



Funciones Lambda (Python)

Algoritmos y Programación I

rev1.0

Paradigma funcional:

La programación funcional es un paradigma declarativo.

Nos importa hacer foco en "qué" estamos haciendo y no en "cómo" lo estamos haciendo a diferencia del enfoque imperativo.

Esto quiere decir que nosotros expresaremos nuestra lógica sin describir controles de flujo; no usaremos ciclos o condicionales.

Paradigma funcional:

Desarrollando software en este paradigma, trabajaremos principalmente con funciones, evitando los datos mutables, así como el hecho de compartir estados entre funciones.

Con este paradigma las funciones serán tratadas como **ciudadanos de primera clase (first-class citizen)**. Las funciones podrán:

- ser asignadas a variables
- utilizadas como parámetros de entrada
- utilizadas como retorno de funciones

Funciones anónimas:

Términos equivalentes dependiendo el lenguaje de programación:

- Anonymous functions
- **Lambda functions**
- Lambda expressions
- Lambda abstractions
- Lambda form
- Function literals

Sintáxis:

```
lambda arguments: expression
```

Esta función puede tener cualquier número de argumentos, pero sólo una expresión, que se evalúa y devuelve.

Debe tener en cuenta que las funciones lambda están restringidas sintácticamente a una única expresión.

Ejemplo:

función anónima

```
>>> (lambda x: x + 1)(2)
```

```
3
```

función anónima asignada a variable

```
>>> increment = lambda x: x + 1
```

```
>>> increment(2)
```

```
3
```

Ejemplo:

```
>>> add = lambda x,y: x + y
```

```
>>> increment = lambda x: add(x,1)
```

```
>>> increment(2)
```

```
3
```


Ejemplo:

```
>>> def procesar(valor, operacion):  
    return operacion(valor)
```

```
>>> incremento = lambda x: x+1
```

```
>>> procesar(5, incremento)
```

```
6
```