Lo primero de todo es descargar el compilador de Python en nuestro equipo

<https://www.python.org/>

El siguiente paso es descargar y configurar nuestro IDE.

Descargamos Visual Studio Code

Debemos instalar extensiones dentro de Visual Studio

Vamos a comprobar si lo ha realizado correctamente

Tendremos una carpeta llamada Python y desde ahí incluiremos nuestros ficheros de Python que finalizan con .py

Cuando creamos ficheros de programación debemos tener en cuenta algunas características

1. Sin espacios en los nombres
2. Sin caracteres especiales
3. No deben comenzar con número

Nosotros vamos a utilizar una nomenclatura visual para saber los ficheros de Python que realizamos

python01test.py

El lenguaje diferencia mayúsculas de minúsculas

Mostramos un mensaje en nuestro programa

Los textos que escribámosles de forma literal deben ir entre comillas dobles

“Hoy es juernes”

Para mostrar un mensaje se utiliza la palabra

print(“Nuestro mensaje”)

Para ejecutar el código, pulsamos sobre el Play

Ya podremos visualizar nuestro primer programa

INSTALAR GIT Y CONFIGURARLO

Windows:

<https://git-scm.com/downloads/win>

Agregamos la extensión de Git Hub dentro de Visual Studio Code

Nos validamos con GitHub

Una vez que nos hemos validado tendremos una opción en el menú que será Backup and Sync Settings

Esto nos permite sincronizar nuestras extensiones y personalizaciones de Visual Studio Code

A continuación vamos a subir nuestro Repositorio a GitHub. Debemos pulsar en Publish to Github

Al intentar subir a la nube nuestros ficheros, nos indica que no puede realizarlo porque no nos hemos dado de alta en el sistema operativo

Debemos abrir una ventana de comandos ( cmd ) y escribir las siguientes instrucciones

**git config --global user.email "you@example.com"**

**git config --global user.name "Your usuario github "**



Vamos a visualizar cómo podemos descargar nuestro proyecto y utilizarlo dentro de Git.

1.- Creamos una nueva carpeta dentro de Documento llamada pruebagit

2.- Abrimos esa carpeta con VS Code y creamos un nuevo fichero Python llamado test.py

print(“Soy una prueba”)

3.- Subimos nuestra carpeta a Github en un nuevo repositorio

4.- Hacemos cambios en el fichero test.py y los subimos a Github

5.- Cerramos visual studio y eliminamos la carpeta dentro de Documents llamada pruebagit

6.- Sincronizamos de nuevo nuestra carpeta con Git (descargar con Git Clone)

PYTHON

Es un lenguaje de programación, nos permite, mediante instrucciones poder ordenar características a nuestros equipos/programas

Todos los lenguajes tienen unas normas:

Los ficheros deben terminar en .py

No deben comenzar con número ni tener caracteres extraños

Todo el código (por ahora) estará dentro de un mismo archivo.

Muy importante: Las tabulaciones en Python son básicas, delimitan el código del programa

Las variables no se declaran con tipado, se asigna un valor y ese será su tipo de dato.

Numero = 2

Texto = “Mi string”

Si las variables son números, podremos sumar, restar o dividir

Si son textos, podremos convertir a mayúsculas o recorrer letra a letra

Creamos un nuevo fichero python02variables.py

**print("Ejemplo variables")**

**#ESTO ES UN COMENTARIO**

**#COMENTAR: COMMAND + K + C**

**# DESCOMENTAR: COMMAND + K + U**

**# Las variables no se declaran con tipado**

**# las variables las declaramos en minuscula**

**numero = 99**

**texto = "Variable de tipo texto (string)"**

**print(numero)**

**print(texto)**

Operaciones matemáticas

* Suma
* Resta

/ división

* Multiplicación

% Resto

\*\* POTENCIA

Para poder dibujar utilizamos print, pero podemos agregar algo más a print, es decir, lo que se llama una concatenación.

Mostrar un texto junto a una variable

print(“TEXT”, variable)

Tenemos otra posibilidad de concatenar que es utilizar el símbolo universal (+), esto nos permite realizar mensajes más complejos

Veremos que nos está dando un error:

Si utilizamos un + con una variable INT, está intentado sumar un string con un número.

Para solucionar esto, tenemos una funcionalidad llamada Funciones de conversión.

Python se basa en funciones. Una función es un bloque de código que “hace algo”.

Funciones de conversión:

\* str(variable): Convierte una variable a tipo string

* float(variable): Convierte una variable a decimal

\* int(variable): Convierte una variable a número entero

**print("Numero: " + str( numero))**

Las funciones de conversión NO hacen milagros.

Dentro de cualquier programa tenemos dos tipos de errores:

* Errores de compilación: Algo está mal escrito en el programa y el programa no compila
* Errores en ejecución: El programa funciona y compila correctamente, pero tiene algún fallo en algún momento de la ejecución
* Errores lógicos: Todo funciona, mi programa va perfecto, no tengo ningún fallo, pero no hace lo que quiero.

Creamos un nuevo fichero llamado **python03mates.py**

**print("Ejemplo mates variables")**

**numero1 = 20**

**numero2 = 3**

**suma = numero1 + numero2**

**multiplicacion = numero1 \* numero2**

**division = numero1 / numero2**

**resta = numero1 - numero2**

**print("Resta ", resta)**

**print(str(numero1) + " \* " + str(numero2) + "=" + str(multiplicacion))**

**print("División ", division)**

**print("Suma: ", suma)**

**redondeo = int(division)**

**print("Redondeo division: ", redondeo)**

PEDIR VALORES AL USUARIO

Hemos utilizado valores estáticos, pero a continuación vamos a combinar la petición de valores para las variables dentro del programa.

Para pedir valores se utiliza la función input()

Cuando pedimos valores a un usuario, siempre serán de tipo STRING, aunque pidamos valores numéricos

Modificamos el ejemplo anterior y pedimos valores al usuario

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

CONDICIONALES

Un condicional es una pregunta dentro del código

Dicho código ya no será lineal, sino que irá por un lado o por otro, dependiendo de las preguntas.

Los condicionales se escriben mediante la palabra if

Para utilizar condicionales dentro de Python debemos aplicar TABULACIONES

Mediante las tabulaciones, será código del IF o código del resto del programa

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Necesitamos saber los operadores de comparación

> Mayor

>= Mayor o igual

< Menor

<= Menor o igual

== Igual

**!=**

Distinto

Creamos un nuevo programa llamado python04condicionales.py

Texto

El contenido generado por IA puede ser incorrecto.

También tenemos la posibilidad de utilizar un ELSE, en el caso de que la condición no se cumpla

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un nuevo fichero llamado python05positivo.py

Pedimos un número al usuario e indicamos si es Positivo o negativo o cero

**print("Número positivo/negativo/cero")**

**print("Introduzca un numero")**

**numero = int(input())**

**if (numero > 0):**

**print("Positivo")**

**else:**

**if (numero == 0):**

**print("Cero")**

**else:**

**print("Negativo")**

**print("Fin de programa")**

Tenemos una sintaxis que nos permite preguntar de nuevo sobre una misma variable dentro de un IF, se llama ELSE IF y en Python se escribe elif

Modificamos nuestro código anterior para mostrar la nueva sintaxis:

**print("Número positivo/negativo/cero")**

**print("Introduzca un numero")**

**numero = int(input())**

**if (numero > 0):**

**print("Positivo")**

**elif (numero == 0):**

**print("Cero")**

**else:**

**print("Negativo")**

**print("Fin de programa")**

Vamos a crear otro fichero llamado **python06numeromayor.py**

Pedimos al usuario dos números y mostramos qué número es mayor o mostramos si los números son iguales

**print("Número mayor dos números")**

**print("Introduzca número 1")**

**numero1 = *int*(input())**

**print("Introduzca número 2")**

**numero2 = *int*(input())**

**if (numero1 > numero2):**

**print("El número " + *str*(numero1) + " es mayor que " + *str*(numero2))**

**elif (numero1 == numero2):**

**print("Los dos números son iguales")**

**else:**

**print("El número mayor es ", numero2)**

**print("Fin de programa")**

**LIBRERIAS EN PYTHON**

Una librería es una funcionalidad con una serie de características de algún tipo. Son métodos especializados en algo y que no están por defecto dentro de nuestro programa.

Tenemos varios tipos de librerías, Nativas de Python y otras que tendremos que instalar de forma externa.

Por ejemplo, si deseamos leer un fichero, ahora mismo NO podemos, nuestro programa no tiene las herramientas. Debemos darle una librería para que pueda leer.

Por ejemplo, si deseamos leer una base de datos Oracle, el programa no tiene ni idea, no tiene ni las herramientas ni las librerías.

Tenemos una librería que nos permite saber la tangente o el coseno con operaciones.

Dichas operaciones si deseamos utilizarlas en nuestro programa debemos **importar la librería math**

**PROGRAMA PY**

import miLibreria

miLibreria.Metodo1()

miLibreria.Metodo2()

Tenemos otra sintaxis:

**PROGRAMA PY**

from miLibreria import Metodo1, Metodo2

Metodo1()

Metodo2()

Vamos a visualizar un ejemplo sencillo con **math**

Creamos un nuevo fichero llamado **python07libreriamath.py**

**print("Ejemplo de librerías")**

**#SINTAXIS CON from**

**from math import floor, ceil, trunc**

**numero1 = 20**

**numero2 = 3**

**division = numero1 / numero2**

**print("La división es ", division)**

**# DECLARAMOS VARIABLES PARA ALMACENAR LOS VALORES**

**varFloor = floor(division)**

**varCeil = ceil(division)**

**varTrunc = trunc(division)**

**print("Floor ", varFloor)**

**print("Ceil ", varCeil)**

**print("Trunc ", varTrunc)**

**print("Fin de programa")**

**DEBUG APLICACIONES**

La depuración de aplicaciones es algo básico dentro de cualquier entorno de desarrollo.

Esta funcionalidad nos permite visualizar que va sucediendo con nuestro código

Paso a paso, podemos ir por los diferentes elementos de nuestro código y ver que sucede.

La depuración se utiliza para errores lógicos.

Vamos a visualizar la funcionalidad con el programa de math.

Debug en VS Code

1. Debemos indicar la línea donde desamos comenzar la depuración en nuestro código. Se puede realizar mediante la tecla F9

Texto

El contenido generado por IA puede ser incorrecto.

1. Pulsamos sobre **Run and Debug** en el menú lateral

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Una vez ejecutado, nuestro programa se detendrá en esa línea seleccionada

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

En la parte de la izquierda podremos visualizar lo que está haciendo nuestra aplicación con el valor de las variables

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

OPERADORES RELACIONALES

Estos operadores nos permiten realizar más de una pregunta dentro del mismo IF

Tenemos dos palabras para realizar estas consultas:

* **and: Todas las condiciones deben cumplirse**
* **or: Cualquier condición entrará en el IF**

Vamos a realizar un programa en el que pediremos tres números.

El programa nos dirá el mayor, el menor y el intermedio.

Creamos un nuevo programa llamado **python10mayortresnumeros.py**

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

SOLUCION

**print("Ejemplo mayor tres numeros")**

**print("Introduzca numero 1")**

**numero1 = *int*(input())**

**print("Introduzca numero 2")**

**numero2 = *int*(input())**

**print("Introduzca numero 3")**

**numero3 = *int*(input())**

**mayor = 0**

**menor = 0**

**intermedio = 0**

**#COMPARAMOS CADA NUMERO CON LOS OTROS DOS**

**if (numero1 >= numero2 and numero1 >= numero3):**

**mayor = numero1**

**elif (numero2 >= numero1 and numero2 >= numero3):**

**mayor = numero2**

**else:**

**mayor = numero3**

**#MISMA PREGUNTA PERO CAMBIANDO A SIMBOLO MENOR**

**if (numero1 <= numero2 and numero1 <= numero3):**

**menor = numero1**

**elif (numero2 <= numero1 and numero2 <= numero3):**

**menor = numero2**

**else:**

**menor = numero3**

**suma = numero1 + numero2 + numero3**

**intermedio = suma - mayor - menor**

**print("Mayor ", mayor)**

**print("Menor ", menor)**

**print("Intemedio ", intermedio)**

Creamos un nuevo programa **python11horasextra.py**

CALCULAR SALARIO DE TRABAJADORES

* Pediremos a un usuario las horas trabajadas, precio hora y los kilómetros

El trabajador tendrá horas extra a partir de la hora 36

Cada hora extra será 2€ más

Solamente mostraremos mensajes

* Si el trabajador hace menos de 100 km las dietas son LOCALES
* Si el trabajador hace entre 101km y 500 km las dietas son PROVINCIALES
* Si hace más km, serán NACIONALES
* Si el precio final es menor a 250€ SIN RETENCIONES
* Si es entre 250 y 600, 20% retención
* Si es mayor 40 % retención

**Número horas: 20**

Horas extra: 4

**Precio hora: 20**

Salario base: 4000

Salario Extra: 0

Salario total: 4000

Texto

El contenido generado por IA puede ser incorrecto.

SOLUCION

**print("Ejemplo Horas extra")**

**print("Horas trabajadas")**

**horasTrabajadas = *int*(input())**

**print("Precio hora")**

**precioHora = *int*(input())**

**print("Kilometros recorridos")**

**km = *int*(input())**

**horasExtra = 0**

**salarioExtra = 0**

**salarioBase = 0**

**salarioTotal = 0**

**dietas = ""**

**retencion = ""**

**#PREGUNTAMOS SI TENEMOS HORAS EXTRA**

**if (horasTrabajadas > 36):**

**#HORAS EXTRA**

**horasExtra = horasTrabajadas - 36**

**salarioBase = precioHora \* 36**

**salarioExtra = horasExtra \* (precioHora + 2)**

**else:**

**#NO HA HECHO HORAS EXTRA**

**horasExtra = 0**

**salarioExtra = 0**

**salarioBase = horasTrabajadas \* precioHora**

**salarioTotal = salarioBase + salarioExtra**

**if (km <= 100):**

**dietas = "LOCALES"**

**elif (km >= 101 and km <= 500):**

**dietas = "PROVINCIALES"**

**else:**

**dietas= "NACIONALES"**

**if (salarioTotal < 250):**

**retencion = "SIN RETENCION"**

**elif (salarioTotal >= 250 and salarioTotal <= 600):**

**retencion = "20% Retencion"**

**else:**

**retencion = "40% Retención"**

**#INFORME**

**print("Informe de salario")**

**print("Horas trabajadas ", horasTrabajadas)**

**print("Horas extra ", horasExtra)**

**print("Precio hora ", precioHora)**

**print("Precio extra ", (precioHora + 2))**

**print("Salario base ", salarioBase)**

**print("Salario extra ", salarioExtra)**

**print("Salario total ", salarioTotal)**

**print("Retenciones ", retencion)**

**print("Dietas ", dietas)**

BUCLES

Un bucle es una secuencia que se repite con una condición o una serie de veces.

Tenemos dos tipos de bucles:

1. **while:** Se realiza mediante una condición. Mientras que la condición no se cumpla, permanece dentro del bucle.

Debemos indicar, de alguna forma dentro del bucle, que la condición se debe cumplir.

Tenemos la posibilidad de realizar bucles infinitos

Sintaxis del bucle while:

while (condicion == true):

# instrucciones

condicion = false

1. **for:** Este bucle nos permite repetir una secuencia N veces. La secuencia se realiza con un contador declarado en la sintaxis del propio bucle.

Entra y sale solo, no hace bucles infinitos

Sintaxis Bucle Contador (for)

for **contador** in range(final):

#instrucciones

La variable **contador** siempre comienza en CERO con esta sintaxis

Tenemos una sintaxis para indicar el inicio y fin del bucle

for **contador** in range(inicio, final):

#instrucciones

Creamos un nuevo programa llamado **python12bucles.py**

**print("Ejemplo bucles")**

**print("WHILE")**

**# NECESITAMOS UNA VARIABLE PARA LA CONDICION DEL BUCLE**

**contador = 0**

**while (contador <= 5):**

**# DEBEMOS INDICAR QUE SALDREMOS DEL BUCLE**

**print("Contador ", contador)**

**contador = contador + 1**

**print("Fin de programa")**

Tabla

El contenido generado por IA puede ser incorrecto.

**print("Ejemplo bucles")**

**print("For")**

**# NORMALMENTE LAS VARIABLES DE LOS BUCLES CONTADORES**

**# SE REPRESENTAN CON UNA SOLA LETRA (i, z, j)**

**for i in range(5):**

**print("Valor de i ", i)**

**for i in range(1 , 6):**

**print("Valor i: ", i)**

**print("WHILE")**

**# NECESITAMOS UNA VARIABLE PARA LA CONDICION DEL BUCLE**

**contador = 0**

**while (contador <= 5):**

**# DEBEMOS INDICAR QUE SALDREMOS DEL BUCLE**

**print("Contador ", contador)**

**contador = contador + 1**

**print("Fin de programa")**

Vamos a realizar una aplicación en la que pediremos al usuario un número inicial y un número final y mostraremos los números pares que existan entre los dos.

Creamos un nuevo programa llamado **python13rangopares.py**

**print("Rango de números pares")**

**print("Introduzca un inicio")**

**inicio = *int*(input())**

**print("Introduzca un valor final")**

**fin = *int*(input())**

**# REALIZAMOS UN BUCLE DESDE UN INICIO HASTA UN FINAL + 1**

**for i in range(inicio, fin + 1):**

**# PREGUNTAMOS SI EL NUMERO ES PAR**

**if (i % 2 == 0):**

**print(i)**

**print("Fin de programa")**

Vamos a realizar un ejemplo para demostrar la conjetura de Collatz

La conjetura dice lo siguiente:

Todo número positivo siempre llegará a ser 1 cumpliendo estas dos condiciones:

* Si el número es PAR, dividimos entre 2
* Si el número es IMPAR, multiplicamos \* 3 y sumamos 1

6, 3, 10, 5, 16, 8, 4, 2, 1

Creamos un nuevo programa llamado **python14collatz.py**

**print("Conjetura de Collatz")**

**print("Introduzca un número")**

**numero = *int*(input())**

**while (numero != 1):**

**if (numero % 2 == 0):**

**numero = *int*(numero / 2)**

**else:**

**numero = numero \* 3 + 1**

**print(numero)**

**print("Fin de programa")**

Realizamos un programa nuevo en el que mostraremos la tabla de multiplicar de un número.

Pediremos al usuario el número y le mostraremos la siguiente tabla

5 \* 1 = 5

5 \* 2 = 10

Texto

El contenido generado por IA puede ser incorrecto.

**print("Tabla multiplicar")**

**print("Introduzca número")**

**numero = *int*(input())**

**for i in range(1, 11):**

**operacion = numero \* i**

**print(*str*(numero) + " \* " + *str*(i) + "=" + *str*(operacion))**

**print("Fin de programa")**

**CLASE STRING PYTHON**

Las clases, por pertenecer a algún tipo (string) tienen una serie de métodos para trabajar con dichos tipos.

Existen clases simples como, por ejemplo, la clase int que no tiene nada o clases más complejas como pueden ser la clase string o una clase que acceda a leer ficheros.

Una clase de objeto contiene una serie de métodos que tienen que ver con su tipado, por ejemplo, dentro de la clase string tenemos un método que nos permite poner el texto en mayúsculas.

Tenemos un método para saber la longitud de un texto: **len()**

Todo texto comienza siempre en índice CERO.

En realidad un string es un conjunto de caracteres, podemos recorrer cada carácter si lo deseamos.

Se diferencia mayúsculas de minúsculas en los string

texto = “Lenguaje Python”

longitudTexto = len(texto) 🡪 15

También tenemos la posibilidad de acceder a cada carácter mediante su posición de índice

Objeto[**índice**]

texto[0] 🡪 L

texto[1] 🡪 e

Tenemos una serie de métodos para trabajar con estos objetos string:

* lower(): Convierte a minúscula un texto
* upper(): Convierte a mayúscula un texto
* find(“Contenido a buscar”): Busca el contenido dentro del texto y nos devuelve su posición. Si no lo encuentra, nos devuelve -1
* rfind(“Contenido a buscar”): Busca el contenido dentro del texto y nos devuelve su posición. Si no encuentra, nos devuelve -1. Comienza a buscar desde el final

Dentro de Python tenemos POO y eso nos permite tener sobrecarga en algunos métodos/funciones

* find(“Contenido a buscar”, índice): Busca el contenido dentro del texto y nos devuelve su posición comenzando a buscar desde el índice. Si no lo encuentra, nos devuelve -1
* startswith(“contenido”): Indica si el texto comienza con el contenido
* endswith(“contenido”): Indica si el texto finaliza con el contenido
* replace(“antiguo”, “nuevo”): Sustituye el contenido del texto antiguo por el nuevo
* count(“texto”): Nos devuelve el número de veces que aparece el texto

Tenemos métodos para preguntar por el tipo de contenido del string:

* isdigit(): Indica si el contenido del texto son números
* isalpha(): Indica si el contenido del texto son letras
* isalnum(): Indica si el contenido está formado por letras y números
* isupper(): Indica si un texto está en mayúsculas
* islower(): Indica si el texto está en minúscula

Dentro de Python con string tenemos la posibilidad de extraer subcadenas de un texto.

En Python a esto se le llama **STRING SLICING**

texto = “primer python”

Tenemos dos posibilidades para extraer subcadenas:

1. Queremos la subcadena desde una posición concreta en adelante. Objeto[índice:]

texto[8:] 🡪 YTHON

1. Devuelve desde una posición inicial hasta una posición final. objeto[inicio:final]

texto[0:2] 🡪 PR

Creamos un nuevo programa **python16string.py**

**print("Clase string y funciones")**

**texto = "primero python"**

**#VAMOS PROBANDO METODOS Y VIENDO QUE DEVUELVEN**

**print("upper ", texto.upper())**

**print("replace " + texto.replace("o", "@"))**

**print("Letra 0: " + texto[0])**

**print("Logitud (len)", len(texto))**

**print("find P: ", texto.find("p"))**

**print("find Z: ", texto.find("z"))**

**# SOBRECARGA DE FIND (contenido a buscar, indice)**

**print("find p sobrecarga ", texto.find("p", 1))**

**print("rfind p ", texto.rfind("p"))**

**print("startswith A ", texto.startswith("A"))**

**print("endswith n ", texto.endswith("n"))**

**print("isdigit() ", texto.isdigit())**

**print("isalpha() ", texto.isalpha())**

**print("isalnum() ", texto.isalnum())**

**# Vamos a visualizar que pasa con SLICING**

**# SUBSTRING**

**# QUEREMOS RECUPERAR DESDE LA POSICION 2 EN ADELANTE**

**substring = texto[2:]**

**print("Posición 2 en adelante ", substring)**

**# EN PYTHON PODEMOS RECUPERAR UNA SUBCADENA**

**# DESDE UNA POSICION (2) A OTRA POSICION (5)**

**subtexto = texto[2: 5]**

**print("texto[2: 5] ", subtexto)**

**# PODEMOS RECORRER CADA CARACTER DE UN TEXTO**

**longitud = len(texto)**

**for i in range(longitud):**

**letra = texto[i]**

**print("Posición " + *str*(i) + ": " + letra)**

**# PODEMOS HASTA COMPROBAR QUE EL USUARIO HA ESCRITO NUMEROS O NO**

**print("Introduzca un número")**

**# PRIMERO A UNA VARIABLE AUXILIAR**

**aux = input()**

**if (aux.isdigit()):**

**print("Esto es un número!!!")**

**else:**

**print("No me has dado un número, campeón...")**

**print("Fin de programa")**

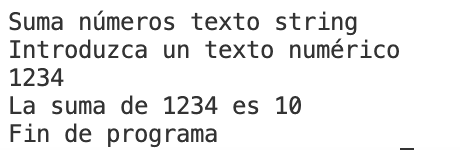
Necesito un programa llamado **python17sumanumeros.py**

Pediremos al usuario un texto numérico

“1234”

Debemos sumar cada carácter del texto y mostrar la suma de cada número/letra

**La suma es 10**



**print("Suma números texto string")**

**print("Introduzca un texto numérico")**

**textoNumeros = input()**

**suma = 0**

**longitud = len(textoNumeros)**

**# RECORREMOS CADA LETRA DEL TEXTO**

**for i in range(longitud):**

**# ALMACENAMOS CADA LETRA DE CADA POSICION**

**letra = textoNumeros[i]**

**# CONVERTIMOS CADA LETRA A INTEGER**

**numero = *int*(letra)**

**# SUMAMOS CADA NUMERO**

**suma = suma + numero**

**print("La suma de " + textoNumeros + " es " + *str*(suma))**

**print("Fin de programa")**

Vamos a realizar un programa para validar un Email con Python

El usuario introducirá un email y le indicaremos si está mal y la razón.

**NO QUIERO BUCLES, TODO CON METODOS**

* Que el email contenga @
* Que el email tenga un punto
* @ ni al principio ni al final
* Punto ni al principio ni al final
* Que el email solamente tenga una @
* Que exista un punto después de la @
* El dominio debe ser de 2 a 3 caracteres

SOLUCION

**print("Validacion Email")**

**print("Introduzca un email")**

**email = input()**

**# EMAIL CON @**

**if (email.count("@") == 0):**

**print("Email sin @")**

**# QUE EXISTA UN PUNTO**

**elif (email.find(".") == -1):**

**print("Email sin punto")**

**# EMAIL SIN @ AL INICIO NI FINAL**

**elif (email.startswith("@") == True or email.endswith("@") == True):**

**print("@ al inicio o al final del email")**

**# Punto ni al inicio ni al final**

**elif (email[0] == "." or email.endswith(".") == True):**

**print("Punto al inicio o al final")**

**# Email con una sola @**

**elif (email.count("@") > 1):**

**print("Existe más de una @")**

**# Exista un punto despues de @**

**elif (email.find("@") > email.rfind(".")):**

**print("Debe existir un punto despues de la @")**

**else:**

**# Necesitamos recuperar el dominio**

**ultimoPunto = email.rfind(".")**

**# RECUPERAMOS EL DOMINIO A PARTIR DEL ULTIMO PUNTO EN ADELANTE**

**dominio = email[ultimoPunto + 1:]**

**# LONGITUD DEL DOMINIO**

**longitudDominio = len(dominio)**

**# Comprobar la longitud 2-3**

**if (longitudDominio >= 2 and longitudDominio <= 3):**

**print("Email correcto " + email)**

**else:**

**print("El email debe tener un dominio de 2 a 3 caracteres")**

**print("Fin de programa")**

## FUNCIONES/METODOS DENTRO DEL LENGUAJE PYTHON

Una función es un método.

Son herramientas que contienen los programas de Python y las clases de Python.

Clase de Python: (string) upper()

Programas de Python: python19email: len(dominio)

Un programa de Python se ejecuta y UTILIZA clases de Python

Los métodos/funciones nos permiten realizar códigos NO LINEALES

Actualmente mi programa realiza la validación de un Email.

¿Qué sucede si también queremos validar un ISBN?

¿Qué sucede si también queremos validar un DNI?

Un método nos permite poder reutilizar código además de organizar nuestros programas.

Ya no estará el código lineal, sino que haremos las llamadas a otros códigos cuando lo necesitemos

Funciona mediante llamadas y declaraciones de métodos

PROGRAM PYTHON (main)

print(“Validacion Email”)

print(“Introduzca un email”)

email=input()

**validarEmail()**

En otro lugar del código o programa Python:

validarEmail(emailRecibido):

# REALIZAMOS LAS VALIDACIONES

Para crear métodos dentro de Python se utiliza una palabra clave llamada **def** y a continuación el nombre del método

Los métodos pueden estar en el mismo fichero Program main o en otros ficheros.

Si están en el mismo Program, lo primero que debemos hacer es escribir los métodos que necesitaremos.

Los métodos/funciones pueden recibir parámetros

Creamos un nuevo programa llamado **python20metodos.py**

**# ES UN PROGRAMA QUE DESEAMOS EJECUTAR**

**# LOS METODOS SE DECLARAN AL INICIO**

**# LA SINTAXIS DE NOMBRES DE METODOS ES**

**# primeraSegundaTercera**

***def* primerMetodo():**

**# ESTE CODIGO NUNCA SE EJECUTARA SI NO LO**

**# LLAMAMOS DE FORMA EXPLICITA**

**print("Soy el primer metodo")**

***def* segundoMetodo():**

**print("Segundo metodo")**

**# AQUI TENEMOS EL CODIGO PRINCIPAL**

**print("Ejemplo de metodos")**

**# PODEMOS LLAMAR AL METODO SI LO DESEAMOS**

**primerMetodo()**

**segundoMetodo()**

**primerMetodo()**

**print("Fin de programa")**

### PARAMETROS EN LOS METODOS

Un método puede recibir parámetros, es decir, información para la ejecución del método.

Los parámetros se reciben dentro de los paréntesis del método y, si recibimos más de un parámetro, irán separados mediante comas.

Los nombres de parámetro no tienen que coincidir con los nombres de variable que estamos enviando.

Los parámetros solamente se utilizarán dentro del método.

Ejemplo:

def metodoParametros(**nombre**):

print(“Su nombre es “ + nombre)

name = input()

metodoParametros(**name**)

def metodoParametros(**nombre, num1, num2**):

Creamos un nuevo programa llamado **python21metodosparametros.py**

***def* saludar(*nombre*):**

**print("Bienvenido a Python Mr/Mrs " + nombre)**

***def* despedirse(*nombre*, *dia*):**

**print("Un placer hoy " + dia + " Mr/Mrs " + nombre)**

**# -----------------------------------------------------**

**print("Metodos con parametros")**

**name = "Alumno"**

**dia = "miercoles"**

**saludar(name)**

**despedirse("Jueves", dia)**

**print("Fin de programa")**

Los métodos podemos dividirlos entre dos grupos:

* Métodos de acción: Ejecutan una serie de acciones
* Métodos **return**: Ejecutan una serie de acciones y devuelven un valor después de ejecutarlas

Acción: **input()**

Return: len()

Creamos un nuevo programa llamado **python22metodosreturn.py**

***def* convertirMayusculas(*texto*):**

**return texto.uppper()**

***def* convertirMinusculas(*texto*):**

**return texto.lower()**

***def* concatenar(*texto1*, *texto2*):**

**resultado = texto1 + texto2**

**return resultado**

***def* mostrarMenu():**

**print("Seleccione una opción")**

**print("1.- Convertir mayúsculas")**

**print("2.- Convertir minúsculas")**

**print("3.- Concatenar textos")**

**# PROGRAMA PRINCIPAL**

**print("Metodos return")**

**print("Introduzca un texto")**

**valor = input()**

**mostrarMenu()**

**opcion = *int*(input())**

**resultado = ""**

**if (opcion == 1):**

**resultado = convertirMayusculas(valor)**

**elif (opcion == 2):**

**resultado = convertirMinusculas(valor)**

**else:**

**print("Ponga otro texto")**

**otro = input()**

**resultado = concatenar(valor, otro)**

**print(resultado)**

**print("Fin de programa")**

Creamos un nuevo programa **python23calculadora.py**

1. Pediremos al usuario dos números
2. Debemos mostrar un menú con estas operaciones:

1.- Sumar

2.- Restar

3.- Multiplicar

Tendremos tres funciones/métodos **return** para cada una de las opciones y mostraremos el resultado de lo que el usuario ha seleccionado