Clase 6 – Recursión

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

12-09-2019

Resumen de la clase

- Las funciones en Python:
  - Pueden recibir parámetros
  - Hacen un cálculo o ejecución de cierto código
  - Devuelven un valor

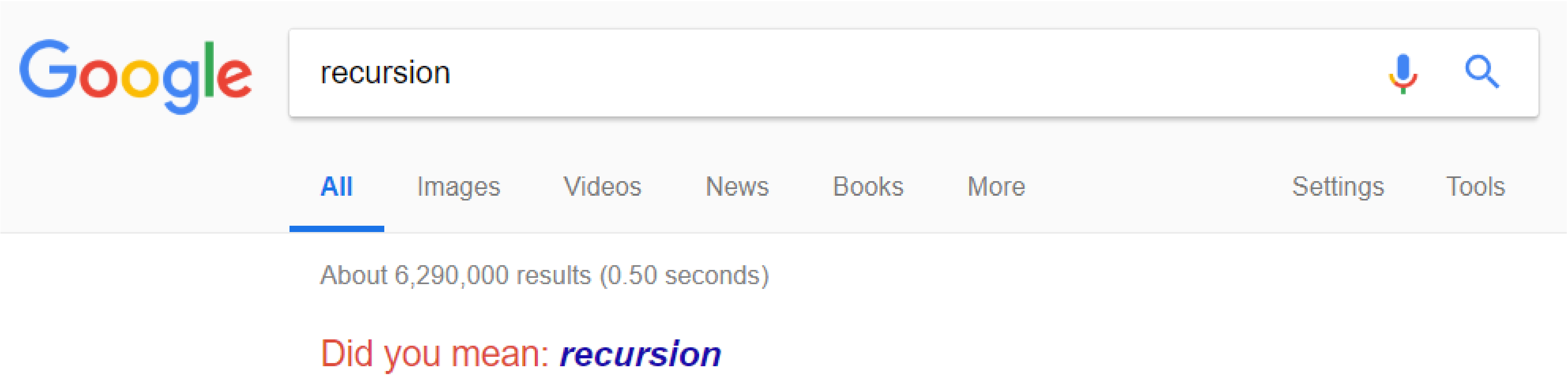
Parametro(s) → Función → Valor retornado

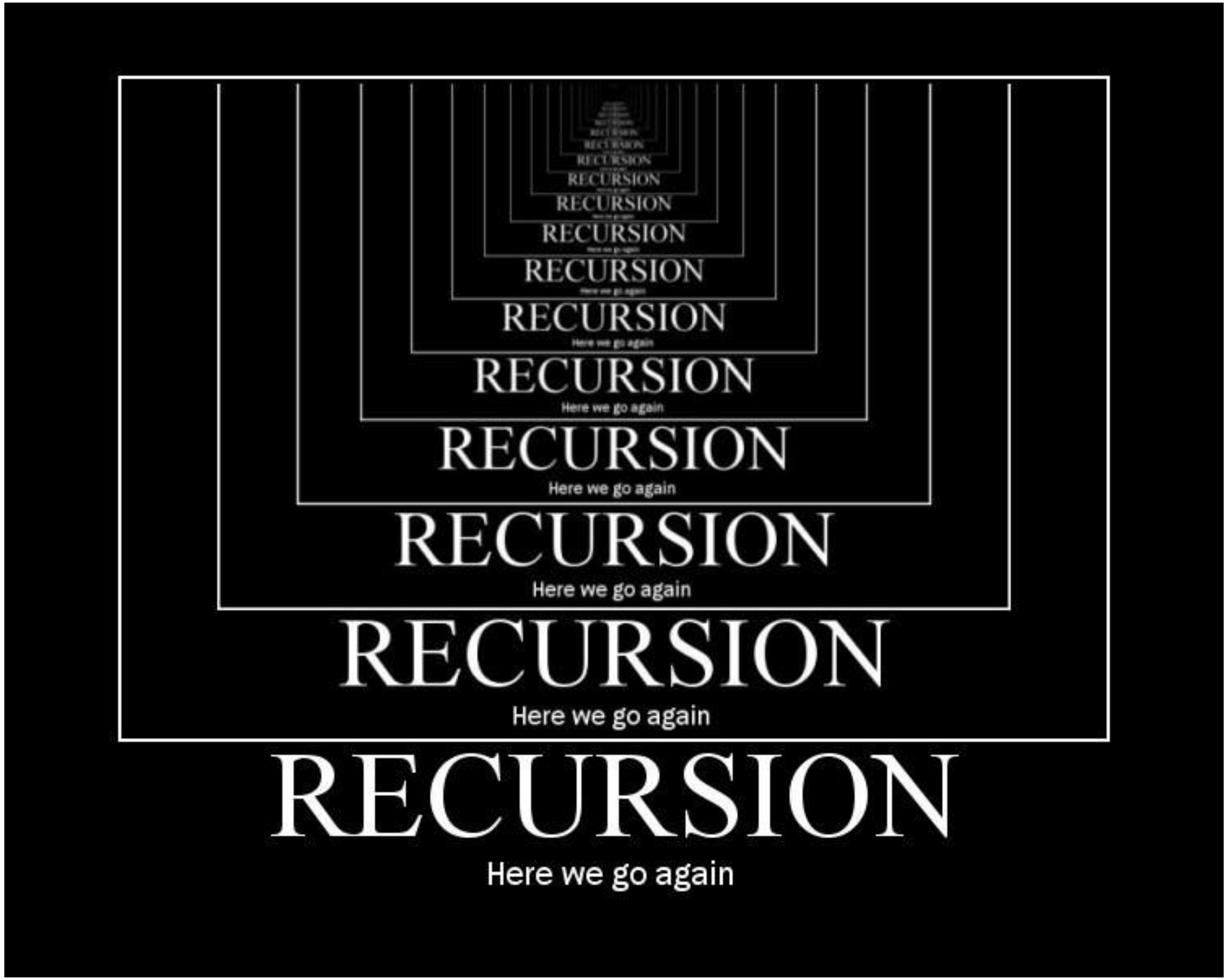
- ¿Cómo podemos definir una función en Python?

```
def nombre_funcion(a,b,c):
    #ejecución de cierto código
     return (valor retornado)
```

Contenidos

1. Recursión
2. Ejercicios


Google
recursion
All
Images
Videos
News
Books
More
Settings
Tools
About 6,290,000 results (0.50 seconds)
Did you mean: recursion


RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
RECURSION
RECURSION
RECURSION
RECURSION

“Es legal para una función llamar a otra. También es legal a una función llamarse a si misma. Puede no ser obvio por qué es algo bueno, pero resulta ser una de las cosas más mágicas que un programa puede hacer”

(Think Python, Downey 2013)

Recursión

- ¿Qué pasa si ejecutamos la siguiente función?

```python
def ejemplo():
    print("hola")
    ejemplo()
```

Recursión

- ¿Qué pasa si ejecutamos la siguiente función?

```
def ejemplo(param):
    print(param)
    ejemplo(param+1)
```

Recursión

- En ambos casos anteriores, las iteraciones eran infinitas.
- ¿Podríamos definir alguna condición para que esto termine?

Recursión

- En ambos casos anteriores, las iteraciones eran infinitas.
- ¿Podríamos definir alguna condición para que esto termine?

```python
def ejemplo(param):
    if param == 10:
        return
    print(param)
    ejemplo(param+1)
```

Recursión

- ¿Cómo podríamos definir a la recursión? En base a dos elementos
  - Caso base:
    - Donde termina la recursión.
  - Caso recursivo:
    - La llamada de la función a si misma.

Recursión

- ¿Cómo podríamos definir a la recursión? En base a dos elementos
    - Caso base:
        - Donde termina la recursión.
    - Caso recursivo:
        - La llamada de la función a si misma.

No queremos que sea una iteración (aunque sea recursivamente) infinita

Recursión

- ¿Cómo podríamos definir a la recursión? En base a dos elementos
  - Caso base:
    - Donde termina la recursión.
  - Caso recursivo:
    - La llamada de la función a si misma.
- ¿Es posible representar una iteración (o ciclo) como una recursión?

No queremos que sea una iteración (aunque sea recursivamente) infinita

Recursión

- ¿Cómo podríamos definir a la recursión? En base a dos elementos
  - Caso base:
    - Donde termina la recursión.
  - Caso recursivo:
    - La llamada de la función a si misma.
- ¿Es posible representar una iteración (o ciclo) como una recursión?
- **Sí**

No queremos que sea una iteración (aunque sea recursivamente) infinita

Recursión

- Ejemplo:

```python
def for_simulado(i,n):
    if i > n:
        return
    print(i)
    for_simulado(i+1,n)

for_simulado(0,10)
```

Recursión

- Ejemplo:

```
def for_simulado(i,n):
    if i > n:
        return
    print(i)
    for_simulado(i+1,n)

for_simulado(0,10)
```

¿Y si quisiéramos hacer una sumatoria $\sum k$?

Recursión

¿Y si quisiéramos hacer una sumatoria $\sum k$?

```
def for_simulado(i,n,k):
    if i > n:
        return k
    return for_simulado(i+1,n,k+i)

res = for_simulado(0,10,0)
print(res)
```

```
55
```

Recursión

- ¿Qué elementos podríamos agregar a la definición en base al ejemplo anterior?

Recursión

- ¿Qué elementos podríamos agregar a la definición en base al ejemplo anterior?
  - **Caso base:**
    - Donde termina la recursión y devolvemos un valor específico.
  - **Caso recursivo:**
    - La llamada de la función a si misma.
  - **Parámetros:** que le entregamos a la función al llamarla por primera vez.
  - **Reducción de un problema:** donde el caso más simple es el caso base.

Recursión

Ejemplo: Sumatoria $\sum_1^n k$

Recursión

Ejemplo: Sumatoria $\sum_1^n k$

- **Caso base:**
- **Caso recursivo:**
- **Parámetros:**
- **Reducción de un problema:**

Recursión

Ejemplo: Sumatoria $\sum_{i=1}^{n} k$?

- **Caso base:**
  - k = 1.
- **Caso recursivo:**
- **Parámetros:**
- **Reducción de un problema:**

Recursión

Ejemplo: Sumatoria $\sum_1^n k$?

- **Caso base:**
  - k = 1.
- **Caso recursivo:**
  - Sumo n + $\sum_1^{n-1} k$
- **Parámetros:**
- **Reducción de un problema:**

Recursión

Ejemplo: Sumatoria $\sum_{1}^{n} k$?

- **Caso base:**
  - k = 1.
- **Caso recursivo:**
  - Sumo n + $\sum_{1}^{n-1} k$
- **Parámetros:** n
- **Reducción de un problema:**

Recursión

Ejemplo: Sumatoria $\sum_1^n k$?

- **Caso base:**
  - k = 1.
- **Caso recursivo:**
  - Sumo n + $\sum_1^{n-1} k$
- **Parámetros:** n
- **Reducción de un problema:** ¿Cuál es el caso más simple?

Recursión

Ejemplo: Sumatoria $\sum_1^n k$?

- **Caso base:**
  - k = 1.
- **Caso recursivo:**
  - Sumo n + $\sum_1^{n-1} k$
- **Parámetros:** n
- **Reducción de un problema:** ¿Cuál es el caso más simple? El caso base. Entonces vamos reduciendo el problema hasta que llegamos a algo que sabemos. Es decir, $\sum_1^1 k = 1$

Recursión

Ejemplo: Sumatoria $\sum_1^n k$?

```python
def sumatoria(n):
    if n == 1:
        return 1
    else:
        return sumatoria(n-1)+n

print(sumatoria(6))
```

```
21
```

Recursión

Ejemplo: Factorial

Recursión

Ejemplo: Factorial $n!$

Recursión

Ejemplo: Factorial $n!$ $\Rightarrow$ $\prod_1^n k$

Recursión

Ejemplo: Factorial $\prod_1^n k$

- **Caso base:**
  - ¿Dónde termina la recursión? k=10
  - ¿Qué devolvemos? k
- **Caso recursivo:**
  - La llamada de la función a si misma.
- **Parámetros:**
  - Límite inferior = 1
  - K (parte en 1)
  - Límite superior (n) = 10
- **Reducción de un problema:** Donde el caso más simple es el caso base.

Recursión

Ejemplo: Factorial

```
def factorial(i,n,k):
        if i > n:
                return k
        return factorial(i+1,n,k*i)

res = factorial(1,10,1)
print(res)
```

```
3628800
```

- ¿Es posible definir una función para calcular el factorial de un número de otra forma a la vista anteriormente?

- ¿Es posible definir una función para calcular el factorial de un número de otra forma a la vista anteriormente?
- No tiene sentido imitar a un for con una recursión… para eso podemos ocupar un for.

- ¿Es posible definir una función para calcular el factorial de un número de otra forma a la vista anteriormente?
- No tiene sentido imitar a un for con una recursión… para eso podemos ocupar un for.
- La definición formal de recursión dice:

> "En programación, un método usual de simplificación de un problema complejo es la división de este en subproblemas del mismo tipo. Esta técnica de programación se conoce como divide y vencerás y es el núcleo en el diseño de numerosos algoritmos de gran importancia."

Recursión

Ejemplo: Factorial $n!$

- **Caso base:**
  - ¿Dónde termina la recursión? n=1
  - ¿Qué devolvemos? 1
- **Caso recursivo:**
  - n*factorial(n-1)
- **Parámetros:**
  - n = 4
- **Reducción de un problema:** Dividimos el problema n*factorial(n-1) hasta que n=1.

Ejercicio

Mediante una función recursiva, calcule el n-ésimo término de la sucesión de Fibonacci.

Ejercicio

Mediante una función recursiva, calcule el n-ésimo término de la sucesión de Fibonacci.

- **Caso base:**
  - Primer término de la sucesión de Fibonacci = 0.
  - Segundo término de la sucesión de Fibonacci = 1.
- **Caso recursivo:**
  - n-ésimo término de la sucesión de Fibonacci = (n-1)-ésimo término de la sucesión de Fibonacci + (n-2)-ésimo término de la sucesión de Fibonacci
- **Parámetros:** Término de la sucesión.
- **Reducción de un problema:** ¿Qué sabemos? Los valores de los dos primeros términos.. En base a eso vamos sumando el resto de los términos.

Ejercicio

Mediante una función recursiva, calcule el n-ésimo término de la sucesión de Fibonacci.

```python
def fibonacci(n):
  if n == 0:
    return 0
  elif n == 1:
    return 1
  else:
    return fibonacci(n-1)+fibonacci(n-2)

print(fibonacci(7))
```

```
13
```

Ejercicio

Mediante el uso de una función recursiva, calcule la sumatoria $\sum_{1}^{n} k^2$

Recursión

Ejemplo: Sumatoria $\sum_1^n k^2$?

- **Caso base:**
  - k^2 = 1.
- **Caso recursivo:**
  - Sumo $n^2 + \sum_1^{n-1} k^2$
- **Parámetros:** n
- **Reducción de un problema:** ¿Cuál es el caso más simple? El caso base. Entonces vamos reduciendo el problema hasta que llegamos a algo que sabemos. Es decir, $\sum_1^1 k^2 = 1$

Ejercicio

Mediante el uso de una función recursiva, calcule la sumatoria $\sum_1^n k^2$

```
def sumatoria_k2(n):
    if n == 1:
        return 1
    else:
        return sumatoria_k2(n-1)+n**2

print(sumatoria_k2(6))
```

```
91
```

Resumen

- ¿Qué elementos podríamos agregar a la definición en base al ejemplo anterior?
  - **Caso base:**
    - Donde termina la recursión y devolvemos un valor específico.
  - **Caso recursivo:**
    - La llamada de la función a si misma.
  - **Parámetros:** que le entregamos a la función al llamarla por primera vez.
  - **Reducción de un problema:** donde el caso más simple es el caso base.

Bibliografía

- http://runest.ing.puc.cl/recursion.html

Links

- https://repl.it/@FelipeLopez/IIC1103Recursion que contiene todos los ejemplos de la clase.

Clase 6 – Recursión

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

12-09-2019