como nombre en el parámetro
* Función **del** Lista [indice]: elimina la lista
* Lista**.clear():** Elimina todos los elementos de la lista
* Lista**.sort():** Organiza todos los elementos de menor a mayor no aplica para strings
* Lista.**sort(reverse=True):** Organiza los elementos de mayor a menor, no aplica para strings
* Lista.**reverse():** Ordena los elementos de la lista en forma contraria, peor no de mayor a menor sino en el mismo orden que estan originalmente.
* **Comprehension de listas** [print (x) for x in Lista]

**Métodos de diccionarios:**

* Diccionario.**keys():** devuelve una lista con las claves del diccionario
* Diccionario.**items():** devuelve una lista con las claves-valores del diccionario
* Diccionario.**get(key):** trae en valor de propiedad que corresponde a la llave, la diferencia de usar diccionario[key] es que esta ultima me trae un error de excepción.
* Diccionario.**clear():** elimina todo el diccionario
* Diccionario.**pop(key1, key2):** Elimina la clave que se indique

**Inputs:**

* Función que permite pedirle un dato al usuario
* El dato que nos devuelve input siempre es in string
* Para convertir a numero el input obtenido se usa **int(variable)** ó **float()**

**Bucles:**

* Repetir de forma controlada una sección de código
* Un elemento es iterable cuando tiene algo que defina como se puede iterar
* Un iterador es un elemento que se crea para seguir métodos iter() y next()
* **Bucle FOR**
* En Python se usa **for** elemento **in** objeto
* Se pueden iterar 2 listas al mismo tiempo siempre y cuando tengan la misma cantidad de elementos

Nombres =[“pepe”, “juan”, “Jaime”]

Edades =[30,40,20]

For nombre, edad **in zip**(Nombres, Edades)

Print (f“se llama {nombre} y tiene{edades}

* For num **in range(numero inicio, numero fin):**  
   print(num)

Iterar en range nos permite hacer iteración entre los números que se ingresen en el rango, si solo se pone un parametro arranca de cero hasta el numero del parametro

* For índice, iterador in **enumerate(lista):**

Print(f”el índice es {indice} y el valor es {iterador}”)

La función enumerate permite acceder al índice y traerlo como impresión

* For iterador in lista:

Print(f“el valor es {iterador}”)

**Else:**

**Print**(“el bucle termino”)

El else después del formsiempre se ejecuta

* For key in **diccionario.items():**

Print (key)

Sirve para recorrer diccionarios

* For iterador in lista

**If iterador == ‘excepción’:**

**Continue // break**

Print(f“texto {iterador}”)

Continue Permite crear condiciones de excepción y que no se muestre en el recorrido de la lista la excepción condicionada pero se termina de recorrer el elemento.

Break termina el recorrido del bucle cuando se encuentre la excepción.

* **Bucle while**
* Se ejecuta siempre que se cumpla una condición, a diferencia del for que recorre elementos

**Funciones:**

* Códigos que ya estan creados y que podemos reutilizar para no repetir código, además hace el programa fácilmente modular y permite el fácil mantenimiento gracias a su lectura
* **Funciones biuldin:** Son funciones ya integradas con Python que podemos usar.
  + **Max()**: ayuda a encontrar el numero mas alto de un iterable
  + **Min():** ayuda a encontrar el numero menor en in iterable
  + **Roud(numero, decimales):** redondea números y agrega los decimales que necesitemos
  + **Bool(valor):** devuelve false cuando se pasa vacio o falso, si se pasa un valor diferente a cero o vavio devuelve True
  + **All(valor):** chequea que todos los valores sean true, si uno es False devuelve False
  + **Filter ()**
* **Funciones propias:** se crean funciones con **def nombre\_de \_la\_función ()** 
  + Funciones con parámetros **def nombre\_de\_la\_funcion (parametro1, parametro2)**
  + Funciones con retorno **def nombre\_de\_la\_funcion (parámetros)**

**Código de función**

**Return**

Devuelve el valor de resultado de la función y se puede asignar a una variable o ser usado sin necesidad de que tenga que ser conocido por el usuario.

* + Parámetros por defecto: en la definición de parámetros se puede igualar el valor del parametro a un valor por defecto, lo que hace que el parametro no se obligatorio de validar cuando se llama la función.
* **Funciones lambda** Una función lambda es una función que no tiene nombre y se hace para evitar el retur
  + Las funciones lambda se pueden igualar a una variable directamente
  + Son aptas para instrucciones unicas

5:17:42

12/08/2023