El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

CICLO 1

EJE TEMÁTICO 4

FUNCIONES

Universidad Industrial de Santander

MisionTIC2022

FUNCIONES

Ruta de aprendizaje

Funciones

Este recurso busca dar a conocer el concepto de funciones y cómo se encuentran éstas conformadas, cuáles son sus partes principales y qué tipos de funciones podemos encontrar en Python para realizar determinada tarea.

Palabras claves: Argumentos, Funciones, Módulos, Parámetros, Retorno

Contextualización

¿Has visto alguna vez una carrera de autos de fórmula uno? Pues bien, hay un momento en la competición en la que los autos deben entrar a pits. La razón es que al auto se le debe hacer un mantenimiento a las llantas y se le debe suministrar combustibles. Ambas funciones deben llevarse a cabo luego de un determinado número de vueltas, cuando el ingeniero automovilístico encargado lo determine. Supongamos que la función general de la entrada a pits es realizar ambas tareas (cambio de llantas y suministrar combustible) ejecutadas una seguida de la otra. Así pues, una función se puede definir como una secuencia de instrucciones que tiene como finalidad llevar a cabo una tarea específica; como por ejemplo, realizar la suma de dos números, contar las palabras de una cadena de caracteres, etc.

Las funciones en programación reciben un nombre que debe ser coherente con su función. Por lo general, un programa es dividido por diferentes tipos de funciones, que llevan a cabo diferentes tipos de tareas, de tal manera que se logra la solución de un problema más grande. Adicionalmente, así como un auto de carreras entre a la zona de pits las veces que sea necesario, en un programa, las funciones son usadas las veces que se desee, esto es son estructuras de códigos reutilizables!

Introducción

En la vida diaria realizamos muchas actividades o tareas, que nos permiten desarrollarnos como seres humanos, desde que empezamos el día con la preparación de una taza de café, hasta las actividades que nos colocan en el colegio, la universidad o el trabajo. Cada una de esas actividades o tareas pueden relacionarse con el concepto de funciones en los lenguajes de programación.

Introducción

De acuerdo a John Snowden (2020) (Python For Beginners: A Practical Guide For The People Who Want to Learn Python The Right and Simple Way):

"Una función es un bloque de código que tiene asociado un nombre y que recibe o no una serie de argumentos como entrada, para luego seguir una serie de instrucciones que tienen como finalidad llevar a cabo una tarea específica y retornar un valor".

Podemos realizar una analogía entre las funciones en el entorno de programación y las tareas que realizamos a diario. De esta manera, los argumentos de entrada de una función se pueden asociar a que requieres para realizar una tarea; las instrucciones dentro de una función se pueden ver como el proceso que llevamos a cabo para resolver la tarea, y el valor que retorna una función, como el resultado que se obtiene al realizar determinada tarea.

Introducción

Ejemplo:

Si se desea realizar una taza de café se requiere **agua**, **café** y **azúcar**, que análogamente serían los **parámetros de entrada** de la función.
El café se puede realizar con **una olla** y **una estufa** o con **una máquina para hacer café**; este proceso serían las **instrucciones** de la función, junto con la aplicación del azúcar.
Como resultado se obtiene **una taza de café**, que sería lo que **retorna la función**.

Introducción

Tarea: Hacer una torta de banana y crema

No todos los problemas que nos encontramos en la vida diaria son tan sencillos como preparar una taza de café, es por esto que, una tarea puede ser dividida en sub-tareas. Así mismo, en los lenguajes de programación podemos encontrar problemas extensos que pueden ser resueltos a partir de estructuras de códigos más pequeñas, a las cuales podemos asociar funciones.

Las funciones permiten dividir un código o aplicación en pequeñas tareas que hace parte de la solución a un problema o tarea más grande.

Introducción

Existen ventajas en la estructuración de programas mediante el uso de **funciones**:

- Te permite **asignar un nombre** a un conjunto de órdenes o instrucciones, lo que hace que el programa sea más fácil de leer, entender y depurar.
- Hacen el **programa más pequeño,** ya que se elimina código repetido.
- Ayuda en la **depuración de programas largos**, ya que se puede depurar por funciones.
- Una función puede ser útil para el **desarrollo de otros programas**.

Como podemos ver, dentro del contexto de programación, las funciones cumplen un papel muy importante, facilitándonos el desarrollo de programas o aplicaciones. Es por eso, que en este curso, aprenderemos cómo desarrollarlas y cómo usarlas.

Revisión de lo aprendido

1. El primer paso es agregar el café y el agua en la máquina corresponde a:

a) Parámetros de entrada de la función

b) Instrucciones de una función

c) Retorno de la función

Revisión de lo aprendido

2. El proceso que lleva a cabo la máquina para hacer el café se relaciona con:

a) Instrucciones de una función

b) Nombre de la función

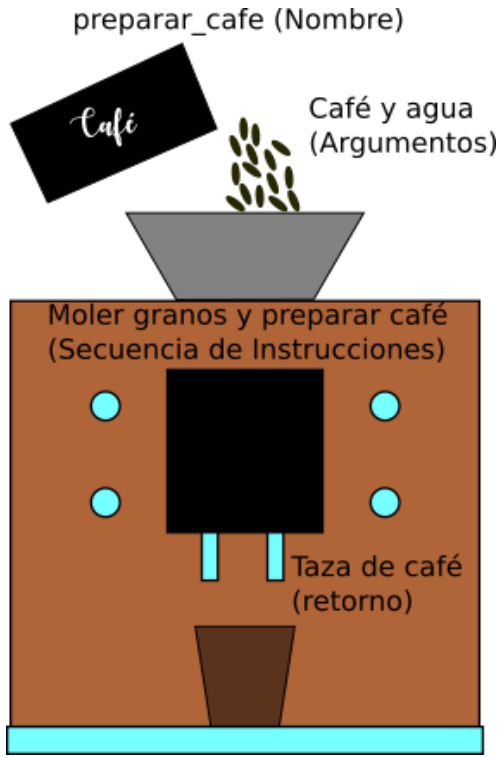
c) Retorno

Revisión de lo aprendido

3. La taza de café que se obtiene como resultado se relaciona con:

a) Retorno de la función
b) Instrucciones de una función
c) Parámetros de entrada de la función

Composición de una función



Estructura básica de una función haciendo analogía a la preparación de una taza de café.

En cualquier tipo de lenguaje de programación se pueden definir funciones usando una estructura básica. A continuación, se muestran cada uno de los elementos de esa estructura básica.

Composición de una función

Elementos de una función:

Nombre: Es un identificador que permite identificar la función. Este nombre puede estar asociado a la tarea que realiza la función.

Argumentos: Son parámetros que recibe la función y pueden ser usados para llevar a cabo la tarea para la cual fue creada. Una función puede o no recibir argumentos de entrada.

Secuencia de instrucciones: Además del nombre que identifica la función y sus argumentos, se requiere construir un algoritmo o secuencia de instrucciones que permita resolver la tarea que se requiere que lleve a cabo la función.

El retorno: El retorno de la función son los datos que se generan después de que la función lleva a cabo todas las tareas para la cual fue creada. Una función no necesariamente retorna un valor, en algunos lenguajes a las funciones que no retornan valores se les conocen como funciones vacías y en otros lenguajes simplemente retornan un valor nulo.

Composición de una función

Estructura de una función en programación:

```
define <nombre de función>(<argumento 1>, <argumento 2>, …) {
    <secuencia de instrucciones>
    retorna <resultado>
}
```

La palabra `define` es siempre una palabra reservada del lenguaje de programación que se use, la cual permite definir la función dándole un nombre, <nombre de función>, y unos argumentos (si son necesarios), `<argumento 1>`, `<argumento 2>`, etc.

Dentro de la función van, en el siguiente orden, la secuencia de instrucciones, `<secuencia de instrucciones>`, que permite realizar la tarea para la cual es creada, y el retorno, `<resultado>`. Cabe mencionar que `retorna` es también una palabra reservada del lenguaje de programación que se use.

Composición de una función

Ciclo de vida de una función:

Las funciones son **secuencias de instrucciones** que poseen una estructura definida, ellas son definidas y son implementadas en donde se requiera a lo largo del desarrollo de un código o aplicación. Para ser usadas, ellas pueden o no recibir un conjunto de parámetros de entrada, los cuales son procesados internamente y como resultado es retornado o no, correspondiente a la tarea para la cual fue creada.

Una función tiene un **ciclo de vida** bien definido, el cual empieza cuando es **declarada** y continúa con su posterior **uso** para finalmente **morir**. Luego de que una función muere, esta puede ser reutilizada las veces que sea necesaria.

Revisión de lo aprendido

1. Coloque en **orden** la estructura de una función de acuerdo a los siguientes elementos:

Se retorna un resultado

Se define la función dándole un nombre

Se suministran los parámetros de entrada

Se definen las tareas que van a ser ejecutadas por la función

Revisión de lo aprendido

2. Una función puede recibir la cantidad de argumentos de entrada que sean necesarios.

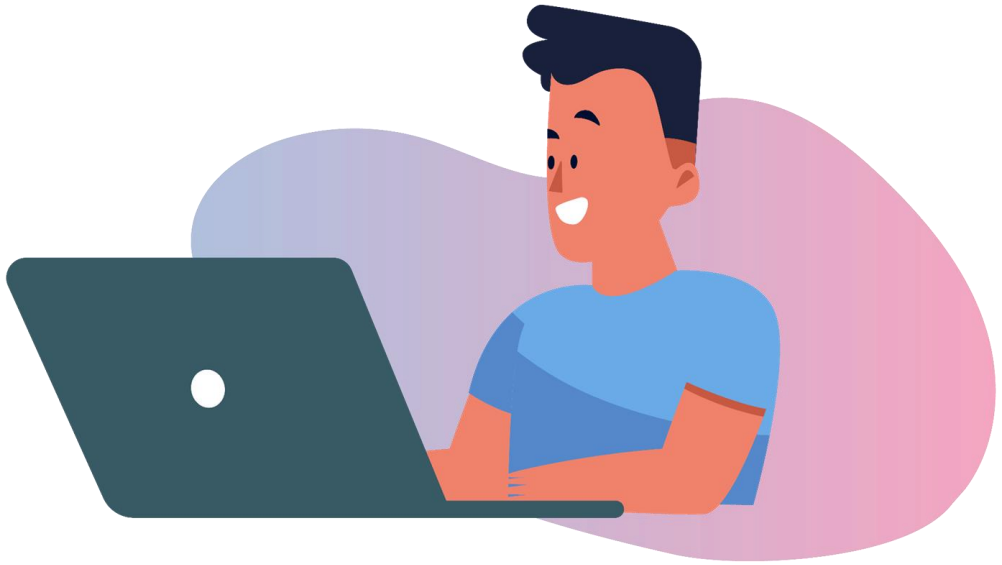
a) Verdadero.

b) Falso.

Revisión de lo aprendido

3. Complete los espacios con la información correcta

El ciclo de vida de una función empieza cuando se _____. Luego de que es declarada, la función es _____. Y posteriormente _____.

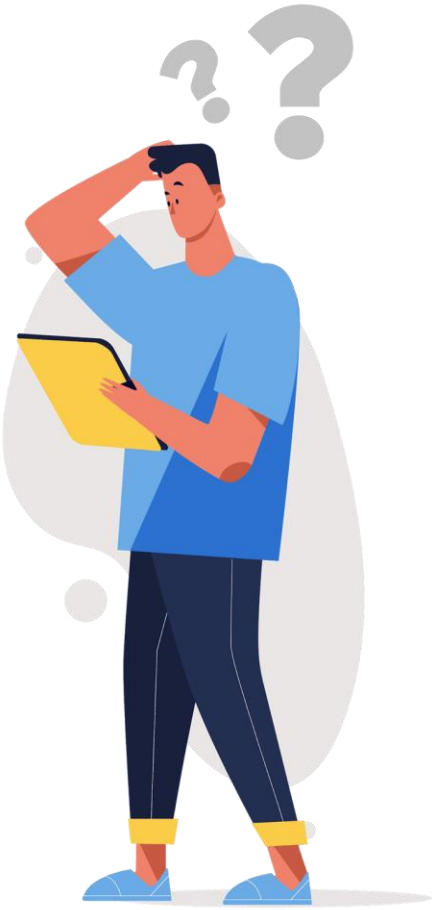


Funciones y módulos predefinidos en Python

En los lenguajes de programación, se puede definir cualquier función que se desee; pero, para facilitarnos un poco la vida existen funciones predefinidas, o sea que alguien más ya las creó y fueron incorporadas en el lenguaje de programación, en este caso **Python**.

En **Python**, se pueden encontrar dos tipos de funciones predeterminadas, las cuales son:

- **Funciones predefinidas**.
- **Los módulos predefinidos**. son archivos que contienen **métodos predefinidos** (funciones), que no pueden existir por sí solos, ya que se encuentran asociados a determinado **objeto** o tipo de dato (listas, cadenas, caracteres, etc), de tal manera que, operan sobre ellos.



Funciones y módulos predefinidos en Python.

Funciones predefinidas

Las **Funciones predefinidas** son funciones que no requieren operar sobre algún tipo de dato. Algunas de ellas son las funciones: `type`, `max`, `min`, `sorted`, `print`, `len`, `sqrt`, `input`, `help`, `dir` entre otras. Estas funciones son identificadas con un **nombre**, reciben o no **argumentos** de entrada, realizan una **tarea** especifica, y pueden o no **retornar** datos.

Usando el intérprete de Python y la función `help`, puedes ver para qué sirve cada una de estas funciones. Abre el intérprete y ejecuta la función help con cada una de las funciones predefinidas como argumento. Por ejemplo al ejecutar `help(min)` se obtiene:

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

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

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

Funciones y módulos predefinidos en Python. Módulos predefinidos

Los módulos predefinidos son archivos que contienen **métodos predefinidos** (funciones) que no pueden existir por sí solos, ya que se encuentran asociados a determinado objeto o tipo de dato, de tal manera que operan sobre ellos.

Dentro de la estructura de Python podemos encontrar archivos que contienen estructuras de datos, las cuales a su vez contienen **funciones** o **métodos** asociadas a esas estructuras. Estos archivos conforman lo que se conoce como **módulos**.

Funciones y módulos predefinidos en Python. Módulos predefinidos

Los **módulos predefinidos** en Python se pueden usar dentro de cualquier fichero, estos se pueden importar mediante el uso de la palabra reservada `import`, y se hace de la siguiente manera:

```
>>> import <nombre del módulo>
```

donde `<nombre del módulo>` es el nombre que se le da al fichero.

Si se desea importar el módulo con un nombre distinto se hace de la siguiente manera:

```
>>> import <nombre del módulo> as <otro nombre>
```

donde `<otro nombre>` es el nombre que se le quiere dar al módulo.

Funciones y módulos predefinidos en Python. Módulos predefinidos

También se pueden importar **elementos** que se encuentren dentro de los **módulos**, como por ejemplo los **submódulos** y los **métodos**. Esto se hace de la siguiente manera haciendo uso de la palabra reservada from:

```
>>> from <nombre del módulo> import <elemento 1>, <elemento 2>
```

donde `<elemento 1>`, `<elemento 2>` son los elementos que se quieren importar del módulo.
Si se desea hacer uso de algún submódulo o **método** incluido dentro del **módulo** se hace uso del **operador punto**. Esto es:

```
>>> <nombre del módulo>.<elemento>
```

donde `<elemento>` es el nombre del submódulo o método que se desea usar.
Algunos módulos predefinidos de Python son: `os`, `glob`, `sys`, `math`, `random`, `datetime`.

Revisión de lo aprendido

1. ¿Cuál es la diferencia entre un método y una función predefinida?

   a. Los métodos están asociados a un objeto o tipo de dato, las funciones no.
   b. Las funciones están asociadas a un objeto o tipo de dato, los métodos no.
   c. Las funciones retornan valores al ser ejecutadas, los métodos no.
   d. Los métodos no reciben argumentos, las funciones si.

Revisión de lo aprendido

2. ¿Cuáles es el papel que cumplen cada una de las siguientes funciones predefinidas de Python?

type ->	Imprime el objeto o tipo de dato pasado como argumento.
len ->	Determina el valor máximo de una serie de parámetros.
print ->	Determinar el tipo de dato o clase de un objeto o tipo de dato que es pasado como argumento.
max ->	Determina el número de elementos que tiene un objeto o tipo de dato.

Revisión de lo aprendido

3. ¿Qué palabras reservadas son usadas al momento de importar un submódulo o un método que se encuentre dentro de un módulo?

a. import
b. from
c. input
d. include
e. for

Revisión de lo aprendido

4. ¿De qué manera se accede al método sqrt dentro del módulo numpy?

   a. numpy.sqrt
   b. sqrt
   c. numpy.math.sqrt
   d. .sqrt

Funciones o métodos para caracteres y cadenas

Los distintos **tipos de datos** definidos en Python, poseen una serie de **métodos** que permiten manejarlos de manera más fácil. Por medio de un **tipo de dato** se puede crear un **objeto**, y este objeto por pertenecer a ese **tipo de datos** posee **métodos** o **funciones predeterminadas** que facilitan su procesamiento.

Ejemplo:

Se tienen los tipos de datos **cadena de caracteres**, los cuales poseen un **método**, `find`, que permite buscar un carácter en particular dentro de dicha cadena.

```
>>> # se crea el objeto tipo cadena de caracteres
>>> saludo="hola"
>>> # se busca dentro de este la letra "o"
>>> saludo.find("o")
1
>>> # retorna un valor entero asociado a la posicion
>>> # de latra dentro de la cadena, en este caso 1.
>>> # Retorna -1 si la letra no es encontrada.
```

Funciones para caracteres y cadenas

Existen una serie de **métodos** asociados a las **cadenas de caracteres**, estos pueden ser consultados pasando, como argumento de entrada, a la función predeterminada `dir`, un objeto tipo cadena de caracteres.

Considerando que `saludo` es un objeto tipo **cadena de caracteres**, al pasarlo como atributo de entrada a la función `dir` se obtiene una **lista** de todos los **métodos** asociados a las **cadenas de caracteres**.

```
>>> dir(saludo)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '
__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewar
gs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le
__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce_
_', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__size
of__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'coun
t', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index'
, 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'isl
ower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'lj
ust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex'
, 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith
', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

Funciones para caracteres y cadenas

Si se desea conocer cuál es la descripción de cada uno de estos **métodos** se hace uso de la función predeterminada `help`. Considerando nuevamente que `saludo` es un objeto tipo **cadena de caracteres**, se pasa como argumento de entrada, por ejemplo, el método count. Para ello se hace empleo del operador punto.

```
>>> help(saludo.count)
```

Como resultado se obtiene:

```
Help on built-in function count:

count(...) method of builtins.str instance
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
(END)
```

Revisión de lo aprendido

1. ¿Con qué método se puede encontrar en qué posición está una letra en particular?

   a. find
   b. seek
   c. found
   d. search

Revisión de lo aprendido

2. ¿Con qué método es posible saber cuántas veces aparece dentro de una cadena de caracteres una letra o una frase?

a. count
b. search
c. cont_words
d. seek

Revisión de lo aprendido

3. Si la variable values tiene almacenada una cadena de caracteres dada por '1, 2, 3, 4, 5', ¿de qué manera se puede extraer el valor de 2 como un valor entero?

a. `int(values.split(',')[1])`

b. `int(values.split(',')[2])`

c. `values.split(',')[1]`

d. `int(values.divide(',')[1])`

Funciones de pseudocódigo

Un **pseudocódigo** permite describir la manera en como opera un programa o algoritmo. Las funciones dentro de los algoritmos o códigos se pueden representar por medio de **pseudocódigos**, los cuales nos permiten **describir de manera informal** cuáles son las **funcionalidades** que caracterizan a dicha función.

Código

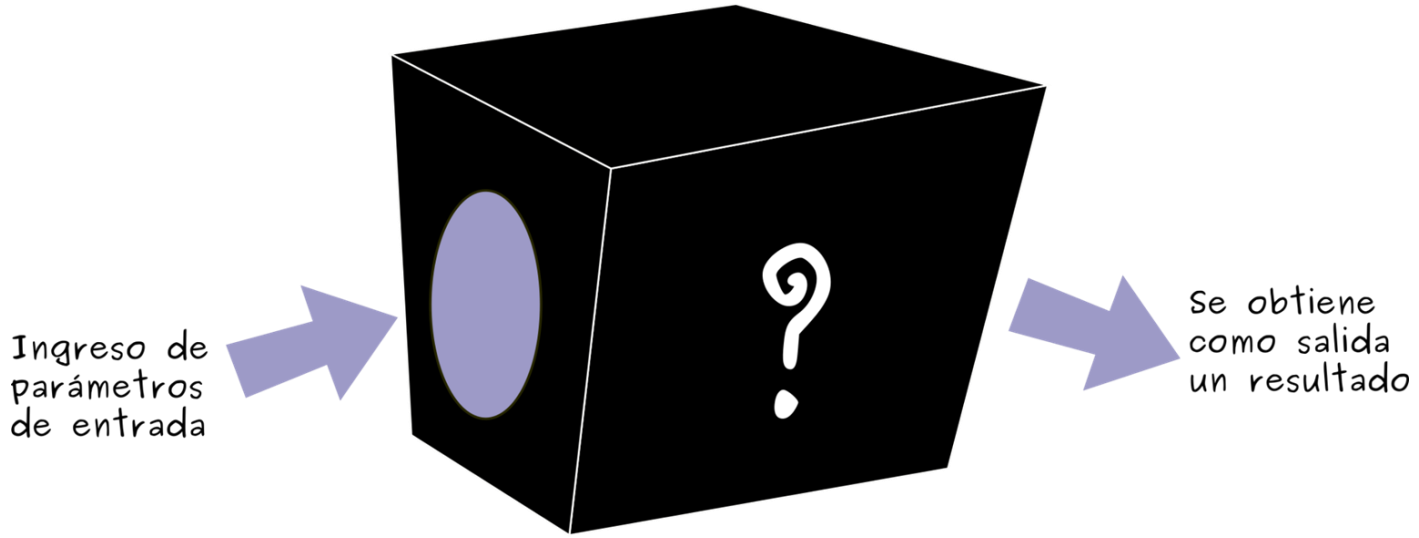
```
def is_negative_number(list_of_numbers):
    for number in list_of_numbers:
        if number < 0:
            print("The number {} is negative".format(number))
```

Pseudocódigo

```
function is_negative_number(list_of_numbers)
    for each value in list_of_numbers as number do
        if number is less than 0 then
            print that number is a negative value
```

Funciones y argumentos

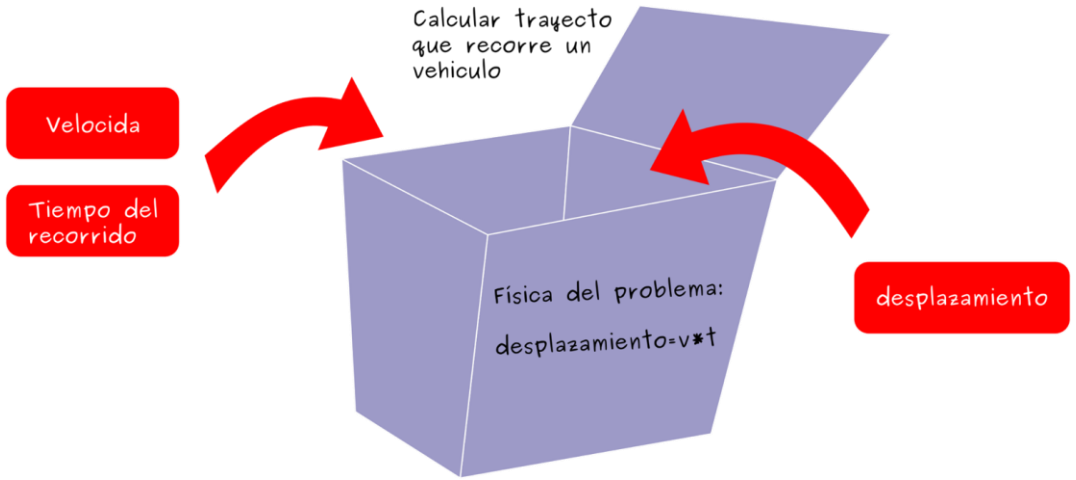
Por el momento, las **funciones predeterminadas** son unas cajas negras que sabemos que cumplen una función, pero no sabemos cómo lo hacen. No estudiaremos cómo estas **funciones predeterminadas** llevan a cabo su trabajo, pero si revisaremos cómo definimos nuestras **propias funciones**.



Funciones y argumentos

Las **funciones predeterminadas** son bastante útiles, pero son muy **genéricas**. Esto nos indica que para poder realizar alguna **tarea más particular,** hay que construir nuestra **propia función**.

Si en un programa que se esté desarrollando se requiere calcular el **trayecto que recorre un vehículo**, es necesario definir dicha función. Si por ejemplo, usamos la función `calcular_trayecto()`, esta generaría un error porque no ha sido definida previamente. Debido a que no se encuentra definida, se debe definir primero para ser implementada dentro del programa.



Funciones y argumentos.
Definición de una función en Python

Definición de una función:

Una función en Python es definida mediante la palabra reservada `def` seguida del **nombre** de la función, para luego entre paréntesis colocar los **argumentos** que esta recibirá. Al finalizar cada línea de definición de una función se colocan y en los próximos renglones, teniendo en cuenta la tabulación, la **secuencia de instrucciones** y el **retorno** cuando es necesario, haciendo uso de la palabra reservada `return`.

Ejemplos:

```
def suma(a,b):
    c = a + b
    return c
```

`suma` es el **nombre** de la función definida, `a` y `b` son sus parámetros de entrada o **argumentos**, y `c` es el **retorno**.

Funciones y argumentos

Ejemplo de una función:

Pensemos en una función hipotética para convertir unidades de velocidad, `kms_to_ms()`, que permite transformar unidades en **km/h** a unidades en **m/seg**. La conversión es simple, se requiere tener en cuenta que **1 km** son **1000 m** y que **1 h** equivale a **3600 seg**. A partir de esto, mediante regla de tres, tenemos que la conversión es la siguiente:

```
velocidad en m/seg ← (velocidad en km/h) * (1000m / 1km) * (1h / 3600 seg)
```

De aquí, la función se puede definir como:

```
def kms_to_ms(velocity_kms):
    kilometer_in_meter = 1000
    seconds_in_hour = 3600
    velocity_ms = velocity_kms * kilometer_in_meter / seconds_in_hour
    return velocity_ms
```

Funciones y argumentos. Funciones recursivas

Las **funciones recursivas** son una función que se llama a sí misma al momento de llevar a cabo una tarea, y en Python es posible definirlas.

Ejemplo:

Pensemos en la función factorial que permite determinar el factorial de determinado número, en donde el factorial de un número entero y positivo es el número multiplicado por los números que le preceden.

El factorial de `1` es `1`, el factorial de `2` es `2x1=2`, el factorial de `3` es `3x2x1=6` y así sucesivamente.

```
factorial(2) = 2*factorial(1)

factorial(3) = 3*factorial(2)

factorial(n) = n*factorial(n-1)
```

Revisión de lo aprendido

1. ¿Cuál es la tarea de los pseudocódigos?

   a. Describir la manera en como opera o funciona un programa.
   b. Convertir un código de un lenguaje a otro.
   c. No tienen una funcionalidad.
   d. Cumplen la misma tarea que cualquier otro lenguaje de programación.

Revisión de lo aprendido

2. Complete los espacios en blanco

Se desea crear una función que calcule el volumen de un líquido sabiendo su densidad y su masa. Si el volumen de un líquido se calcula dividiendo la masa del líquido por su densidad, la masa y la densidad son _____ , la división entre la masa y la densidad es la _____ y el valor que resulta de esa división es el _____.

Revisión de lo aprendido

3. Se quiere desarrollar una función que permita extraer de una cadena de caracteres dada en el siguiente formato, '(0.5, 10.0, 5.0)', la información de las coordenadas en el mismo orden pero en una lista de valores punto flotantes. ¿Cómo quedaría esta función?

a)
```
def extract_information(coord_str):
    coord = [float(value) for value in coord_str.replace('(', '').replace(')', '').split(',')]
    return coord
```

b)
```
def extract_information(coord_str):
    coord = [value for value in coord_str.replace('(', '').replace(')', '').split(',')]
    return coord
```

c)
```
def extract_information(coord_str):
    coord = [float(value) for value in coord_str.split(',')]
    return coord
```

d)
```
def extract_information(coord_str):
    coord = float(coord_str)
    return coord
```

El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

CICLO 1

EJE TEMÁTICO 4

FUNCIONES

Universidad Industrial de Santander

UIS

MisiónTIC 2022