TALENTOFUTURO

# Clase: Programación Funcional

Módulo 2: Programación Avanzada en Python.

Apoyado por:
CORFO

# Clase de hoy

01

## Funciones anónimas

Lambdas en Python y formas de uso.

02

## Operaciones anónimas

Uso de funciones lambda sobre listas (map, filter, reduce).

# 3A Funciones anónimas

## Bloque A

# Qué veremos en Bloque A

- Definición del concepto “función anónima”.
- Funciones lambda en Python.
- Funciones lambda con condicionales.

# Funciones anónimas

## ¿Qué es una función anónima?

- Hasta ahora hemos revisado el concepto de función.
- Hemos asumido hasta ahora que las funciones deben tener un nombre.
- Además, hemos discutido la necesidad de que este nombre sea significativo para la operación que expresa la función.
- En Python es posible tener funciones sin nombre: las llamaremos en forma genérica como funciones anónimas.
- Más específicamente, en Python estas funciones se implementan usando el concepto de “lambda”.

# Funciones anónimas

## ¿Por qué usar una función anónima?

- Si tenemos funciones cuyo nombre ayuda a comprender las operaciones realizadas en el programa, ¿qué sentido tiene usar funciones anónimas?

- La principal motivación para el uso de funciones anónimas es la aplicación de operaciones típicamente genéricas a un conjunto de valores obviando la obligación de definir una función formalmente.

- Por ejemplo, si nos pasan una lista de estudiantes con notas y nos piden transformar la nota en “aprobado” versus “reprobado”, en vez de definir una función con nombre (y que probablemente se usará solo una vez), podemos definir una operación que se aplique a los datos en forma anónima.

# Funciones anónimas

## Lambdas en Python

- Introduciremos el concepto de “lambda” en Python.
- Las operaciones de tipo “lambda” nos permiten implementar funciones anónimas en la forma:

```
lambda parámetro: operación
```

- Ejemplo:

```
lambda x: x**2  # elevar al cuadrado
```

- Las funciones lambda SIEMPRE deben retornar algo.

# Funciones anónimas

## Lambdas en Python

- Hay dos formas de aplicar una función lambda: usando un esquema definición-atributos y llamando a una lambda almacenada en una variable.
- Esquema definición-atributos:

```
(lambda x: x+2)(2) # retorna 4
```

- Lambda almacenada en una variable:

```
cuadrado = lambda x: x+2
cuadrado(2) # retorna 4
```

# Funciones anónimas

## Lambdas en Python

- Podemos generalizar una lambda para varios parámetros:

```
lambda x, y: x+y # retorna la suma de x e y
```

- Ejemplo:

```
potencia = lambda x, y: x**y
potencia(2,4) # retorna 16
```

# Funciones anónimas

## Lambdas con decisiones

- Podemos incluir decisiones (agregando una condición) en la lambda.
- Pero debemos recordar que una lambda SIEMPRE retorna algún valor por lo que la rama “else” siempre tiene que ser agregada.

```
lambda x, y: “x es mayor” if (x>y) else “x es menor”
```

- Si por alguna razón no queremos retornar un valor, la instrucción “else” debe ir, pero podemos retornar el valor dummy `None`.

```
lambda x, y: “x es mayor” if (x>y) else None
```

# Trabajo grupal – Bloque A

# Paralelo 1

| G1 | G2 | G3 | G4 |
|---|---|---|---|
| Mónica Carvajal Romero | Rodrigo Yanez Pilgrim | Ivan Santivañez | John Gonzalez |
| Patricio Hernández Pavez | Pablo Mahuzier | Francisco Zuloaga | Rodrigo Espinoza |
| Paula Rivera Oberg | Jhonatan Ortiz | Hamid Pinilla Saah | Pablo Silva |
| Valentina Loyola Rodó | Ariel Mora | Freddy Vega | Marlenne Allendes |
| Nicolas Gonzalez | María Ignacia Strobel | Hilda Silva | Jorge San Martin |
| Nicolás Aravena | christian donoso | Teresa Peña | Carlos Gonzalez |

| G5 | G6 | G7 | G8 |
|---|---|---|---|
| Paulina Valenzuela | Gabriela Torres | Miguel Mayorga | Felipe Mercado Lopez |
| Daniela Castro | Matias Araya | Isaac Vega | Francisco Yañez |
| Jorge Moraga Calvo | Alex Riquelme | Francisco Villaseca | Matias Faundes |
| Alejandro Gonzalez | Jorge Troncoso | Matías Macaya | christopher nuñez soto |
| Francisco Vega Contreras | Valentina Gutierrez | Gabriela Karina Lopez | Angelo Farias |
| Alvaro Torres | Jorge Estefane | Rodrigo Fuenzalida Vera | yuri urzua lebuy |

# Trabajo grupal – Ejercicio #1

## Valor absoluto

- Escriba una función lambda que reciba un número y retorne su valor absoluto.
- Recuerde que el valor absoluto realiza lo siguiente: si el número es positivo o cero, devuelve el mismo número; si el número es negativo, devuelve el mismo número pero como valor positivo.

# Trabajo grupal – Ejercicio #2

## Función distancia como “lambda”

- Escriba nuevamente la función de distancia euclidiana.
- Transforme la función a una función anónima (lambda).
- Su programa debe recibir las coordenadas por la entrada estándar y luego llamar a la función anónima.
- La función anónima debe retornar la distancia euclidiana (presente el resultado por la salida estándar).

# Trabajo grupal – Ejercicio #3

## Dígito verificador en un RUT

- Escriba una función lambda que retorne el dígito verificador de un RUT.
- La lambda debe recibir como parámetro un RUT sin dígito verificador y retornar este último.
- Revise el algoritmo para dígito verificador en: https://es.wikipedia.org/wiki/Rol_%C3%9Anico_Tributario (sección “Procedimiento para obtener el dígito verificador”.

# Break!

# 3B Operaciones anónimas sobre listas

## Bloque B

# Qué veremos en Bloque B

- Operaciones de tipo map().
- Operaciones de tipo filter().
- Operaciones de tipo reduce()

# Funciones lambda

## Operaciones de tipo map()

- Las operaciones de tipo map() aplican una función a todos los elementos de una entrada.
- Estas operaciones permiten aplicar una lambda a una lista para luego crear una nueva lista con la lambda aplicada.
- La forma general de map() que usaremos es:

```
lista_nueva = list(map(lambda x: …, lista_original))
```

# Funciones lambda

## Operaciones de tipo map()

- El equivalente al uso de una operación de tipo map() es construir un ciclo que toma cada valor de una lista y construye una lista nueva con valores a los que se les ha aplicado alguna función.

- Es importante destacar que operación map() aplica uno a uno, pasando por todos los elementos, pero no realiza operaciones de reducción o similares.

- Ejemplo:

```
raiz_cuadrada = list(map(lambda x: x**0.5, lista_original))
```

# Funciones lambda

## Operaciones de tipo filter()

- Una operación de tipo filter() permite aplicar una operación lambda sobre elementos de una lista.
- La operación filter() permite construir una lista con todos los elementos que cumplan con la condición definida por la lambda.
- La sintaxis general es la siguiente:

```
lista_nueva = list(filter(lambda x: condición, lista))
```

# Funciones lambda

## Operaciones de tipo filter()

- La operación filter() es equivalente al uso de un ciclo en el cual se va testeando la condición desde una lista original y se crea una lista con los elementos

- Ejemplo:

```
lista_filtrada = list(filter(lambda x: x >= 10, lista_original))
```

# Funciones lambda

## Operaciones de tipo reduce()

- Una operación de tipo reduce() requiere del llamado a una biblioteca específica:

```
from functools import reduce
```

- Las operaciones tipo reduce() permiten la aplicación de una lambda sobre una lista y obtener el resultado de esta operación aplicada a todos los valores.

- La sintaxis general de reduce es la siguiente:

```
resultado = reduce((lambda), […,…,…])
```

# Funciones lambda

## Operaciones de tipo reduce()

- En específico, una lambda para reduce() debe permitir dos parámetros y devolver una operación que involucre al menos uno de los elementos de la lista.

- Ejemplo:

```
suma = reduce((lambda x, y: x + y), [1, 2, 3, 4]) # suma = 10
```

- El procedimiento es el siguiente:
  - Primero, a x e y se les asigna los primeros dos valores.
  - Luego, x mantiene la suma anterior y a “y” se le asigna el siguiente valor.
  - Entonces se obtiene la suma entre x (que tiene el valor anterior) e y que tiene el valor siguiente en la lista.
  - Esto se repite hasta llegar al final de la lista.

# Trabajo grupal – Bloque B

# Paralelo 1

| G1 | G2 | G3 | G4 |
|---|---|---|---|
| Mónica Carvajal Romero | Rodrigo Yanez Pilgrim | Ivan Santivañez | John Gonzalez |
| Patricio Hernández Pavez | Pablo Mahuzier | Francisco Zuloaga | Rodrigo Espinoza |
| Paula Rivera Oberg | Jhonatan Ortiz | Hamid Pinilla Saah | Pablo Silva |
| Valentina Loyola Rodó | Ariel Mora | Freddy Vega | Marlenne Allendes |
| Nicolas Gonzalez | María Ignacia Strobel | Hilda Silva | Jorge San Martin |
| Nicolás Aravena | christian donoso | Teresa Peña | Carlos Gonzalez |

| G5 | G6 | G7 | G8 |
|---|---|---|---|
| Paulina Valenzuela | Gabriela Torres | Miguel Mayorga | Felipe Mercado Lopez |
| Daniela Castro | Matias Araya | Isaac Vega | Francisco Yañez |
| Jorge Moraga Calvo | Alex Riquelme | Francisco Villaseca | Matias Faundes |
| Alejandro Gonzalez | Jorge Troncoso | Matías Macaya | christopher nuñez soto |
| Francisco Vega Contreras | Valentina Gutierrez | Gabriela Karina Lopez | Angelo Farias |
| Alvaro Torres | Jorge Estefane | Rodrigo Fuenzalida Vera | yuri urzua lebuy |

# Trabajo grupal – Ejercicio #1

## Determinar el mayor valor en una lista

- Escriba un programa que reciba una lista y permita determinar el valor mayor.
- Debe usar una operación tipo “reduce()” para realizar las comparaciones.
- El programa debe imprimir por pantalla el valor mayor de la lista.

# Trabajo grupal – Ejercicio #2

## Viento registrado en estaciones de monitoreo

- Cree un programa que solicite el nombre de una estación de monitoreo y los vientos registrados (nudos) en las últimas 5, 10, y 15 horas.
- Almacene esta información en la memoria principal usando diccionarios y listas.
- Su programa debe crear un nuevo diccionario con los vientos registrados en kilómetros por hora.
- Además, el programa debe mostrar por la salida estándar el nombre de la estación y los vientos registrados (convertidos a kilómetros por hora).
- Debe usar operación map().
- Tip: los vientos en una zona calmada están entre los 3 y 10 nudos.

# Trabajo grupal – Ejercicio #3

## Viento registrado en estaciones de monitoreo

- Modifique el programa anterior para mostrar por pantalla los vientos que sobrepasan los 20 kilómetros por hora.

- Utilice una operación filter() para complementar el ejercicio anterior de tal manera que pueda dar cumplimiento a los requisitos de este ejercicio.

TALENTOFUTURO

# ¿Preguntas?

¡Hemos llegado al final de la clase!