Pensamiento Computacional

Funciones

Funciones

Ya has utilizado funciones en Python:

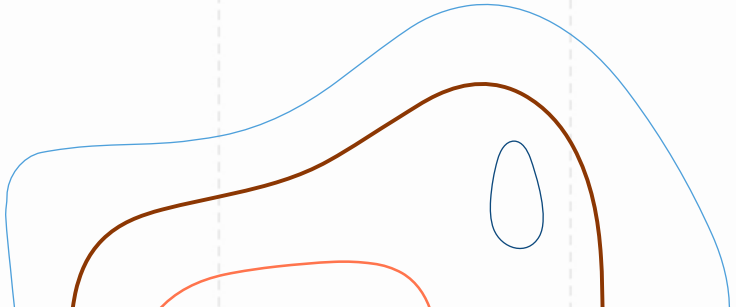
print( )

int( )

len( )

input( )

Al definir una función el código queda definido para usarlo …
Cuantas veces quieras y puedes usarlo *con diferentes valores*



Las funciones nos sirven para reutilizar código y puedes crear tus propias funciones en python…

Funciones propias del lenguaje

- Las funciones son como programas en miniatura
  - Reciben entradas
  - Procesan las entradas
  - Tienen salida

Function	Example	Input	Output
int	int(2.6) is 2	number	number
chr	chr(65) is “A”	number	string
ord	ord('A') is 65	string	number
round	round(2.34, 1) is 2.3	number, number	number

Funciones de Python

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

https://docs.python.org/3.3/library/functions.html

Funciones propias del lenguaje

- La salida de la función es un solo valor
  - Se dice que la función regresa su salida
- Los elementos dentro del paréntesis se llaman argumentos
- Ejemplos:

```
num = int(3.7)          # literal as an argument

num1 = 2.6
num2 = int(num1)        # variable as an argument

num1 = 1.3
num2 = int(2 * num1)    # expression as an argument
```

Funciones definidas por el usuario

- SINTAXIS:

```
def functionName(par1, par2, ...):
    indented block of statements()
    return expression
```

- **def** implica que se está definiendo una función, y la palabra a continuación será el nombre de la misma.
- par1, par2 son variables y se les conoce como **parámetros**
- La expresion puede ser de cualquier tipo
- El encabezado termina dos puntos :
- Cada declaración en el bloque está con la misma identación
- La función puede terminar con return y ésta instrucción regresa la variable que tiene el resultado

Ejemplos:

```
def saludo():
    print("Buenos días")

def saludo(nombre):
    print(f"Buenos días {nombre}")
```

¿Cuál es la diferencia entre estas dos funciones?

¿Cuál es la diferencia en estas tres funciones?

```
def suma(x, y):
    print(x + y)

def suma2(x, y):
    return x + y

def suma3(x, y):
    z = x + y
    return z
```

Puedes programar tus funciones para que:

1. Impriman el resultado dentro de la función
2. Regresen una operación
3. Regresen una variable resultado de una operación

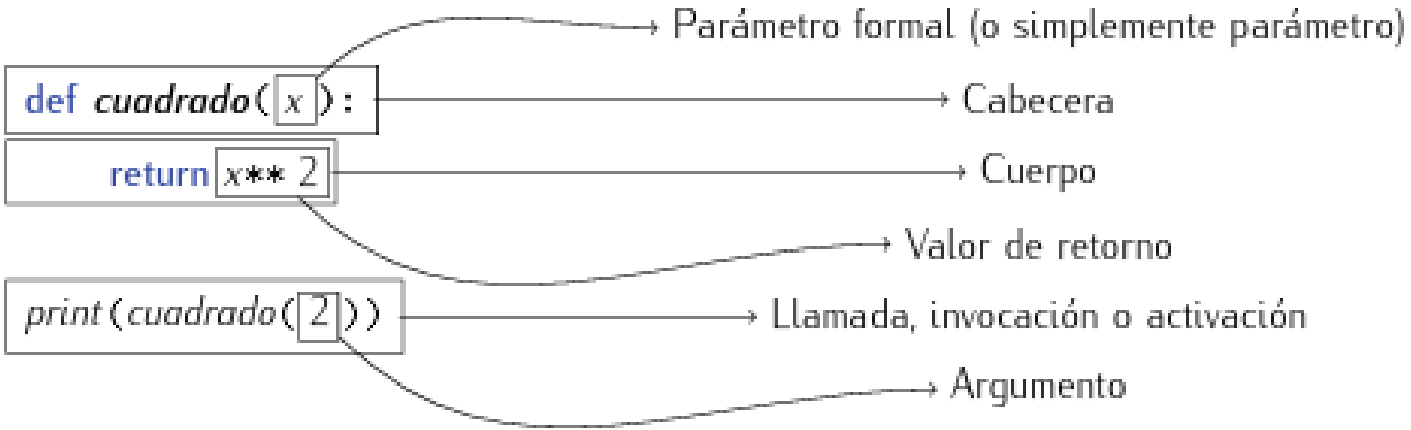
Y dependiendo de cómo la programes, es cómo se manda llamar.

¿Cómo defino una función?

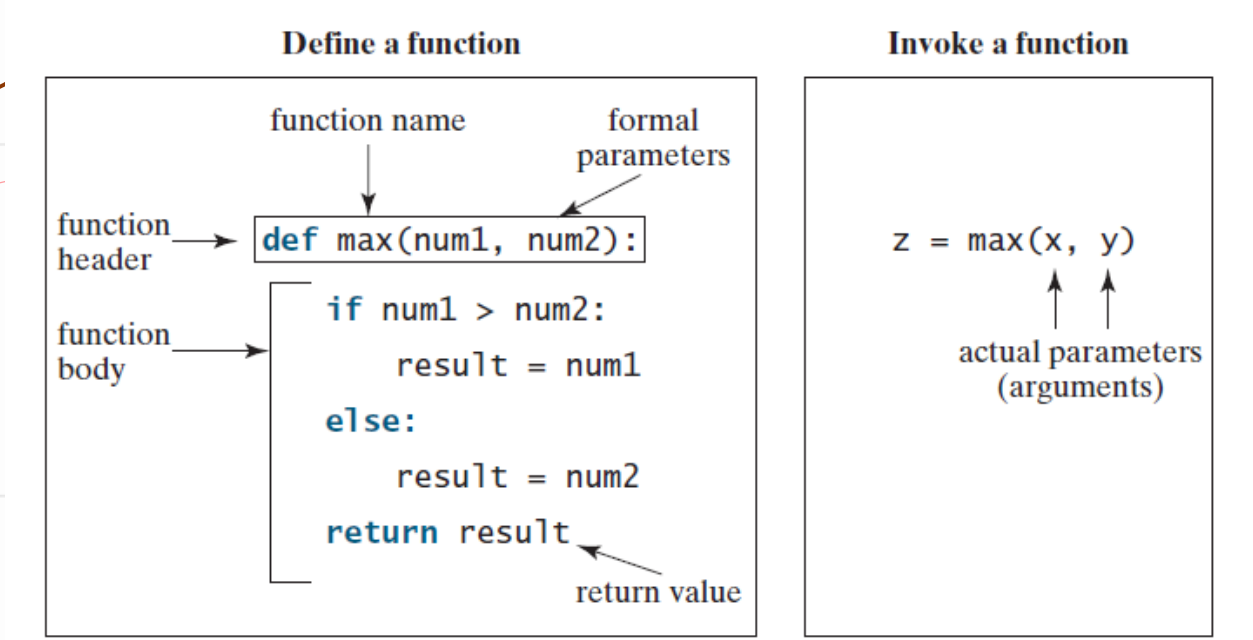
```
def nombreFuncion (parametro1, parametro2):
    #Acciones que realiza
    return/print resultado
```

```
def cuadrado(x):
    resultado=x**2
    return resultado
```

Anatomía de una función



Anatomía de una función



Funciones

- Algunos conceptos que debemos identificar
  - Definición de función
  - Parámetros / Argumentos
  - Valor de retorno
  - Funciones que devuelven un valor.
  - Funciones que realizan sólo acciones (conocidos también como procedimientos)
  - Llamada a una función
  - “Hacernos cargo del valor de retorno”

Invocando/ llamando una función…

- Existen funciones que regresan valores
- Se invocan en el programa principal, generalmente usando ***asignaciones***

```
large = max (a, b)
```

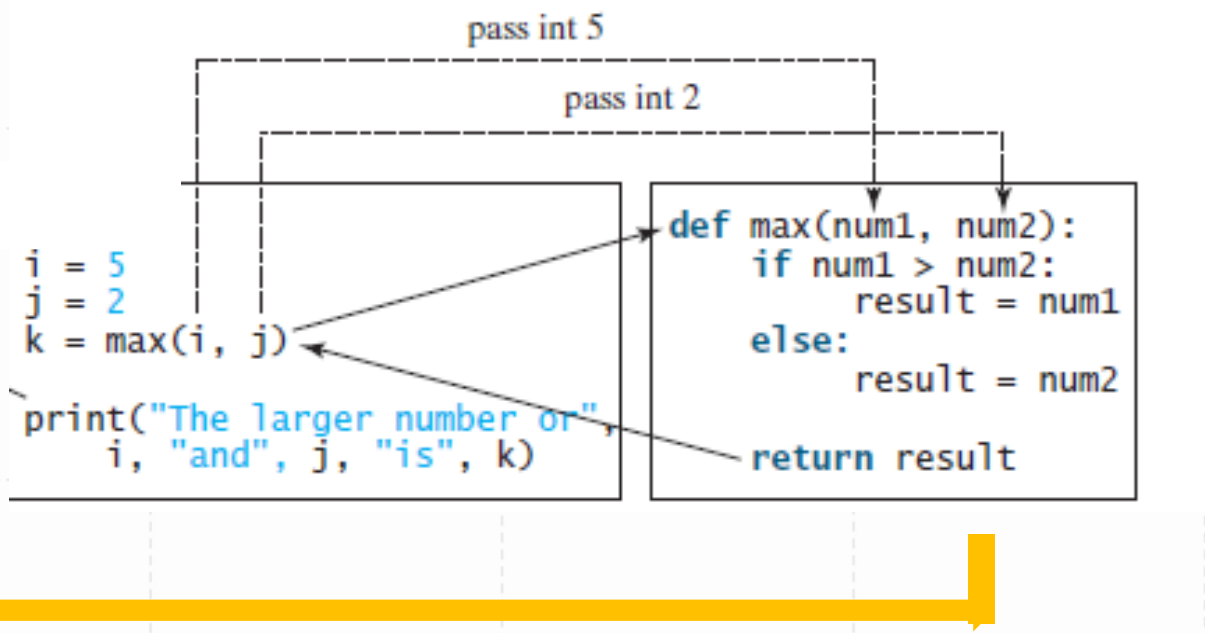
“Nos hacemos cargo del valor de retorno recibiéndolo en la variable large”

- Otras funciones NO regresan valores
- Se invocan como una instrucción

```
print(“Hola Mundo!”)
```

“Realizan sólo acciones, no las igualo a ninguna variable”

Llamando a una función



Función para sumar dos números

ciones.py | descuentos.py * | mayor.py

```
1  def cual_es_mayor(x, y):
2      if x>y:
3          return x
4      else:
5          return y
6
7  #PROGRAMA PRINCIPAL
8  num1=int(input("Dame un número :"))
9  num2=int(input("Dame otro número para ver cuál es el mayor: "))
0  mayor=cual_es_mayor(num1, num2)
1  print(f"El mayor de los número es {mayor}")
```

1

2

3

4

5

```
Dame un número :45
Dame otro número para ver cuál es el mayor: 34
El mayor de los número es 45
>>>
```

1. Se define la función
2. Se capturan los datos para la función
3. Se llama a la función
4. Se utiliza su resultado
5. Se corre el programa

Funciones vacías (void)

```
# Print grade for the score
def printGrade(score):
    if score >= 90.0:
        print('A')
    elif score >= 80.0:
        print('B')
    elif score >= 70.0:
        print('C')
    elif score >= 60.0:
        print('D')
    else:
        print('F')
```

```
score = eval(input("Enter a score: "))
print("The grade is ", end = "")
printGrade(score)
```

Cuando se llama a la función “printGrade”
la función imprime directamente el valor de la calificación y no regresa ningún resultado.

Estas funciones se conocen cómo funciones vacías o void

Anatomía de esta función

```
# Print grade for the score
def printGrade(score):
    if score >= 90.0:
        print('A')
    elif score >= 80.0:
        print('B')
    elif score >= 70.0:
        print('C')
    elif score >= 60.0:
        print('D')
    else:
        print('F')
```

Cuando se llama a la función `printGrade` la función imprime directamente el valor de la calificación a pantalla y NO regresa ningún resultado.

Este tipo de funciones realizan una acción. En este caso imprimir a pantalla un resultado.

Se invocan o llaman de esta manera:

- `printGrade(valor)` --- valor debe ser una variable con un dato asignado.
- `printGrade(50)`

Funciones que regresan valores

La función obtenerLetra regresa cada vez un valor

La impresión se hace en la función principal

```
"""
Programa que regresa la calificación equivalente en letra
REalizado por:
"""
def obtenerLetra(calif):
    if calif>=90.0:
        return "A"
    elif calif>=80.0:
        return "B"
    elif calif >=70.0:
        return "C"
    elif calif >=60.0:
        return "D"
    else:
        return "F"

#Inicia programa principal
#Recibo la calificación en número
numero=float(input("Dame tu calificación "))
#mando llamar a la función con número como parámetro
#recibo el valor de retorno de la función en califLetra
califLetra=obtenerLetra(numero)
#imprimo el valor de califLetra con un mensaje al usuario
print(f"Tu calificación en EU sería {califLetra}")
```

Shell

```
Dame tu calificación 56.67
Tu calificación en EU sería F
>>>
```

Valor None

- ¿Qué significa?
- ¿En qué caso una función nos devolverá un valor None?
- Probemos…
- Escribe en archivo nuevo de Python las dos funciones de suma de la diapositiva 8 y debajo de ellas escribe:
- suma(3,7) y corre el programa
- Posteriormente cámbiala por la siguiente instrucción: print(suma(3,7)) ¿Cuál es el resultado? ¿Por qué el resultado es diferente?
- ¿Cómo podemos probar las funciones suma2 y suma3?

Anatomía de esta función

- Cuando se llama a ésta otra función **getGrade** la función **regresa** un resultado**. Observa qué es diferente…**

- En este tipo de funciones se requiere que nos "hagamos cargo del valor de retorno"

- Se invocan o llaman de esta manera:
  - `letter_grade = getGrade(valor)` --- valor debe ser una variable con un dato asignado.
  - `letter_grade = getGrade(50)`
- O directamente "haciendo algo" con el valor de retorno:
  - `print(getGrade(numero))`

```
# Return the grade for the score
def getGrade(score):
    if score >= 90.0:
        return 'A'
    elif score >= 80.0:
        return 'B'
    elif score >= 70.0:
        return 'C'
    elif score >= 60.0:
        return 'D'
    else:
        return 'F'

#Programa Principal
score = eval(input("Enter a score: "))
print("The grade is", getGrade(score))
```

IMPORTANTE: Variables locales

- Las variables que creas dentro de una función se les denomina variables locales.
- Las variables locales sólo existen dentro de la función. Fuera de esta no pueden ser utilizadas, no se conocen.

Paso por valor

- Todo en Python es un objeto
- Cuando se pasa un argumento, en realidad se usa una referencia al objeto
- Al final, se usa el valor del objeto pero éste no se modifica cuando se ejecuta la función

```
def increment(n):
    n += 1
    print("\tn inside the function is", n)

x = 1
print("Before the call, x is", x)
increment(x)
print("After the call, x is", x)
```

```
Before the call, x is 1
        n inside the function is 2
After the call, x is 1
```

A programar

Ejercicio de comprensión:

- En un archivo nuevo, crea las siguientes funciones:

1. **Función segundos**, que recibe una cantidad de segundos e imprime a pantalla, cuántos días, horas, minutos y segundos puede formar.
2. **Función convierte_pies_cm**, que recibe un parámetro, correspondiente a una cantidad de pies. La función **regresa** el equivalente a cm de los pies recibidos. Por ejemplo:
   - convierte_pies_cm(34)
   - Deberá devolver el valor de 34 pies a su equivalente en cm: 1036.32
3. **Funcion volumen_esfera,** la cual recibe el radio de la esfera en cm y nos devuelve el valor de su volumen.

Ejercicios

4. **Función multiplo_de** que recibe dos datos: el primero es el número a comprobar, el segundo es el divisor. La función debe devolver *True*, si el primer número es múltiplo del segundo y *False* en caso contrario.

5. **Función compara.** Escribe la función compara, la cual recibe dos números. La función debe devolver:
   - -1 si el primer argumento es menor al segundo
   - 0 si los dos números son iguales
   - 1 si el primer argumento es mayor al segundo.

Ejercicios

- Una vez que tengas las 5 funciones crea un solo programa que mande llamar a las 5 funciones secuencialmente o a través de un menú.
- Cuida solicitar las entradas y enviarlas a la función como parámetro.
- Si la función regresa algo no olvides usar return y encargarte del valor de retorno.

El programa debe tener todas las funciones en la parte superior del código.

funciones.py | descuentos.py * | mayor.py

```
"""
Programa que contiene 5 funciones
"""

#Función segundos
def segundos(s):
    dias=s//86400
    ss=s%86400
    horas=ss//3600
    ss=ss%3600
    min=ss//60
    ss=ss%60
    print (f"{s} segundos equivale a:\n{dias} días,\n{horas} horas,\n{min} minutos,\n{ss} segundos")

#Función convierte pies a metros
def pies_a_cm(p):
    return (p*30.48)
```

Thonny - /Users/AnaRaquel/OneDrive-Tec/OneDrive - Instituto Tecnologico y de Estudios Superiores de Monterrey/MateriasTEC/Thonny/funciones.py @ 34 : 76

funciones.py | descuentos.py * | mayor.py

```
23 print(" MENU DE FUNCIONES \n ")
24 print("1. Segundos a días, horas, minutos")
25 print("2. Pies a centímetros")
26 menu=int(input("Elección: "))
27 if menu==1:
28     segs=int(input("Dame la cantidad en segundos a convertir en días, horas y minutos "))
29     segundos(segs)
30 elif menu==2:
31     pies=float(input("Dame los pies a convertir en centímetros: "))
32     cms=pies_a_cm(pies)
33     print(f"{pies} pies equivale a {cms:.3f} cm")  #limito a 3 decimales
34 else:       #yo sólo programé dos opciones, deben ser las 5 y un else final
35     print ("no existe esa opcion ")
36
37
38
```

En el programa principal, haces el llamado a cada función a través de un menú

Shell

```
>>>
>>> %Run funciones.py
   MENU DE FUNCIONES

 1. Segundos a días, horas, minutos
 2. Pies a centímetros
 Elección: 2
 Dame los pies a convertir en centímetros: 45
 45.0 pies equivale a 1371.600 cm

>>>
```

Termina el reto de las 5 funciones apoyado en éste material.

Assistant

The code in funciones.py looks good.

If it is not working as it should, then consider using some general debugging techniques.

Was it helpful or confusing?

Sólo por practicar:

1. Realizar un programa que lea dos números y mediante funciones despliegue su suma, su resta, su multiplicación, su división y su residuo. Valida entradas.
2. Realizar un programa que lea el ancho y largo de un rectángulo y mediante funciones calcule su área y su perímetro.
3. Realizar un programa que lea el número de millas y mediante una función las convierta a kilómetros.
4. Dado un monto de compra, calcular el descuento.

    3. Compras mayores a $1000 el descuento es de 20% , compras menores a $1000, el descuento es de 10%. Utiliza funciones. Pide la cantidad y despliega descuento y cantidad final a pagar.

5. Realiza una función que recibe dos argumentos, la base y la potencia y devuelva el resultado.

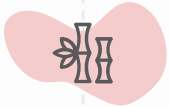
Sólo por practicar:

5. Escribir un programa que pida como entrada la temperatura en grados Celsius, muestre un menú para convertirlos a Fahrenheit o Kelvin y utilizando funciones, calcule los resultados. Muestra las equivalencias en pantalla.
6. Crea un programa que reciba 3 números por el teclado y utilizando funciones, los muestre ordenados de forma ascendente ó descendente.
7. Realiza una función que reciba el año de nacimiento y a partir de ello calcule la edad de la persona (sin validar mes y año)
8. Escribir un programa, que con funciones, verifique si un caracter introducido es un dígito o no.
9. Escribir un programa que mediante el llamado a una función resuelva una ecuación cuadrática, dados los coeficientes del polinomio.

Qué aprendimos...

Función

Un programa en chiquito que puedo mandar ejecutar cuantas veces lo requiera en el programa.

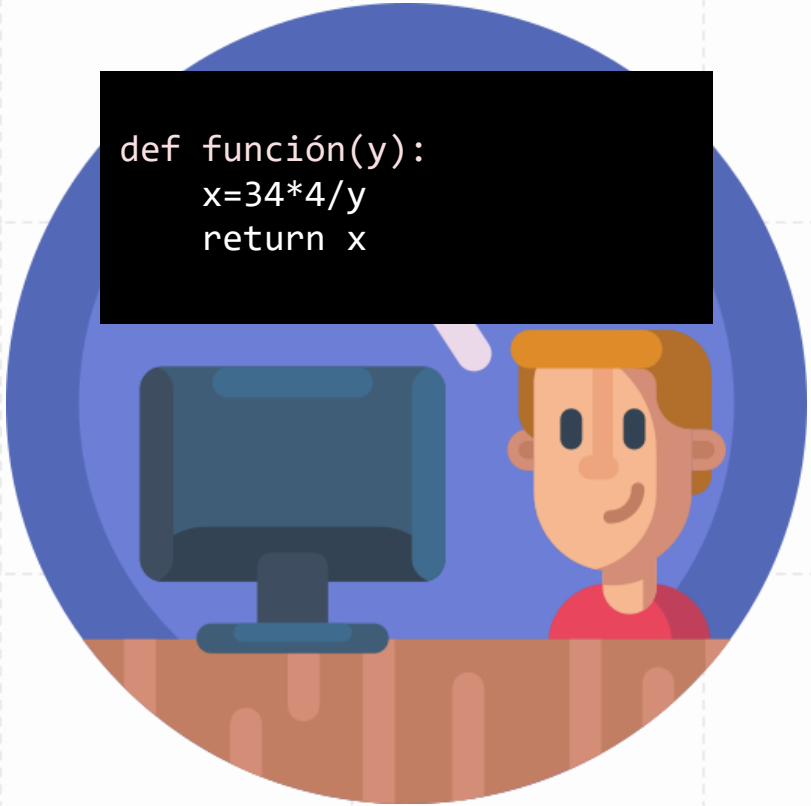


Anatomía

Una función puede regresar un valor y entonces tengo que recibirlo en el programa principal, o puede no regresar nada y entonces sólo la mando llamar.

Posición

Las funciones en Python se declaran al inicio del programa.


def función(y):
x=34*4/y
return x

Recursos recomendados

- Videos sobre funciones desarrollado por Yolanda Martínez.
  - https://youtu.be/gmEiJG7GhqU
  - https://youtu.be/BcIc4cY9Dzc