Introducción a Python

Informática I - 2547100

Departamento de Ingeniería Electrónica
y de Telecomunicaciones

Facultad de Ingeniería

2016-2

Python

Lenguaje de programación...

- de alto nivel
- de propósito general
- de fácil lectura
- interpretado

Creado por el holandés **Guido van Rossum** en 1989 y popular desde su versión 2.0 lanzada en el 2000. Actualmente trabaja en Dropbox.

Python programs

Un programa Python está compuesto por una secuencia de **instrucciones** que son ejecutadas por el **intérprete** en una **terminal**.

```
print('Hola')
pers = 12
print('Hay', pers, 'personas')
```

Sintaxis: *reglas para un programa válido*

Semántica: *significado de un programa*

Data objects

En Python, a los datos se les llama **objetos** y pueden ser **escalares** o **no-escalares**.

Los objetos escalares pueden ser de tipo:

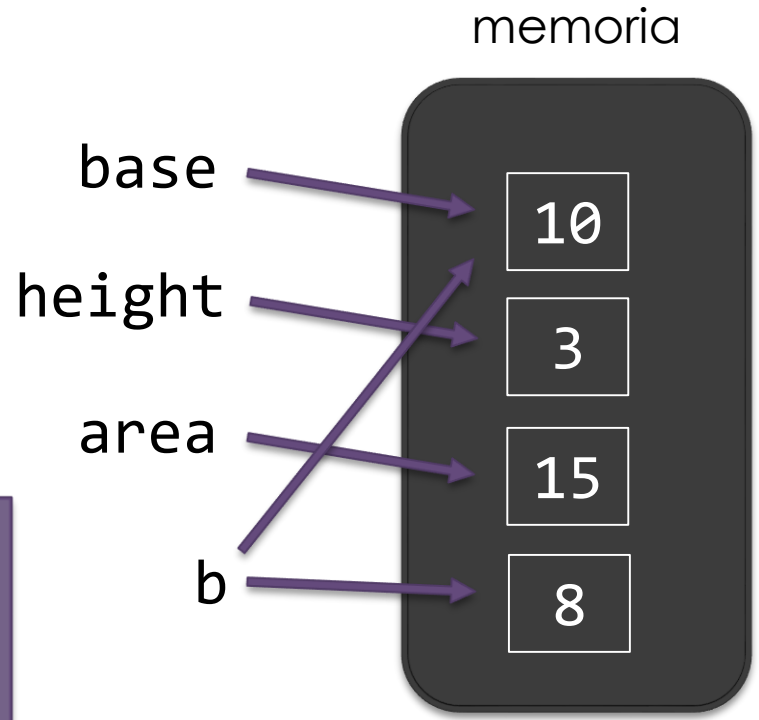
- `int`: números enteros (ej: 5)
- `float`: números reales (ej: 3.465)
- `bool`: True y False (verdadero y falso)
- `None`: vacío (ausencia de datos)

Variables

En Python, las variables son nombres que se pueden **ligar** a los objetos de datos.

```
base = 10
height = 3
area = base*height/2
b = base
b = 8
```

Los nombres de las variables son elegidos por el programador pero tienen algunas restricciones.



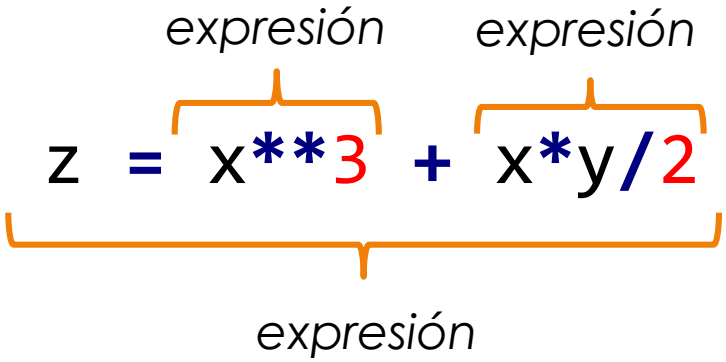
Operators

Los operadores son símbolos que se usan para ejecutar operaciones sobre los datos.

`a + b`	suma
`a - b`	resta
`a * b`	multiplicación
`a / b`	división
`a // b`	división entera
`a % b`	residuo de la division (módulo)
`a ** b`	potenciación
`a = 14`	asignación: ligar un nombre a un objeto de datos

Expressions

La **expresiones** son combinaciones de variables y operadores.



Strings

Un tipo de dato no-escalar muy utilizado es la **cadena de caracteres**, denotada por **`str`** en Python. Se usan las comillas (sencillas o dobles) para representar un `str` literal.

```
name = 'Juan Rodríguez'
country = 'Colombia'
d_quotes = "Comillas dobles también"
```

```
var1 = name + country
var2 = name * 2
var3 = name + 3
var4 = name * country
```

concatenar

replicar

error

Handling strings

```
name = 'Juan Rodríguez'
f = len(name)
```

longitud de un str

```
name[0]
name[f-1]
name[f]
name[-1]
```

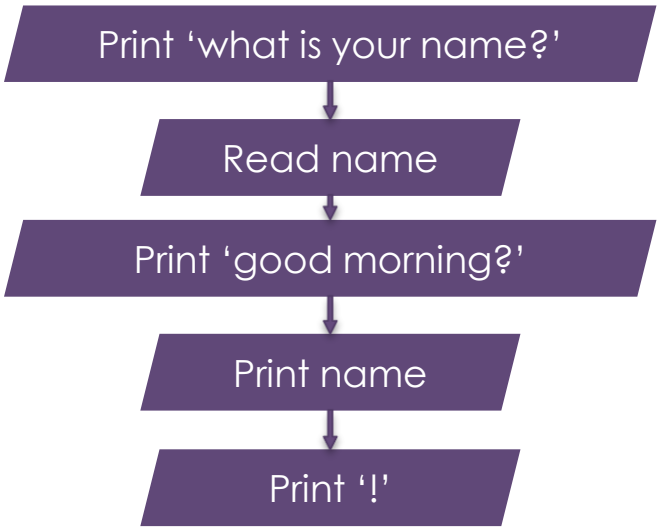
indexar un str

```
name[0:3]
name[0:f]
name[:]
name[:4]
name[7:]
```

partir (slice) un str

Input/Output

Para capturar datos que el usuario quiera ingresar al programa… o para mostrarle resultados:




```
name = input('What is your name?')
print('Good morning', name, '!')
print('Good morning' + name + '!')
```

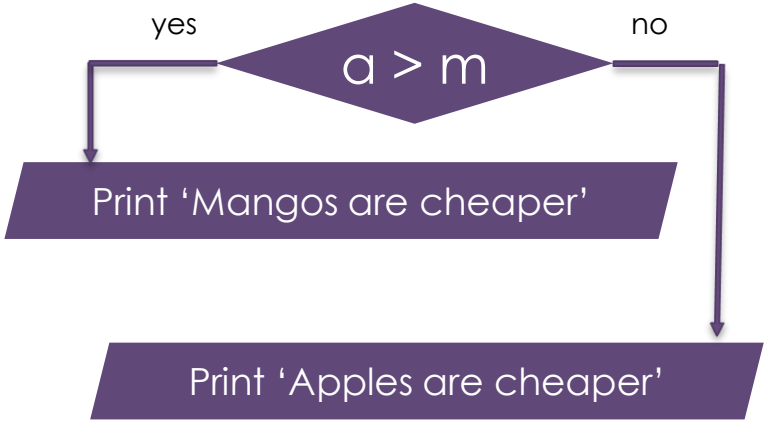
```
age = input('How old are you?')
print('In one year you will be', age+1)
print('In one year you will be', int(age)+1)
```

→ ***error***

Conditionals

Las instrucciones condicionales en Python se escriben así:

```
apples = int(input('Price of apples: '))
mangos = int(input('Price of mangos: '))
if apples > mangos:
    print('Mangos are cheaper')
else:
    print('Apples are cheaper')
```



*Programas de **tiempo constante**: sin importar los datos de entrada, hay un máximo número de instrucciones que el programa ejecuta*

Nested conditionals

Las instrucciones condicionales pueden anidarse:

```
if x%2 == 0:
    if x%3 == 0:
        print('Divisible by 2 and 3')
    else:
        print('Divisible by 2 and not by 3')
elif x%3 == 0:
    print('Divisible by 3 and not by 2')
```

Comparison and logical operators

`a == b`	igual
`a != b`	diferente
`a > b`	mayor
`a < b`	menor
`a >= b`	mayor o igual
`a <= b`	menor o igual
`a and b`	verdadero (True) si *ambas a y b* son verdaderas, de lo contrario, falso (False)
`a or b`	verdadero si *a o b* son verdaderas, de lo contrario, falso (False)
`not a`	verdadero (True) si a es falso y falso (False) si a es verdadero

Iteration

Los ciclos en Python se implementan mediante la instrucción **while** así:

```
N = int(input('Enter a number: '))
num = 1
sum = 0
while num <= N:
    sum = sum + num
    num = num + 1
print('The first', N, 'integers sum', sum)
```