

INICIOS DE PROGRAMACIÓN EN PYTHON




Brayan Steven Burgos Delgado

brayan.burgos@mail.escuelaing.edu.co

brayanburgos1437@gmail.com



AGENDA

- Funciones
 - Recurrencia
 - Turtle en Python
 - Referencias
- 



PROGRAMACIÓN MODULAR (FUNCIONES)

- Esta técnica se conoce como: DIVIDE y VENCERAS
- Un módulo es un programa, llamado función. Es código en Python que permite llevar a cabo una tarea específica.
- Las funciones tienen la capacidad de comunicarse entre si, con el objetivo de compartir los resultados.

VENTAJAS

- Por la independencia de las funciones, un programa se puede diseñar con mayor facilidad.
- Se puede modificar una función sin afectar a las demás.
- Las funciones solamente se escriben una vez, aunque se necesiten en distintas ocasiones a lo largo del programa. Esto permite la reutilización de código en nuestros programas.

FUNCIONES

7

>>>

- Es un método para organizar, potencializar y reutilizar nuestro código en Python. Modularidad es el termino preciso, con el cual realizaremos tareas específicas para CONQUISTAR.

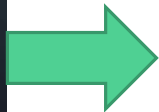
Palabra reservada def

Nombre la función

Parámetros

```
def funtionsumar(numero1, numero2):  
    #utilizaremos la funcion para sumar dos numeros  
    respuesta=numero1+numero2  
    return respuesta #return  
print(funtionsumar(2, 5))
```

TAB



LLAMADO A LA FUNCION (funtionsumar)

RETURN

- Palabra reversada en el lenguaje y es un función con variados usos
- Sus característica es retornar la función y en algunas ocasiones el dato que la función le entregue.
- Su característica es que sirve de caso de escape o caso base para que la función se detenga y entregue el dato

- Al día de hoy, ya hemos utilizado funciones definidas en Python, como por ejemplo: len, range, chr, upper, lower, ord, int, str,.....
- Para el uso de estas funciones se deben especificar de forma CORRECTA los parámetros requeridos: len("casa")

Función llamada saludo

```
def saludo():  
    print("Buen día")  
saludo()
```

```
>>>  
Buen día
```

```
def saludo():  
    print("Buen día")  
  
for x in range(1,6):  
    saludo()
```

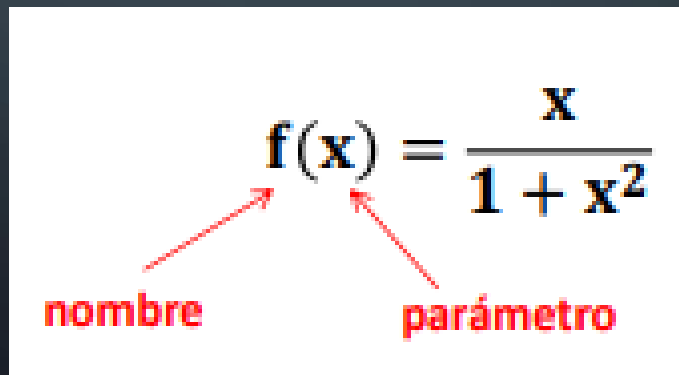
```
>>>  
Buen día  
Buen día  
Buen día  
Buen día  
Buen día
```

```
def saludo(nombre):  
    print("Buen día", nombre)  
  
for x in range(1,6):  
    print("Saludando")  
    n="Juan" + str(x)  
    saludo(n)
```

```
>>>  
Saludando  
Buen día Juan1  
Saludando  
Buen día Juan2  
Saludando  
Buen día Juan3  
Saludando  
Buen día Juan4  
Saludando  
Buen día Juan5
```

PARÁMETROS

- Es un valor que la función espera recibir cuando sea invocada.
- Una función puede esperar cero, uno o más parámetros, los cuales están separados por coma.
- Los parámetros que una función espera, serán utilizados por ésta, dentro de su algoritmo, a modo de variables de ámbito local.



The diagram shows the function notation $f(x) = \frac{x}{1 + x^2}$. A red arrow points from the word 'nombre' to the 'f' in the function name. Another red arrow points from the word 'parámetro' to the 'x' inside the parentheses.

$$f(x) = \frac{x}{1 + x^2}$$

nombre parámetro

DOCUMENTACIÓN DE LAS FUNCIONES

- Con el animo de que la programación se haga de forma profesional, existe una manera que nuestros programas sean organizados y otros programadores puedan entender nuestros códigos
- Tres “ al comienzo y al final permitirán que cada función quede documentada

```
"""PRE: dos numeros que el usuario ingresara
POS: retornara la suma de los dos numeros"""
def funtionsumar(numero1,numero2):
    #utilizaremos la funcion para sumar dos numeros
    respuesta=numero1+numero2
    return respuesta #return
"""PRE: ninguna
POS: mostrara en pantalla la suma, por medio de
funtionsumar(numero1,numero2) """
def main():
    num1=int(input())
    num2=int(input())
    print(funtionsumar(num1,num2))
main()
```



Documentación

FUNCIONES RECURSIVAS

- Son funciones que se invocan a sí mismas en su definición.
- Debe existir un caso base o condición de escape que permite terminar la recursión.
- El uso es de mucho cuidado, puesto que si no se detiene la computadora termina sobrecargada.

```
def factorial(n):  
    if n==0:  
        return 1  
    else:  
        return n*factorial(n-1)  
def main():  
    n=int(input())  
    print(factorial(n))  
main()  
|
```

```
----- RESTART: C:/Users/mipc/Documents/fails/pp.py -----  
2  
2  
>>>  
===== RESTART: C:/Users/mipc/Documents/fails/pp.py =====  
5  
120  
>>>  
===== RESTART: C:/Users/mipc/Documents/fails/pp.py =====  
20  
2432902008176640000  
>>>  
===== RESTART: C:/Users/mipc/Documents/fails/pp.py =====  
40  
8159152832478977343456112695961158942720000000000  
>>>
```

```
from sys import stdin
p=int(stdin.readline())
"""PRE:Fibonacci consiste en la ecuacion de recurrencia  $((an-1)*3)+(an-2)-(an-3)$ 
necesita una entrada de un entero
POS: entregara el numero de fibonacci del numero ingresado"""
def bio (n):

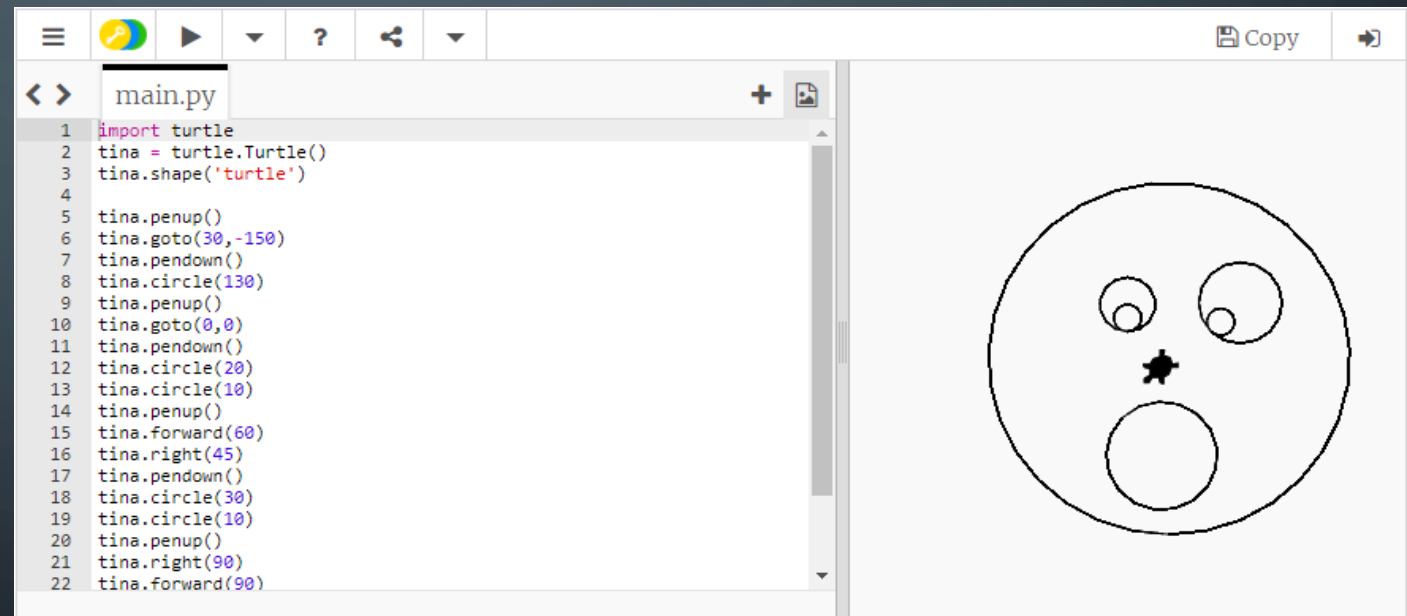
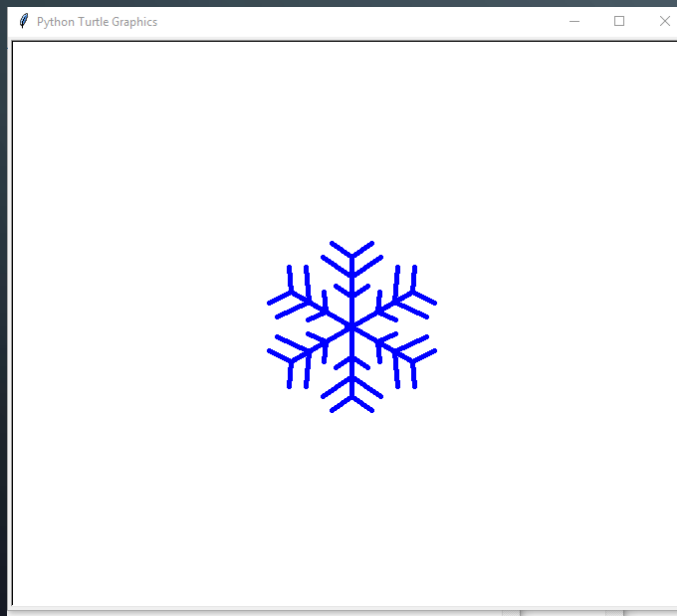
    if n==0:
        return 1
    if n==1:
        return 2
    if n==2:
        return 5
    else:
        return (bio(n-1)*3)+(bio(n-2)-bio(n-3))
print(bio(p))
```

CASOS BASE

Ecuación recurrente

TURTLE EN PYTHON

- La librería Python es la mas divertida de todo el programa, ya que permite programar dibujando
- Presentaremos diferentes usos de esta librería y como se usa de forma



VAMOS A JUGAR CON TURTLE

- <https://hourofpython.trinket.io/a-visual-introduction-to-python#/welcome/an-hour-of-code> guías para ver lo que se logra con TURTLE
- <https://docs.python.org/3.0/library/turtle.html> documentación para aprender a usar las funciones en Python con la LIBRERÍA TURTLE

REFERENCIAS

- <https://docs.python.org/3.0/library/turtle.html>
- <https://hourofpython.trinket.io/a-visual-introduction-to-python#/welcome/an-hour-of-code>
- https://www.ibm.com/thought-leadership/innovation_explanations/article/bob_schultz-cognitive-human-resources.html?lnk=ushpv18f1&lnk2=learn