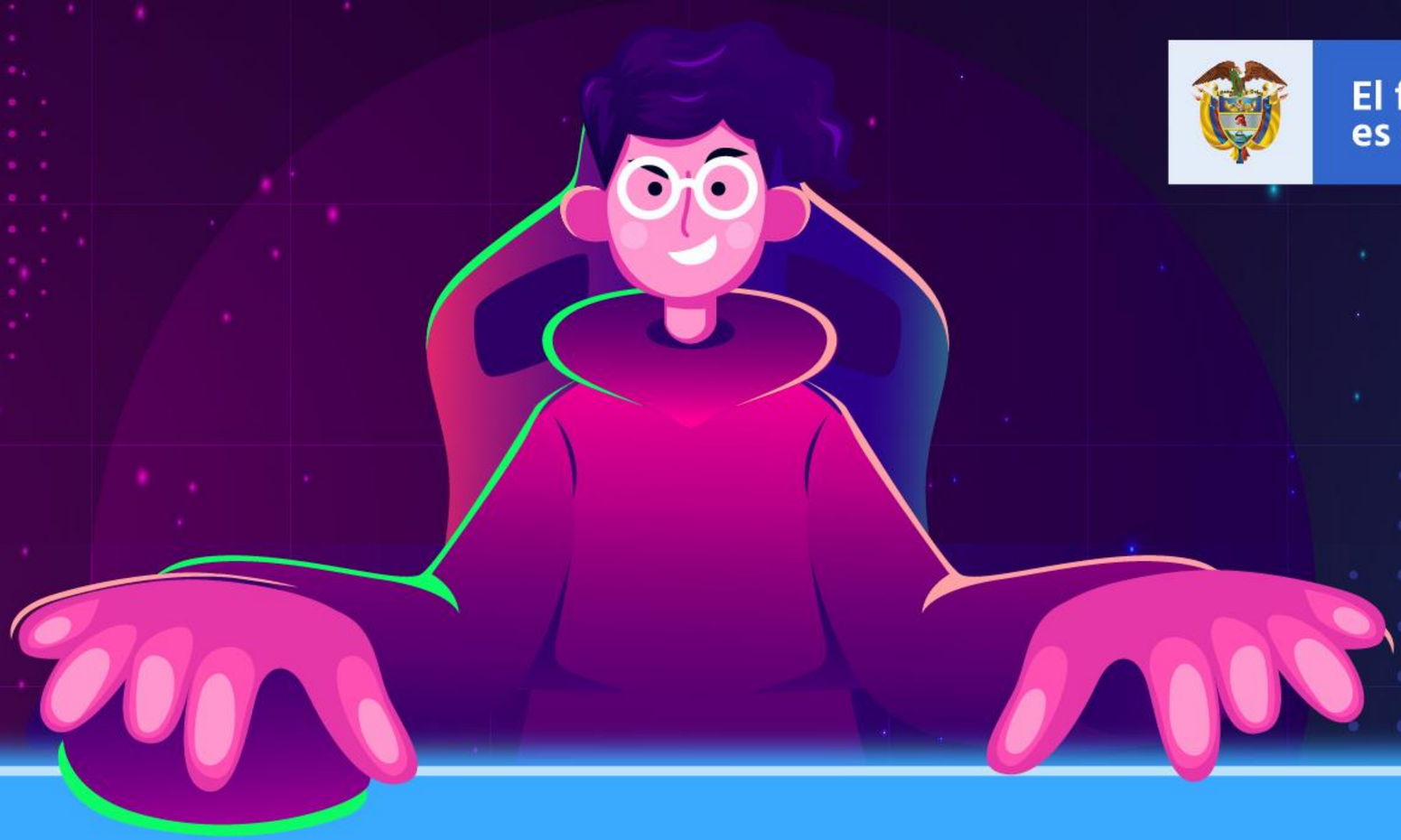


El futuro digital es de todos

MinTIC

Funciones en Python

OPERADO POR:

Universidad Tecnológica de Pereira

Mision TIC 2022

RUTA DE APRENDIZAJE 1

Funciones

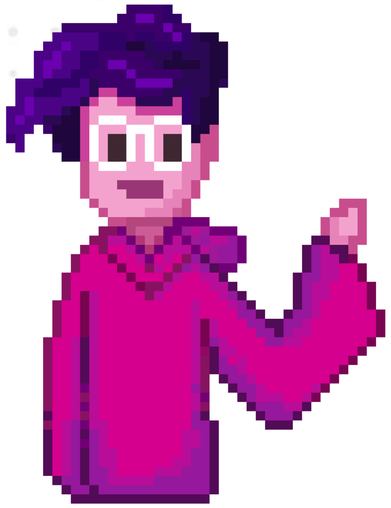
Una función es un bloque de código con un nombre asociado con el que podemos diferenciarlas de otras funciones o nos permite saber que hace.

Una función recibe cero o más valores que conocemos como **argumentos** como entrada, sigue una secuencia de instrucciones, y ejecuta una operación y nos devuelve un valor y/o realiza una tarea. Lo mas interesante es que estos bloques de funciones puede ser llamados cuando se necesite realizar ese calculo, lo que nos facilita mucho el proceso.

El uso de funciones es un componente muy importante en el proceso de la programación, y tiene varias ventajas:

Modularización: permite segmentar o dividir un programa complejo en una serie de partes o módulos más simples, facilitando así la programación y la revisión del código.

Reutilización: permite reutilizar una misma función en distintos programas.



Python tiene una serie de funciones que están adicionadas o integradas en el lenguaje, y también permite crear funciones nuevas definidas por el usuario para ser usadas en su propios programas.

Llamado de funciones

En el contexto de la programación, una función es una secuencia de declaraciones con nombre que realiza un cálculo. Cuando se define una función, se especifica el nombre y la secuencia de declaraciones. Más tarde, puede "llamar" la función por su nombre. Ya hemos visto un ejemplo de llamada de una función:

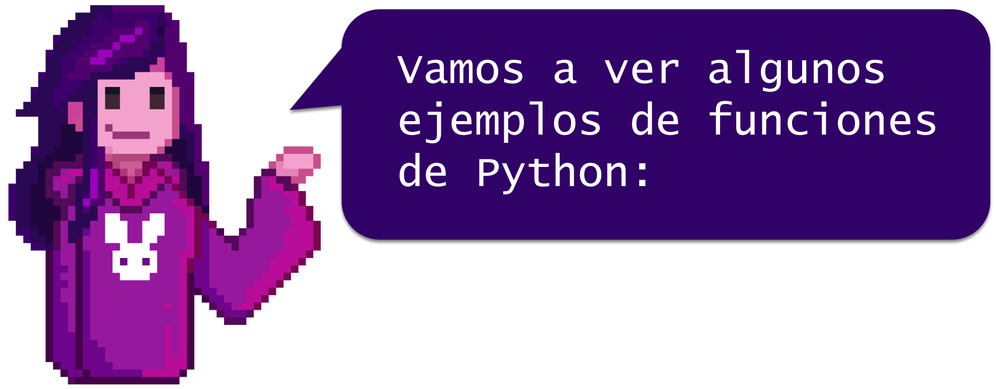
```
In [ ]: print("HOLA MUNDO")
```

El nombre de la función es **"print"**. La expresión entre paréntesis se llama el argumento de la función. El argumento es un valor o variable que pasamos a la función como entrada de la función. El resultado, para la función print, es la impresión en pantalla de la cadena de caracteres que tiene el argumento.

Es común decir que una función "toma" un argumento y "devuelve" un resultado. El resultado se llama el *valor de retorno*.

Funciones propias de Python

Python proporciona una serie de importantes funciones incorporadas que podemos utilizar sin necesidad de proporcionar la definición de la función. Los creadores de Python escribieron un conjunto de funciones para resolver problemas comunes y las incluyeron en Python para que las usáramos.



Max y Min

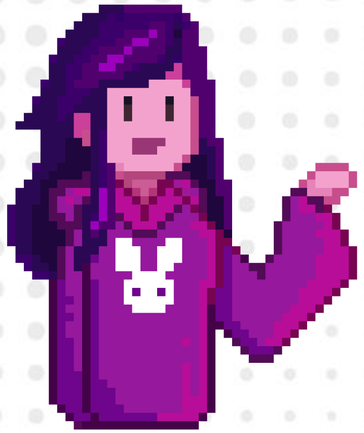
Las funciones max y min nos dan los valores más grandes y más pequeños de un arreglo, respectivamente. Este arreglo puede ser una lista, un vector e incluso un conjunto de variables:

Vamos a construir un código donde Python nos ayude a encontrar cual es el valor mas grande y el valor mas pequeño entre diferentes valores:

Primero vamos a crear variables diferentes y asignamos sus valores.

Luego vamos a usar las funciones que trae Python, creando dos variables nuevas que vamos a llamar “máximo” y “mínimo” las cuales serán iguales al resultado que nos entreguen las funciones con nombre clave “max()” y “min()”

```
var1 = 10
var2 = 30
var3 = 120
var4 = 500

maximo = max()
print(maximo)

minimo = min()
print(minimo)
```



La función `max()` y `min()`, llevan entre los paréntesis, los valores que van a comparar separados por comas. Solo nos queda imprimir el valor para asegurarnos que esta entregando la respuesta correcta.

```
var1 = 10
var2 = 30
var3 = 120
var4 = 500

maximo = max(var1, var2, var3, var4)
print(maximo)

minimo = min(var1, var2, var3, var4)
print(minimo)
```

En este caso tendremos como resultado: 50 0 y 10.

Siempre es importante revisar el resultado que nos entregan las funciones, especialmente si no son funcione de Python y son diseñadas y creadas por nosotros.

Item desbloqueado: Max y Min.

Help

La función **Help** llama al sistema de ayuda integrado de Python.

```
In [ ]: help(print)

In [ ]: help(max)

In [ ]: help(min)

In [3]: help(type)
```

Sin importar si estamos trabajando con spyder o con jupiter, siempre podremos acceder a la ayuda de Python usando la función help().

La función help() es una de las mas útil es, nos permite buscar otras funciones y sabe como trabajar correctamente con ellas, incluso podemos encontrar funciones que no conocemos y una breve explicación de que hace.

Para utilizas la función help(), solo debemos escribir dentro de los paréntesis el comando del que queramos tener mas información.

¡Observación importante!

La función **help()** siempre nos entregara la información en su lenguaje original, en este caso ingles. Esto también es debido a que idioma ingles es mas conocido y mas globalizado que otros idiomas en el mundo.

Por eso es tan importante para un programador saber ingles. No solo para entender mejor que nos dice ython con su función help, también para comprender mas fácilmente las soluciones que otro s programadores en el mundo han encontrado para errores comunes y comparten con la comunidad usando ingles.

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
help(max)
```

```
Help on built-in function max in module builtins:

max(...)
    max(iterable, *[, default=obj, key=func]) -> value
    max(arg1, arg2, *args, *[, key=func]) -> value

    With a single iterable argument, return its biggest item. The
    default keyword-only argument specifies an object to return if
    the provided iterable is empty.
    With two or more arguments, return the largest argument.
```

Sum:

Las función suma nos permite como su nombre dice, realizar una suma de 2 o mas valores, los cuales podemos definir como variables que pueden tener un valor si e usuario lo escribe en consola o nosotros decidimos utilizar una base de datos con un formato especial.

```
In [5]: var1 = 10
        var2 = 30
        var3 = 120
        var4 = 500

        sum((var1, var2, var3, var4))

Out[5]: 660
```

La función sum() en este caso necesita de otro par de paréntesis para definir las variables que va a leer y sumar. De nuevo las variables deben ser separadas por una coma.

Para este ejemplo vamos a definir nuevamente 4 variables y si mplemente vamos a llamar la función suma.

Range():

Las función range() crea un vector donde muestra el valor de inicio separado con una coma, y la cantidad de valores totales.

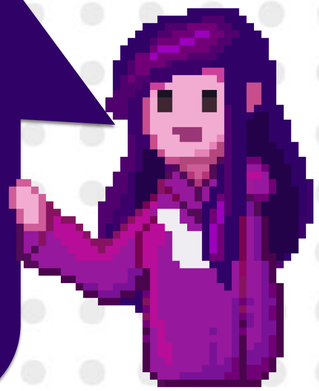
Len():

Entrega el tamaño de un vector, o de un string (cadena de letras o simbolos), listas, etc.

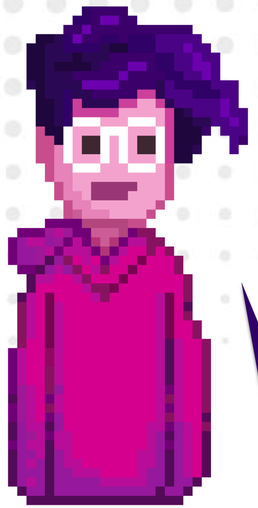
```
In [7]: x = range(5) #DEVUELVE LOS NUMERO DEL 0 AL 4
        print(x)

        range(0, 5)

In [8]: x = range(0,5)
        print(x)

        len(x) #Esto tendria un tamaño de 5 numeros {0, 1, 2, 3, 4}

        range(0, 5)

Out[8]: 5
```

En el ejemplo para la función range(), si solo le damos un valor o argumento a la función, nos entregara como resultado un vector (0,5) donde 0 es el numero de inicio y 5 el valor total de valores que le pedimos.

En el segundo ejemplo, podemos ver que no importa si lo definimos con su valor inicial y el numero total de valores, la respuesta es la misma. La función len() solo me entrega el valor 5,tamaño del vector.



Haciendo nuestras propias funciones



Hasta ahora, sólo hemos usado las funciones que vienen con Python, pero también es posible añadir y crear nuevas funciones.

Una definición de función especifica seria: el nombre de la nueva función y la secuencia de declaraciones o instrucciones que se ejecutan cuando se llama la función. Una vez que definimos una función, podemos reutilizar la función una y otra vez a lo largo de nuestro programa.

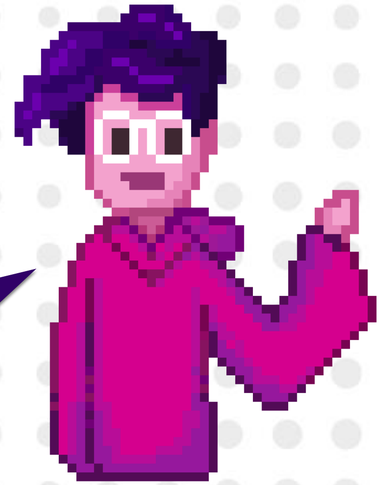
Para definir una función debemos usar el comando **def** con el cual podremos crear una función nueva, solo debemos darle un nombre e indicarle si necesita de valores de entrada para hacer la tarea que vamos a programar.

Para entender mejor el concepto, vamos a hacer un ejemplo.

Primero vamos a escribir el comando **def**, dejamos un espacio y escribimos el nombre de nuestra función en este caso **func** y un par de paréntesis.

Dentro de estos paréntesis podemos separar con comas los valores que nuestra función necesita para hacer correctamente los cálculos.

```
#Crear una función:
def func(parámetro1, parámetro2):
```



Debemos usar los mismos nombres que le asignemos a las variables o los parámetros para que la función los lea correctamente.

¡En Python lo que se conoce como sangría es extremamente importante!

La sangría se genera cuando hacemos **click** en la tecla **Tab** del teclado, lo que nos genera un **espacio grande** en el código como vemos en la línea que dice **#Codigo de la función...**

La sangría nos permite indicar en Python que algunas líneas de código pertenecen a un bloque o conjunto de instrucciones en particular.

En nuestro ejemplo todas las instrucciones que queramos dentro de nuestra nueva función debe llevar sangría para que Python sepa que están dentro de la función o hacen parte de ella.

```python
#Crear una función:
def func(parámetro, parámetro):
    #código de la función <-- Identación o Sangría
    return
```

Sangría

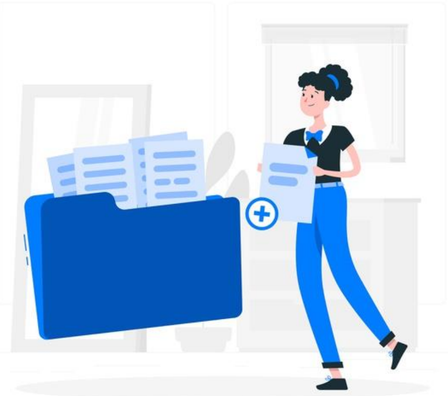
Si dejamos todo en la misma columna o no aplicamos la sangría Python no podrá entenderlo que queremos hacer y nos generara un error al ejecutar el código.

```python
#Crear una función:
def func(parámetro1, parámetro2):
    #Instrucción1
    #Instrucción2
    return
```

Una vez tenemos el cursor en la posición correcta usando la sangría , podemos comenzar a escribir las instrucciones dentro de nuestra nueva función.

Podríamos ver las funciones que creamos en Python como una receta, que vamos a guardar bajo un nombre especial, y toda vez que necesitemos usar esa receta, solo debemos buscarla por su nombre o en el caso de Python llamarla por su nombre.

Para llamar la función solo debemos usar el nombre que usamos para definirla y dentro de su paréntesis escribir los argumentos o valores que necesita la función para hacer su tarea.



Vamos a llamar la función que c
reamos. Solo tenemos que escrib
ir la nombre de la función, en
este ejemplo seria func y entre
sus paréntesis los nombres de l
os argumentos, variables o pará
metros que necesita para realiz
ar su calculo.

```python
#Crear una función:
def func(parámetro, parámetro):
    #código de la función <-- Identación o Sangría
    return

#Llamar la función:
func(argumento, argumento)

#Almacenar el retorno de la función (si tiene uno):
variable = func(argumento, argumento)
```

También es posible guardar el valor que regresa la función utilizando una variable, como hicimos en casos anteriores. Sin embargo debemos hacer la variable igual a la función con sus argumentos.

Estos son los pasos entonces, para crear y llamar una función desde cualquier parte de nuestro código, y con eso mejorar la forma en la que programamos.

Sentencia def

La sentencia def es una definición de función usada para crear objetos, funciones definidas por el usuario.

Su ejecución enlaza el nombre de la función en el namespace local a un objecto función. Este objeto función contiene una referencia al namespace local global para ser usado cuando la función es llamada.

La definición de función no ejecuta el cuerpo de la función solo la usamos para definir cual función queremos crear; la función solo es ejecutada cuando es llamada.

La sintaxis para una definición de función en Python es:

```
def NOMBRE(LISTA_DE_PARAMETROS):

    SENTENCIAS
    RETURN [EXPRESION]
```

A continuación se detallan el significado de pseudo código fuente anterior:

- **NOMBRE**, es el nombre de la función.
- **LISTA_DE_PARAMETROS**, es la lista de parámetros que puede recibir una función.
- **DOCSTRING_DE_FUNCION**, es la cadena de caracteres usada para documentar la función.
- **SENTENCIAS**, es el bloque de sentencias en código fuente Python que realizar cierta operación dada.
- **RETURN**, es la sentencia return en código Python.
- **EXPRESION**, es la expresión o variable que devuelve la sentencia return.

Vamos a crear nuestra primera función.

Vamos a crear nuestra primera función

```
def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')
```

```
imprime_Cosas()
```

El nombre de la función es imprime_Cosas. Las reglas para los nombres de las funciones son las mismas que para los nombres de las variables: las letras, los números y algunos signos de puntuación son permitidos, pero el primer carácter no puede ser un número. No se puede utilizar una palabra clave como nombre de una función, y se debe evitar tener una variable y una función con el mismo nombre.

Los paréntesis vacíos después del nombre indican que esta función no tiene argumentos. Más adelante construiremos funciones que toman argumentos como sus entradas.

La primera línea de la definición de la función se llama **cabecera**; el resto se llama **cuerpo**.

```
def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')
```

```
imprime_Cosas()
```

La cabecera tiene que terminar con dos puntos y el cuerpo tiene que estar con sangría. Por convención, la sangría es siempre de cuatro espacios (o podemos oprimir la tecla tab). El cuerpo puede contener cualquier número de declaraciones.

```
def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')
```

Cabecera

```
imprime_Cosas()
```

Cuerpo

```
def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()
```

```
repetir_funciones()
```

Una vez que se ha definido una función, se puede utilizar dentro de otra función. Por ejemplo, para repetir las frases anteriores, podríamos crear una función nueva llamada repetir_funciones:

Al ejecutar el código tendremos como resultado, que la función repetir_funciones() ¡llamo a la función imprime_Cosas() dos veces!

```
repetir_funciones()
```

```
La clase esta genial
Python es lo maximo
La clase esta genial
Python es lo maximo
```

Reuniendo los fragmentos de código de la sección anterior, todo el programa se ve así:

```
def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')

def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()

repetir_funciones()
```

Este programa contiene dos definiciones de funciones: imprime_Cosas y repetir_funciones. Como es de esperar, hay que crear una función antes de poder ejecutarla. En otras palabras, la definición de la función tiene que ser ejecutada antes de la primera vez que se llama.

Ejercicio 1: Movamos la última línea de este programa a la parte superior, para que la llamada de la función aparezca antes de las definiciones. Veamos que pasa!!

```
repetir_funciones()

def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')

def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()
```

```
repetir_funciones()

def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')

def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()
```

```
La clase esta genial
Python es lo maximo
La clase esta genial
Python es lo maximo
```

Al ejecutar el código tendremos como resultado, que la función repetir_funciones() de la misma manera que lo hicimos con el código anterior.

Ejercicio 2: Ahora hagamos otra prueba, movamos la llamada de la función de nuevo a la parte inferior y luego movamos la definición de imprime_Cosas después de la defínción repetir_funciones. ¡¡Miremos que pasa!!

```
def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()

def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')

repetir_funciones()
```

```
def repetir_funciones():
    imprime_Cosas()
    imprime_Cosas()

def imprime_Cosas():
    print("La clase esta genial")
    print('Python es lo maximo')

repetir_funciones()
```

```
La clase esta genial
Python es lo maximo
La clase esta genial
Python es lo maximo
```

Al **ejecutar** el código tendremos como resultado, se imprimió el mensaje de dos veces de la misma manera que lo hicimos con el primer código.

En Python no es necesario que se tenga un orden especial cuando vamos a definir las funciones, lo que es obligatorio al programar es que la función debe ser definida.

¡Logro desbloqueado!: ¡Funciones!

El futuro digital es de todos

MinTIC

GRACIAS

OPERADO POR:

Universidad Tecnológica de Pereira

MisiónTIC2022