PYTHON APLICADO A LA CLIMATOLOGÍA

Bch. Javier Chiong Ravina

¿Qué es python?

Python es un lenguaje de programación interpretado, multiparadigma y multiplataforma que busca desarrollar un código legible.

Interpretado:

- El código se va traduciendo a lenguaje máquina a medida que es necesario.

Multiparadigma:

- Programación orientada a objetos
- Programación imperativa
- Programación funcional

Multiplataforma:

- IOS, Windows, Linux

Programación orientada a objetos

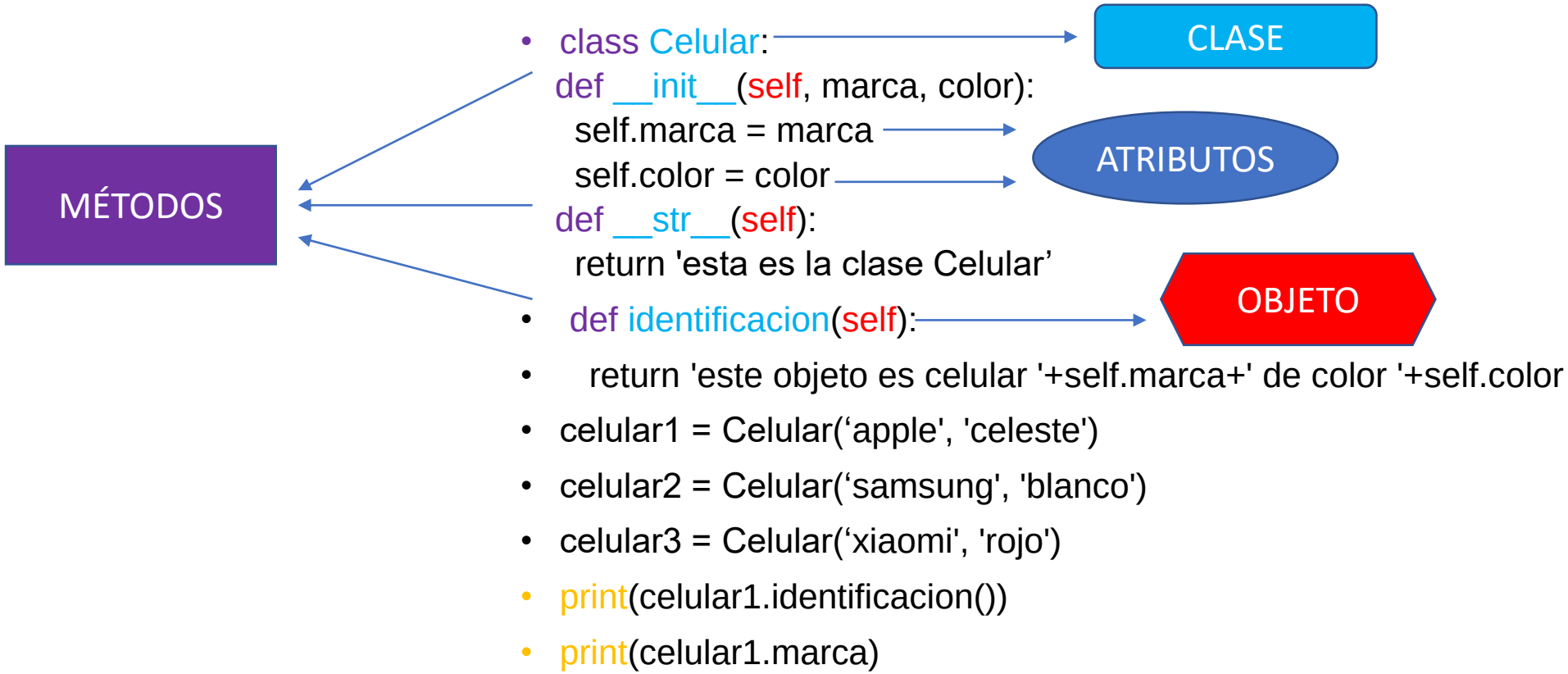
- Celular → CLASE

- Celular Xiaomi Redmi,
- Celular Samsung Galaxy,
- Celular Iphone 12 → OBJETOS

- Marca, Color, Tamaño,
- Peso →ATRIBUTOS

- Llamar, Tomar Fotos,
- Enviar correos → MÉTODOS

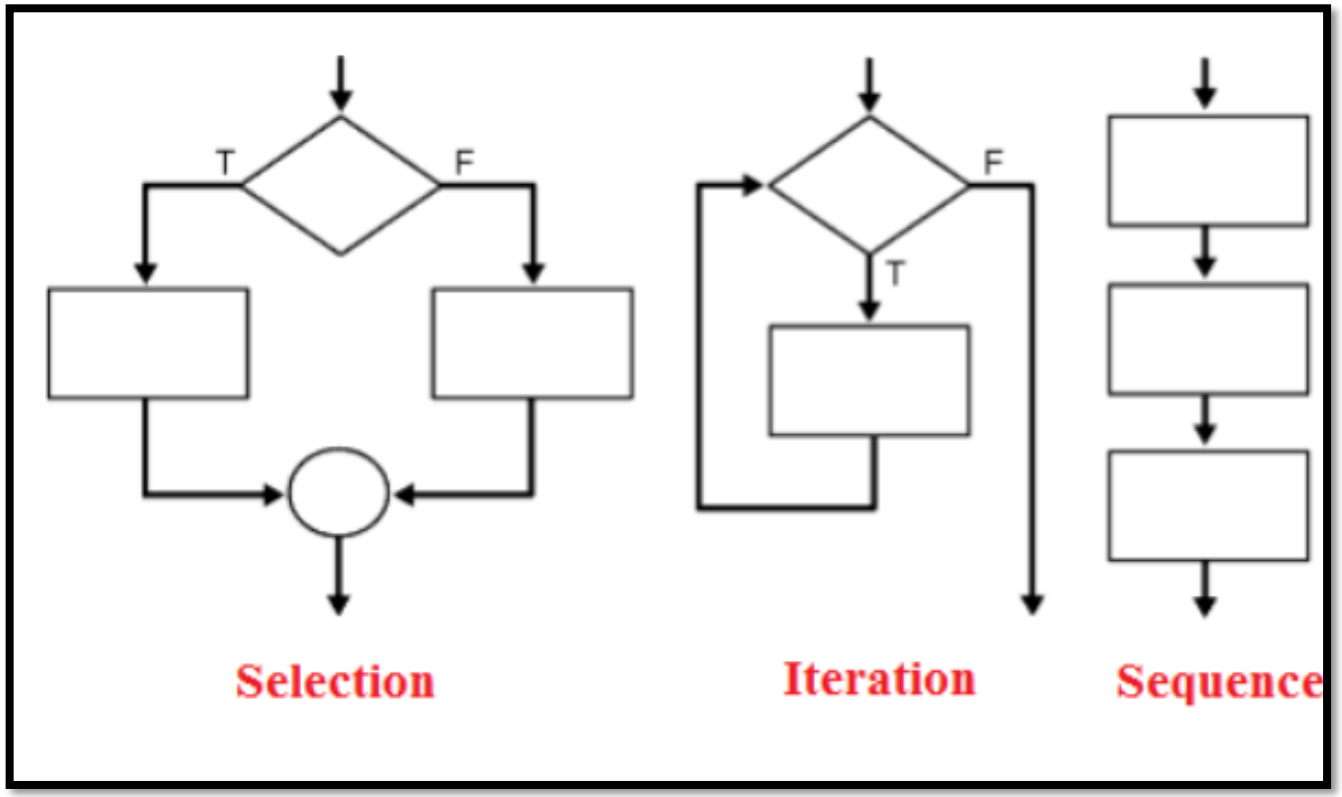
Programación orientada a objetos



Programación orientada a objetos

```
class calculadora:
    def __init__(self,n1,n2):
        self.suma = n1+n2
        self.resta = n1-n2
        self.producto = n1*n2
        self.division= n1/n2

operaciones= calculadora(2,4)
div=operaciones.division
```

Estructuras de control



- Una estructura de control (o flujo de control) es un bloque de programación que analiza variables y elige una dirección en la que ir en función de parámetros dados.

Estructura de control

Tipos de estructuras de control:

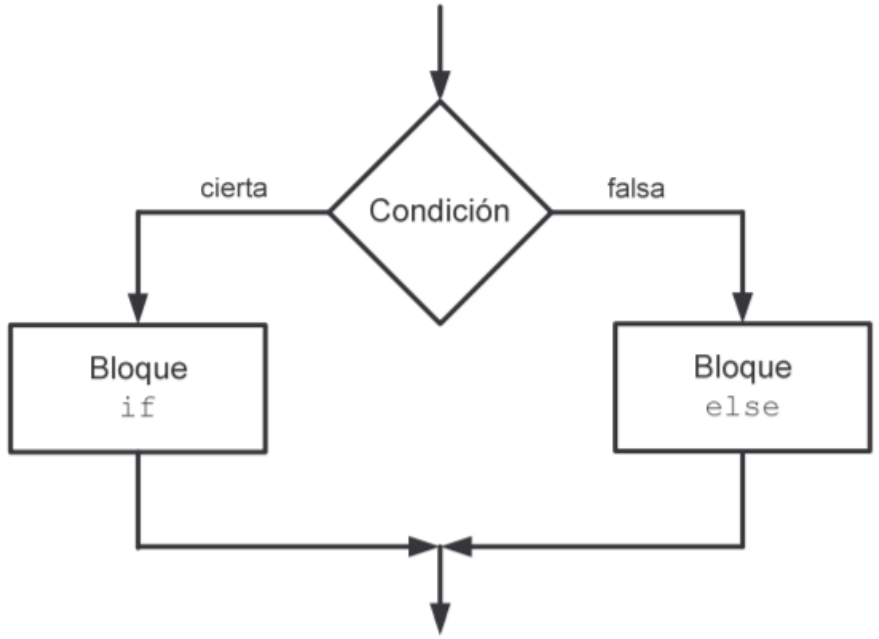
- **Estructuras de control de selección condicionales**: nos permiten comprobar condiciones y hacer que nuestro programa se comporte de una forma u otra, que ejecute un fragmento de código u otro, dependiendo de esta condición. **Ejemplo**: **if**, **elif**, **else**.

- **Estructuras de control iterativas**: nos permiten ejecutar un mismo código, de manera repetida. **Ejemplo**: bucle **while** y bucle **for**.

Estructuras de control de selección condicionales

- Condición if, elif y else: Se utiliza para realizar acciones si estan cumplen con determinadas condiciones. Las condiciones son evaluadas a través de operadores de comparación.

Operador	Significado
==	Igual
!=	Distinto
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual



Estructuras de control de selección condicionales

La condición if, elif y else presenta las siguiente estructura:

MULTIPLES INSTRUCCIONES

```
if condición 1:
   instrucción 1
elif condición 2:
   instrucción 2
elif condición n:
   instrucción n
else:
   instrucción alternativa
```

DOBLE INSTRUCCIÓN

```
if condición 1:
   instrucción 1
else:
   instrucción alternativa
```

Al igual que en las clases, las condicionales terminan en 2 puntos :
En Python no hay necesidad de cerrar las estructuras de control condicionales con endif.

Estructuras de control de selección condicionales

Ejm: Calcular el mayor de 2 números

```
numero1=int(input("Número 1: "))
numero2=int(input("Número 2: "))
if numero1>numero2:
   print("El numero " +str(numero1)+" es mayor al "+str(numero2))
elif numero1<numero2:
   print("El numero " +str(numero2)+" es mayor al " + str(numero1))
else:
   print("El numero " +str(numero2)+" es igual al " +str(numero1))
```

Estructuras de control de selección condicionales

Ejm: Determinar el método de pago.

```
if compra <= 100:
   print "Pago en efectivo"
elif compra > 100 and compra < 300:
   print "Pago con tarjeta de débito"
else:
   print "Pago con tarjeta de crédito"
```

Ejm: Elegir un candidato

```
candidato=input("Candidato elegido: ")
if candidato.upper()=="A":
    print("Usted ha votado por el partido rojo")
elif candidato.upper()=="B":
    print("Usted ha votado por el partido verde")
elif candidato.upper()=="C":
    print("Usted ha votado por el partido azul")
else:
    print("Opción errónea")
```

Estructura de control - iterativa

- **Bucle while**: se encarga de ejecutar una misma acción "mientras que" una determinada condición se cumpla.
- **Bucle for**: se encarga de iterar sobre una variable que puede ser un string, una lista, una tupla o un array.

Estructura del **while**

```
while condición:
    instrucción
```

Estructura del **for**

```
for elemento in variable:
    instrucción
```

Ejercicios con bucle While

Escriba un algoritmo que calcule e imprima la suma de los n primeros números enteros positivos.

```
suma = 0
numero = int(input(u"Ingrese un número"))
while numero>0:
  suma = (numero*(numero+1))/2
  break
print ("Número ingresado:",numero)
print (u"Suma total:",suma)
```

Escriba un algoritmo que pueda encontrar un número en específico.

```
valores = [3, 4, 10, 2, 6, 5]
encontrado = False
indice = 0
longitud = len(valores)
while not encontrado and indice < longitud:
valor = valores[indice]
  if valor == 2:
    encontrado = True
  else:
    indice += 1
if encontrado:
  print(f'El número 2 ha sido encontrado en el índice {indice}')
else:
  print('El número 2 no se encuentra en la lista de valores')
```

Ejercicios con bucle For

Escriba un código para separar números positivos de negativos

```
x = [23,4,-6,23,-9,21,3,-45,-8]
pos = []
neg = []
for i in range(len(x)):
    if x[i] < 0:
        neg.append(x[i])
    else:
        pos.append(x[i])

print("Los numeros positivos son: ",pos)
print("Los numeros negativos son: ",neg)
```

Comandos Zip y enumerate

```
lista_prof = [“Astronomo”,”Arquitecto”,”Meteorologo”]
lista_edades = [32,29,28]
for nombre, edad in zip(lista_prof,lista_edades):
    print(nombre,edad)
```

```
lista_nombres = [‘Patrick’,’Gustavo’,’Henry’,’Kevin’]

for i, nombre in enumerate(lista_nombres):
```

Funciones

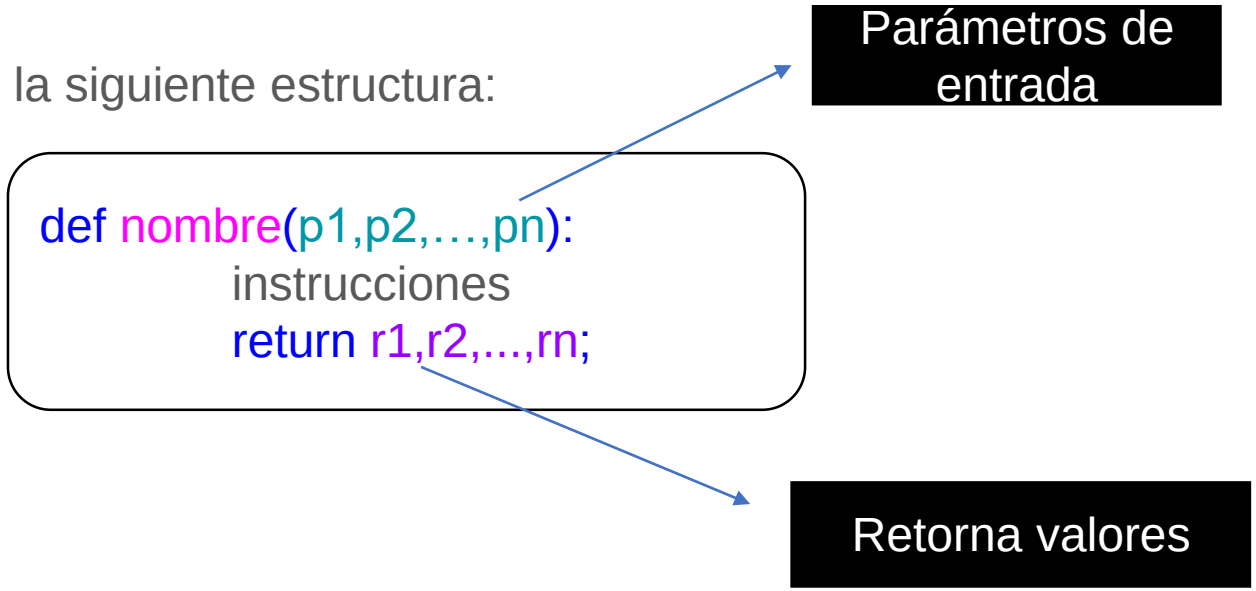
Una función es una serie de instrucciones para desarrollar una tarea en específica.

Algunas de las funciones pre-definidas de python son:

- help() : muestra la descripción de una función.
- print() : imprime un mensaje en la terminal
- type() : indica el tipo de un parámetro
- str() : convierte un parámetro en tipo string
- int() : convierte un parámetro en tipo entero
- float() : convierte un parámetro en tipo flotante (real con decimales)
- input() : lee un string ingresado por terminal

Funciones

- Las funciones en Python tienen la siguiente estructura:



- Donde:
- r1,r2,...,rn son los valores retornados

Funciones

PARÁMETROS SIN DEFINIR

```
def area_triangulo():
    a=5
    b=4
    area=b*a/2
    return area
area_triangulo()
```

PARÁMETROS DEFINIDOS

```
def area_triangulo(b,a):
    area=b*a/2
    return area

area_triangulo(5,6)
```

Funciones

Ejercicios:

- Realizar una función en donde se identifique si un número es primo
- Realizar una función en donde se calcule la distancia entre 2 puntos

```
def primo(num):
    for i in range(2,num):
        if num%i==0:
            return False
    return True

numero=int(input("Número: "))
if primo(numero):
    print("Es primo")
else:
    print("No es primo")
```

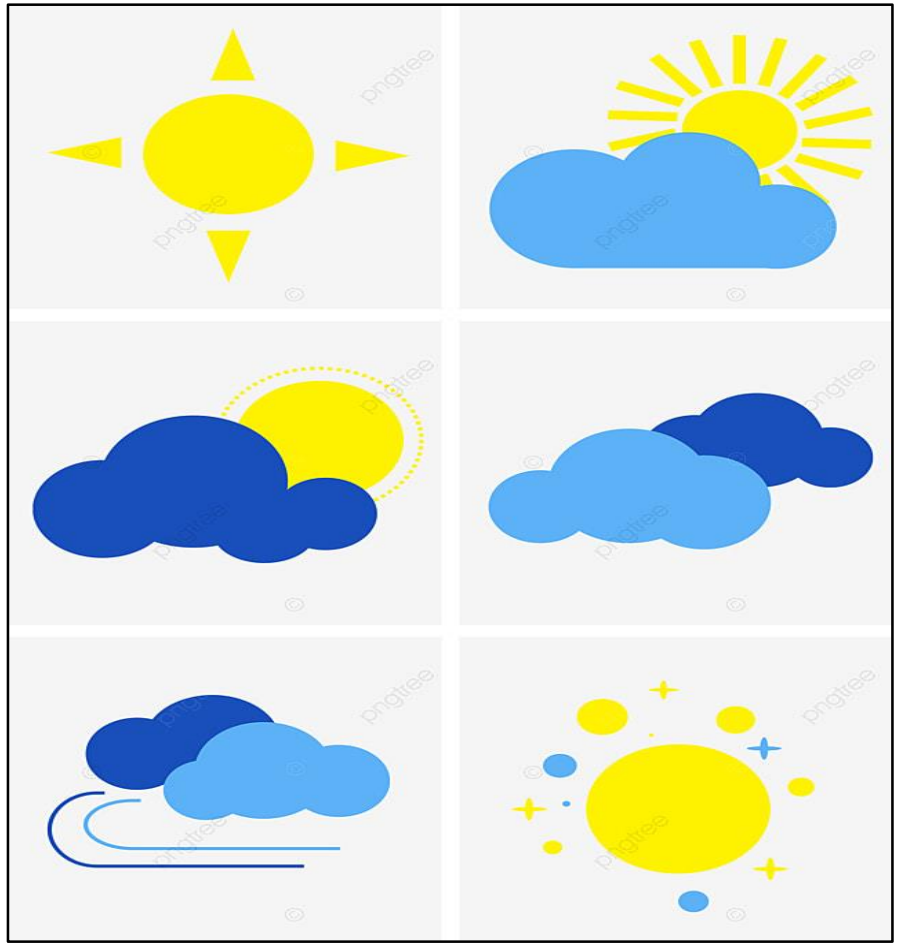
```
def distancia_entre_2_puntos(x1,x2,y1,y2):

    d=((x2-x1)^2-(y2-y1)^2)**(1/float(2))

    return d
```

Funciones

Ejercicios:

- Realizar una función en donde se describa subjetivamente el estado del tiempo.

```
def tiempo(t,hr,p):
    if t >= 20.0:
        if hr>85.0:
            print('El tiempo esta cálido')
        else:
            print('El tiempo esta templado')
    elif t < 20.0:
        print('El tiempo esta templado')
    if p < 1005.0:
        print('Probablemente llovera')
    else:
        print('Probablemente no llovera')
tiempo(22,90,995)
```



Función Lambda

- Las expresiones lambda (a veces denominadas formas lambda) son usadas para crear funciones anónimas. La expresión lambda parameters: expression produce un objeto de función.

Sintaxis

```
lambda p1, p2: expresión
```

Función Lambda

Ejecute una función lambda para extraer el año, mes y dia

```
import datetime
now = datetime.datetime.now()
print(now)
year = lambda x: x.year
month = lambda x: x.month
day = lambda x: x.day
t = lambda x: x.time()
print(year(now))
print(month(now))
print(day(now))
print(t(now))
```

Calcule la suma de 2 números

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

```
suma_dos = lambda
x,y : x + y
```