



# Introducción a la Programación con Python

Profesor Juan Manuel Reyes  
Nivelatorio en Python - Sesión 1

# ¿Qué es Python?



**WHAT IS PYTHON?**

- ▶ A back end programming language
- ▶ High-level & approachable for beginners
- ▶ Has a welcoming & established community

**Used for tasks like:**

- Web Development
- Scripting
- Web Scraping
- Data Analysis
- Automation

**Used by companies like:**

- Google
- Pinterest
- Spotify
- Dropbox

**Used with frameworks like:**

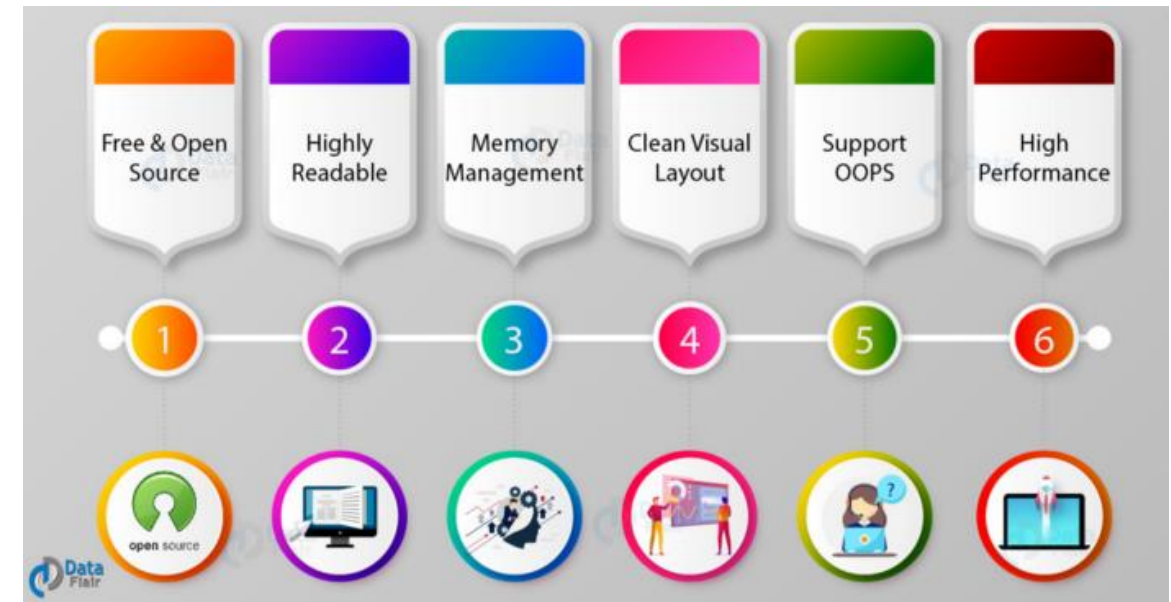
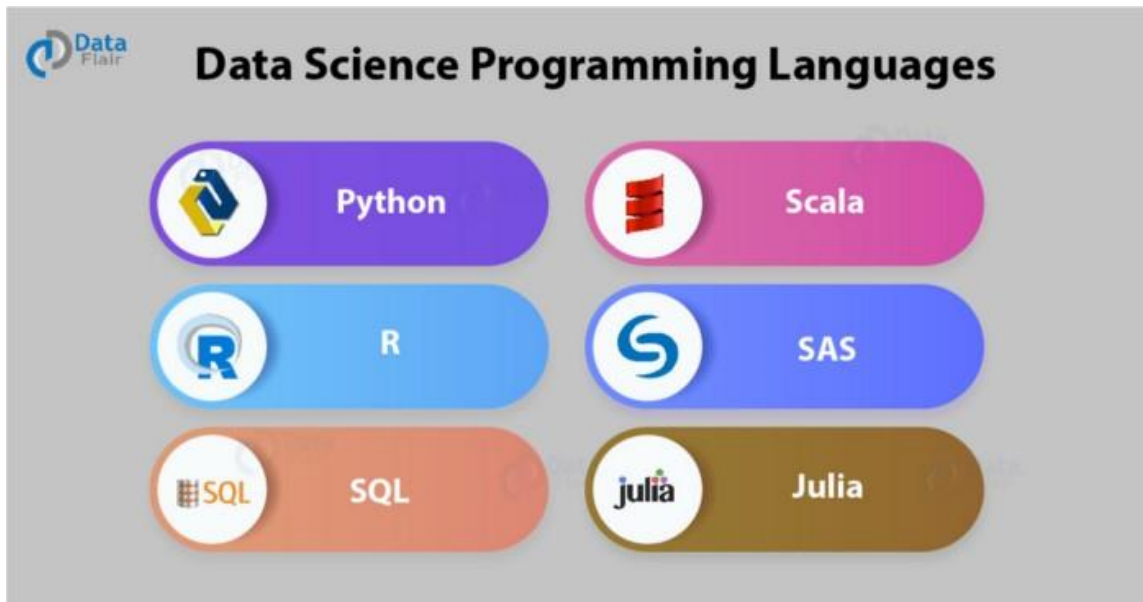
- django
- Flask
- jupyter

**COURSE REPORT** + **UNIVERSITY OF MINDS**

- Es un lenguaje de programación de propósito general, de alto nivel e interpretado.
- Fue creado por el alemán Guido van Rossum en 1991.
- Soporta múltiples paradigmas de programación: estructurado, OO y funcional.

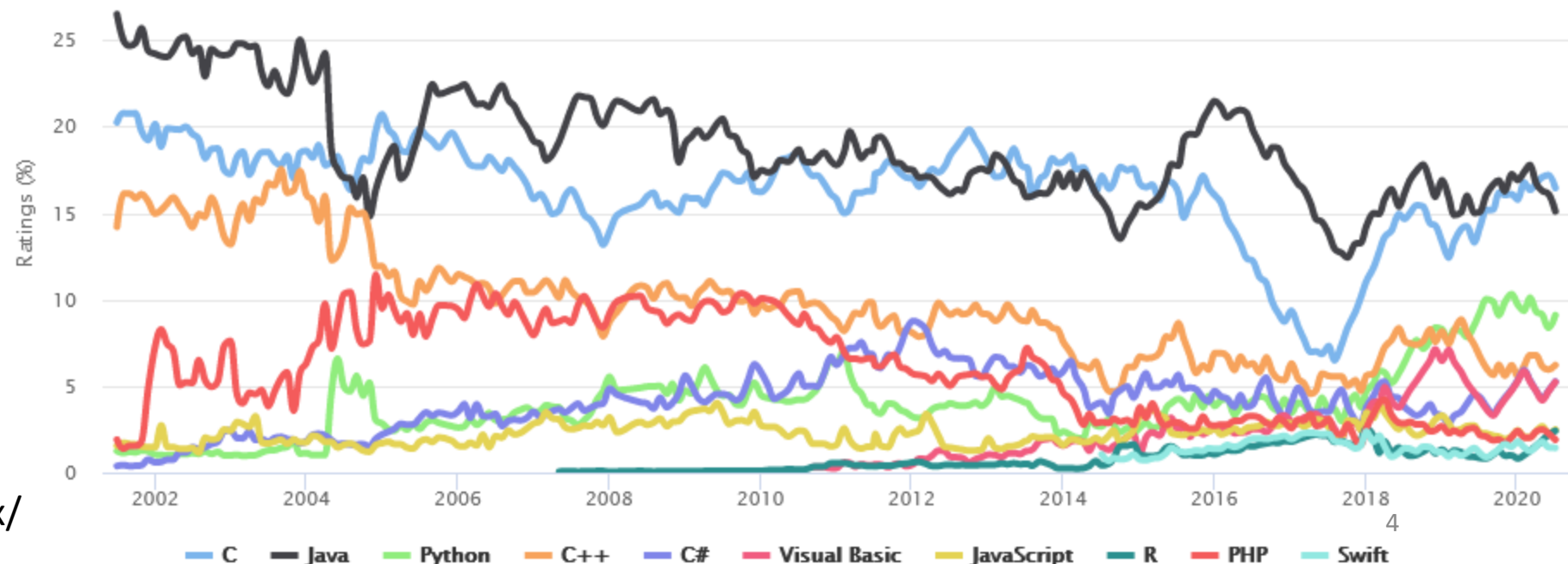


# ¿Por qué Python?



# ¿Por qué Python?

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.45%	+2.24%
2	1	▼	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%

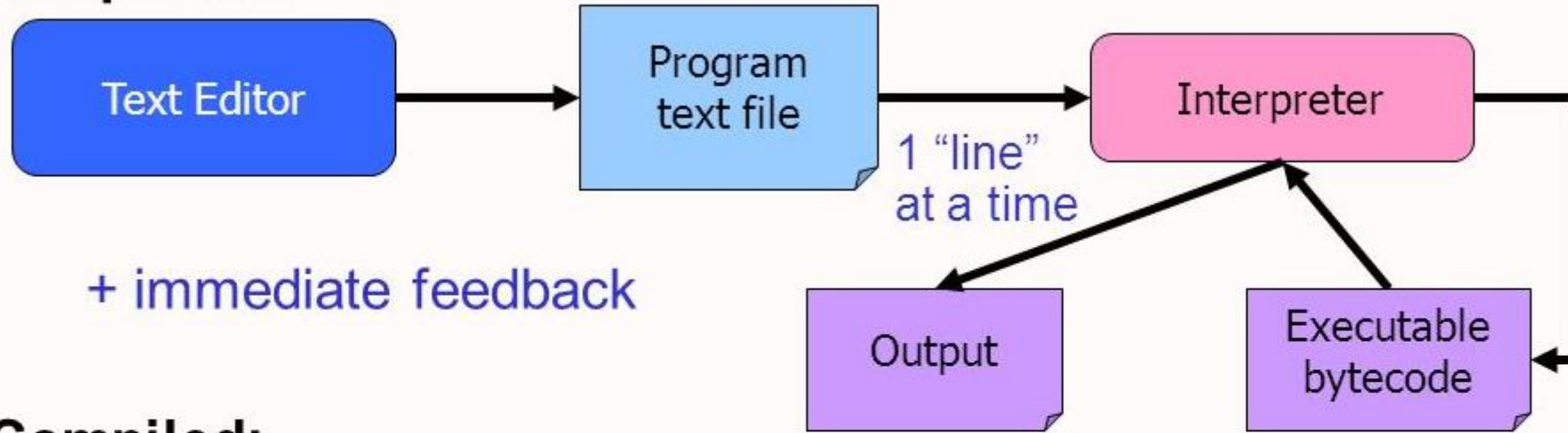


<https://www.tiobe.com/tiobe-index/>

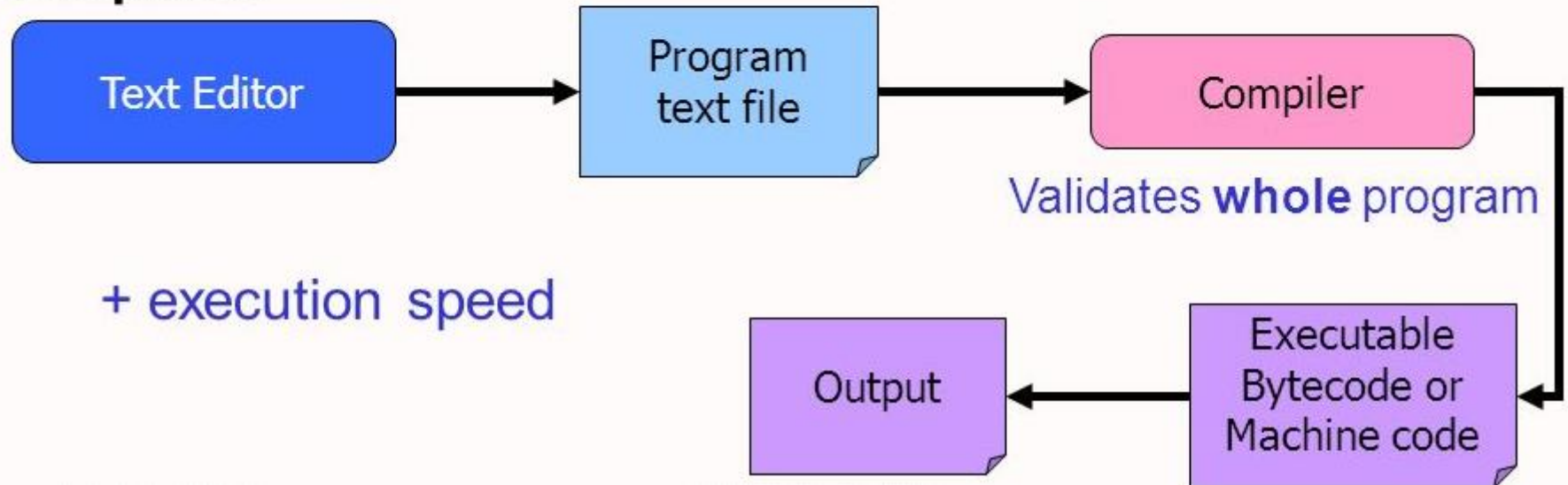


# Compiled vs. Interpreted Languages

## Interpreted:



## Compiled:

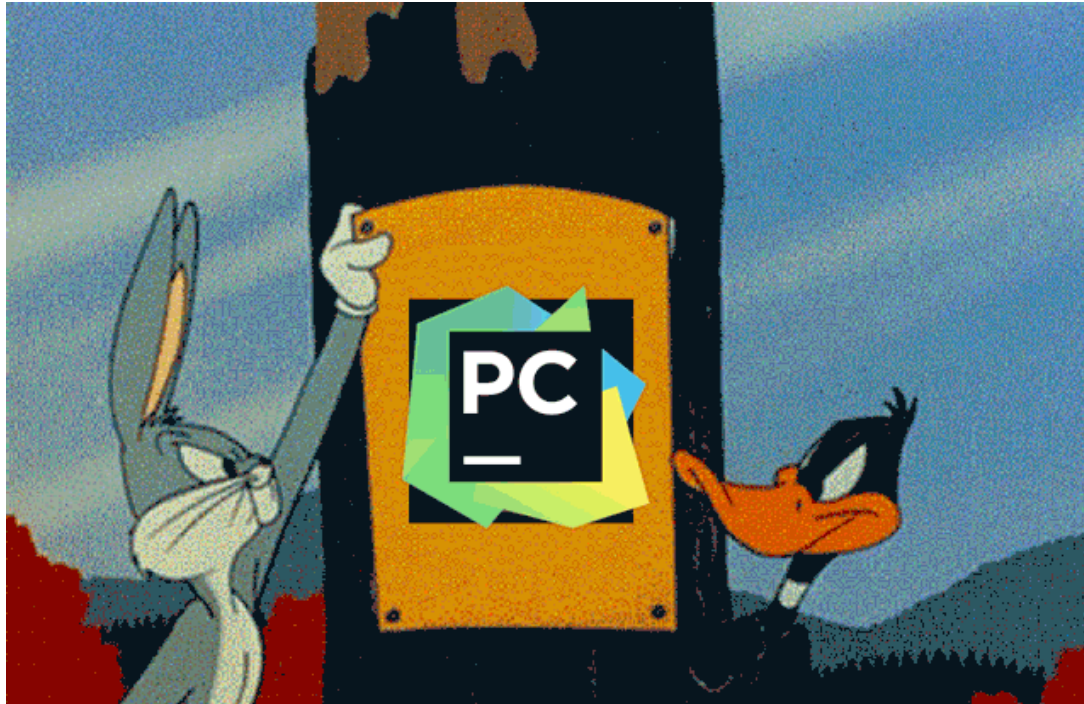


# ¿Qué es un IDE?

- IDE significa Entorno de Desarrollo Integrado (Integrated Development Environment)
- Es un editor de texto plano para programar que permite probar, compilar/interpretar/ejecutar.



# ¿Cuál IDE utilizar?



Utilizaremos Jupyter notebooks por:

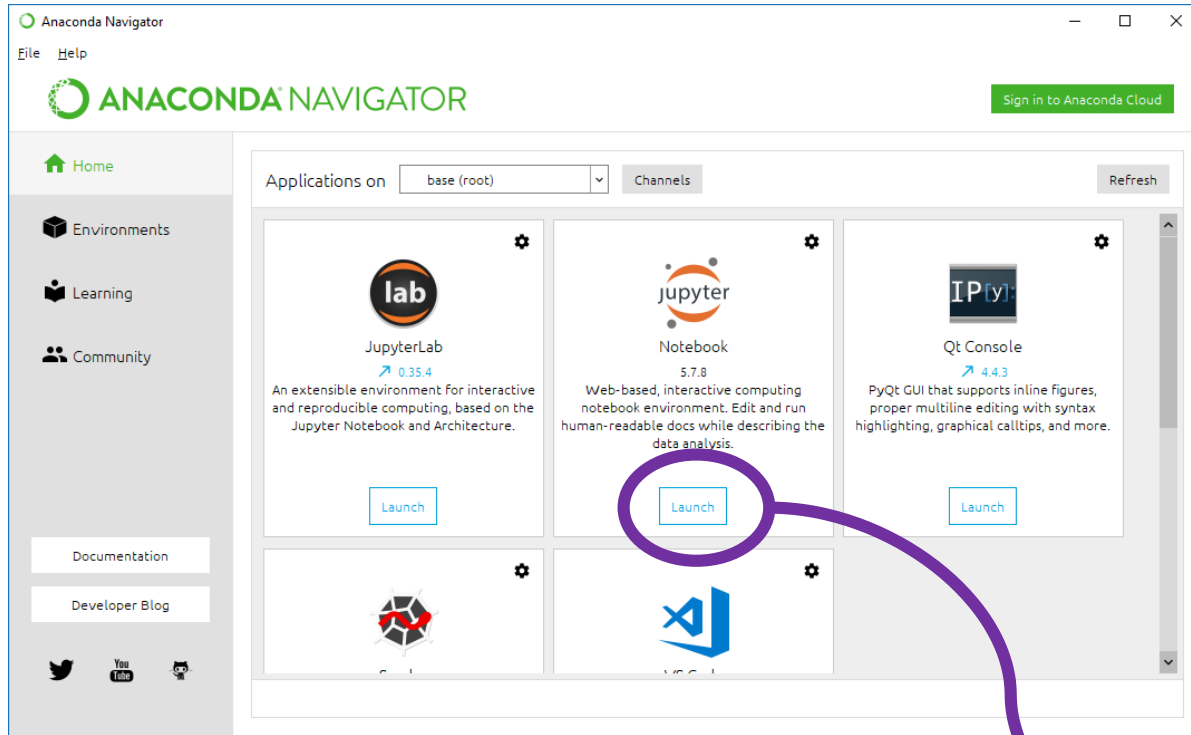
- La facilidad para la organización y presentación de scripts.
- Sólida integración con Anaconda que permite un ambiente de fácil instalación y configuración.
- Bien soportada por su amplio uso entre la comunidad de científicos de datos.

# Nuestras herramientas



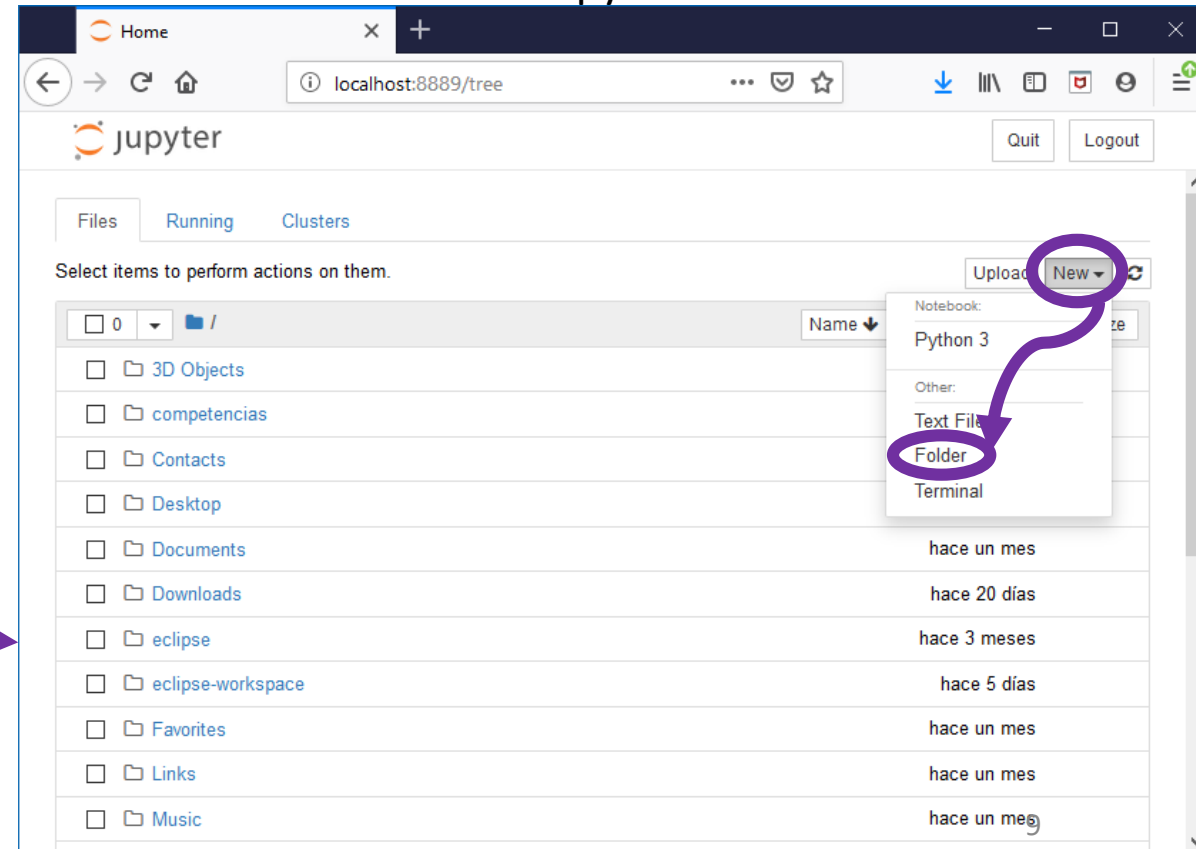


# Empecemos!

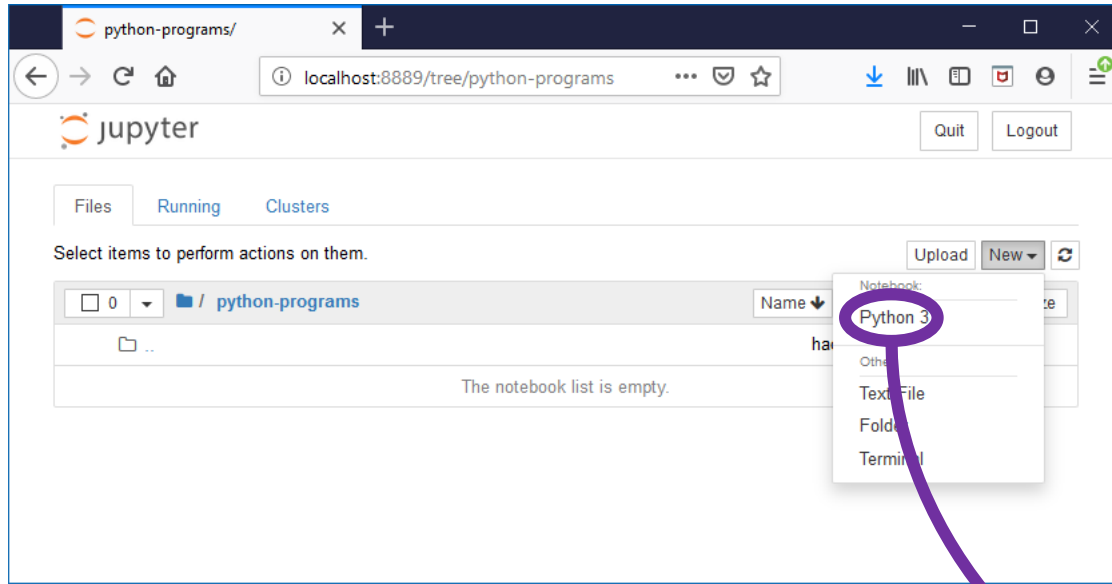


1. Iniciamos Anaconda Navigator

## 2. Lanzamos Jupyter Notebook

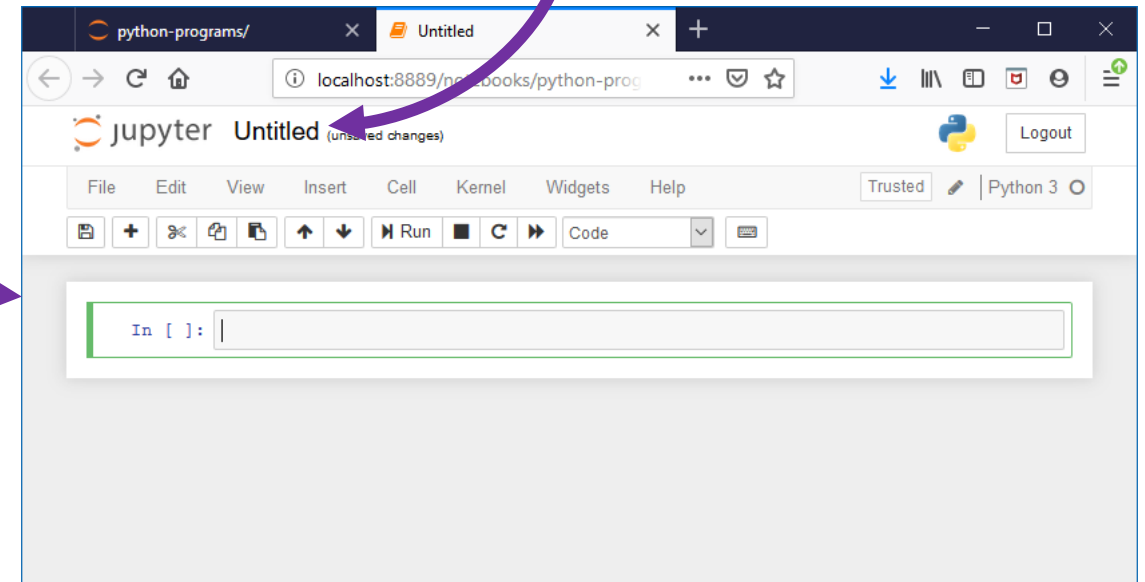


# Nuestro primer cuaderno de notas!



3. Creamos el primer cuaderno de notas

4. Le ponemos un nombre al cuaderno



# Hola Mundo!

La primera línea de código en Python.



```
print ("Hola Mundo")
```

# Tipos de Datos Básicos - Enteros

- En Python no hay límite para el tamaño de los números enteros.

```
print (12345678901234567890+1)
```

- Los valores enteros sin prefijo son considerados por defecto en base 10.

```
print (14)
print (32)
print (27)
```

- Los valores pueden estar en base: binaria, octal o hexadecimal.

```
print (0b10)
print (0o10)
print (0x10)
```

```
print (0b1011)
print (0o21)
print (0xAB)
type (123)
type (0b101)
```



# Tipos de Datos Básicos – Números de Punto Flotante

```
print (3.14)
type (3.1415)
```

```
print (7.)
type (28.)
```

```
print (.1)
type (.83)
```

```
print (.6e4)
print (271828182846e-11)
```

## Límites

```
print (1.79e308)
print (1.8e308)
print (-1.79e308)
print (-1.8e308)
```

```
print (5e-324)
print (2e-324)
```

# Tipos de Datos Básicos – Cadenas de texto

```
print ("Esto es una cadena")
type ("Programar es genial")

print ('esto también es una cadena')
type ('Python es lo mejor!')

print ('su nombre es: 'Ana')
print ("su nombre es: "Ana")

print ('su nombre es: "Ana"')
print ("su nombre es: 'Ana'")
```

## Caracteres de escape \

```
print ('su nombre es:\'Ana\')
print ("su nombre es:\"Ana\"")
print ("este es el caracter de escape \\")
```

```
print ("a
b
c")
```

```
print ("a\
b\
c")
```

# Operaciones Aritméticas Básicas

$2+3$

$76 - 7*3$

$(76-7)*3$

$19 / 3$

$19 // 3$

$20 \% 7$

$27 \% 4$

$6 ** 2$

$4 ** 3$

$2 ** 7$

$(2 + 3j) - (5 - 1j)$

$(7 - 4j) * (2 + 9j)$

$'si' + (2 * 'ri')$

# Variables

- Son espacios de memoria del computador (RAM) que almacenan valores y tienen la facilidad de ser referenciados a través de un nombre.

```
x = 5
print (x*x*x)
```

```
y = 7.5
z = y + x**2
print ("el resultado es:",z)
```

```
nombre = "Juan"
apellido = "Reyes"
print (f"{nombre} {apellido}")
```

```
type (x)
type (z)
```

```
print ("Los %s Magos"%apellido)
print ("%s %s"%(apellido,nombre))
```


```
goles = 0b100
print ("Colombia hizo %d goles"%goles)
```



# Tipos de conversión en formateo de cadenas

Conversion	Meaning
'd'	Signed integer decimal.
'i'	Signed integer decimal.
'o'	Signed octal value.
'u'	Obsolete type – it is identical to 'd'.
'x'	Signed hexadecimal (lowercase).
'X'	Signed hexadecimal (uppercase).
'e'	Floating point exponential format (lowercase).
'E'	Floating point exponential format (uppercase).
'f'	Floating point decimal format.
'F'	Floating point decimal format.
'g'	Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'G'	Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'c'	Single character (accepts integer or single character string).





# Funciones Integradas Lectura Simple de Datos Condicionales

---



# Funciones – Definición

- Una función es una porción de código que lleva a cabo tareas específicas.
- Puede tener parámetros para llevar los valores de entrada al algoritmo que se lleva a cabo dentro de la función.
- Pueden retornar valores que son utilizados desde la porción de código donde es invocada o llamada.

`str` convierte un valor a string

`print` imprime una cadena de texto

`type` indica el tipo de dato de un valor

#-

```
print ("Hola Mundo")
```

Los parámetros son los valores que se pasan entre los paréntesis

#-

```
num = 3.2
```

```
tipo = type(num)
```

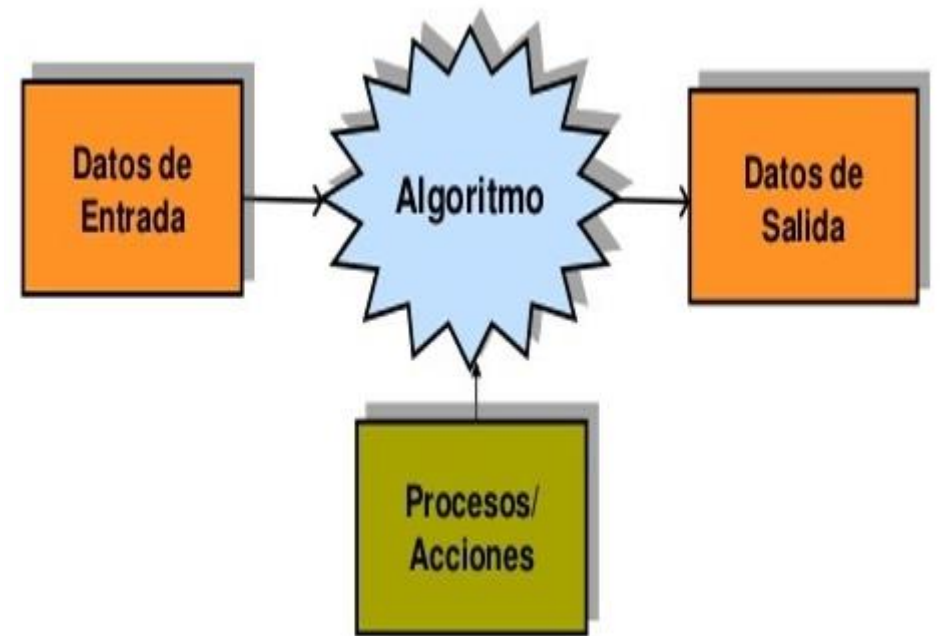
```
resultado = f"El tipo de {num} es {tipo}"
```

```
print(resultado)
```



# Entrada – Algoritmo – Salida

- Podemos decir que una función desempeña una subrutina, procedimiento o **algoritmo**.
- Los algoritmos generalmente reciben una **entradas** y entregan unas **salidas**.
- En las funciones, a esas **entradas** les llamamos **parámetros**, y a las **salidas** les llamamos valores de **retorno**.
- En Python, hay funciones sin parámetros y también funciones sin valores de retorno.





# Funciones – Invocación o Llamado

- Se dice que una función se llama o se invoca en el momento en que es utilizada para que lleve a cabo la tarea para la cual fue desarrollada.
- El primer ejemplo de función que estamos utilizando desde el inicio es **print**, cuyos parámetros son los valores a imprimir en pantalla.

```
cadena = str (8+3)
print("El resultado es "+cadena)
tipo = str(2+6.7)
```

¿La función **print** tiene valor de retorno?

# Funciones Integradas – ejemplos de uso

```
a = "8"
```

```
b = 3.2 + a
```

```
c = int(a)
```

```
b = 3.2 + c
```

```
b
```

```
d = abs(-4)
```

```
print (d)
```

```
e = max (3, 2, 7, 1)
```

```
e
```

```
print (round(6.4))
```

```
print (round(8.5))
```

```
print (round(2.51))
```

```
cp = complex(9,2)
```

```
print (cp)
```

```
print (ord('A'))
```

```
type (True)
```

```
bool ('True')
```

```
bool (0)
```

# Funciones integradas – listado por categorías

## Math

Function	Description
<code>abs()</code>	Returns absolute value of a number
<code>divmod()</code>	Returns quotient and remainder of integer division
<code>max()</code>	Returns the largest of the given arguments or items in an iterable
<code>min()</code>	Returns the smallest of the given arguments or items in an iterable
<code>pow()</code>	Raises a number to a power
<code>round()</code>	Rounds a floating-point value
<code>sum()</code>	Sums the items of an iterable

## Type Conversion

Function	Description
<code>ascii()</code>	Returns a string containing a printable representation of an object
<code>bin()</code>	Converts an integer to a binary string
<code>bool()</code>	Converts an argument to a Boolean value
<code>chr()</code>	Returns string representation of character given by integer argument
<code>complex()</code>	Returns a complex number constructed from arguments
<code>float()</code>	Returns a floating-point object constructed from a number or string
<code>hex()</code>	Converts an integer to a hexadecimal string
<code>int()</code>	Returns an integer object constructed from a number or string
<code>oct()</code>	Converts an integer to an octal string
<code>ord()</code>	Returns integer representation of a character
<code>repr()</code>	Returns a string containing a printable representation of an object
<code>str()</code>	Returns a string version of an object
<code>type()</code>	Returns the type of an object or creates a new type object

# Lectura básica de datos - input

- Para leer valores de entrada del teclado, en Python usamos la función **input**.

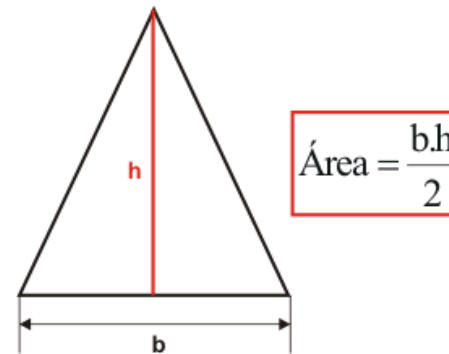
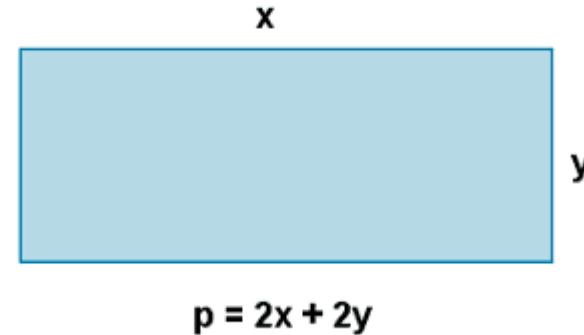
```
variable = input()  
print ("el valor es: ",variable)
```

¿Cuál es el tipo de dato del valor leído desde el teclado?

- ¿Cómo leeríamos una variable de tipo entero?
- ¿Cómo leeríamos una variable de tipo número de punto flotante?
- ¿Cómo leeríamos una variable de tipo cadena de texto?

# Ejercicios

- Se requiere un programa que permita calcular el **perímetro** de un rectángulo dadas las longitudes de sus dos lados.
- Escriba un programa que calcule el **área** de un triángulo dado su base y su altura.



# Operadores Relacionales

```
print (5==4)
```

```
print (2>9)
```

```
x = 6
```

```
y = 12.4
```

```
print (x <= y)
```

```
print (2+7 < 12-3)
```

```
a = 1.1 + 2.2
```

```
b = 3.3
```

```
print (a == b)
```

```
print (abs(a-b)<0.0000001)
```

## Comparison Operators

Operator	Example	Meaning	Result
==	a == b	Equal to	True if the value of a is equal to the value of b False otherwise
!=	a != b	Not equal to	True if a is not equal to b False otherwise
<	a < b	Less than	True if a is less than b False otherwise
<=	a <= b	Less than or equal to	True if a is less than or equal to b False otherwise
>	a > b	Greater than	True if a is greater than b False otherwise
>=	a >= b	Greater than or equal to	True if a is greater than or equal to b False otherwise



# Operadores Lógicos

Operator	Example	Meaning
not	not x	True if x is False False if x is True (Logically reverses the sense of x)
or	x or y	True if either x or y is True False otherwise
and	x and y	True if both x and y are True False otherwise

```
print (True and False)
print (False or True)
print (not(True and False) and True)
```

```
print (5==5.0 and 4>1)
```

```
a = 1
```

```
b = 2
```

```
c = 3
```

```
print (a>b or not(b<c))
```

```
print (c>b and (a+b<c or b+c>a))
```

```
print ((a*2>=b or c//2<b) and b*c<a**c)
```

# Condicionales – clausula if

```
Python

if <expr>:
    <statement>
```

```
edad = 17
if edad < 18:
    print ("Es menor de edad")

nombre = "Juan de la Encarnación"
if len(nombre) > 15:
    print ("Nombre muy largo!")
```

- <expr> es una expresión evaluada en un contexto booleano
- <statement> es una instrucción válida en Python que DEBE estar indentado
- Off-side rule: Los bloques de código en Python se identifican por su indentación

```
Python

1  if <expr>:
2      <statement>
3      <statement>
4      ...
5      <statement>
6  <following_statement>
```

28

# Condicionales – clausula **else**

```
edad = int(input("Digite la edad: "))
if edad < 18:
    print ("Es menor de edad")
else:
    print ("Es mayor de edad")

hora = int(input("Digite la hora: "))
if hora>=18 or hora<=6:
    print ("encender luces")
else:
    print ("apagar luces")
```

```
edad = int(input("Digite la edad: "))
if edad < 2:
    print ("Bebé")
else:
    if edad < 12:
        print ("Niño")
    else:
        if edad <18:
            print ("Adolescente")
        else:
            print ("Adulto")
```

# Condicionales – Clausula elif

```
edad = int(input("Digite la edad: "))  
if edad < 2:  
    print ("Bebé")  
elif edad < 12:  
    print ("Niño")  
elif edad <18:  
    print ("Adolescente")  
else:  
    print ("Adulto")
```

# Ejercicios utilizando Condicionales

---

# Ejercicio

- Escriba un programa que dado un número entero indique si es par o impar



# Ejercicio – Auxilio de Transporte

- Escriba un programa que dado el salario base de un empleado indique si tiene derecho a auxilio de transporte y cuánto es total devengado. Un empleado recibe auxilio de transporte si devenga menos de dos salarios mínimos.
- El salario mínimo establecido por el gobierno para 2020 es de 877.803
- El valor mensual por concepto de auxilio de transporte fue establecido en 102.854



# Ejercicio - Nombres

- Escriba un programa que dados tres nombres, imprima el nombre mas corto y el mas largo de la siguiente forma:

El nombre mas corto es XXXXX

El nombre mas largo es HHHH



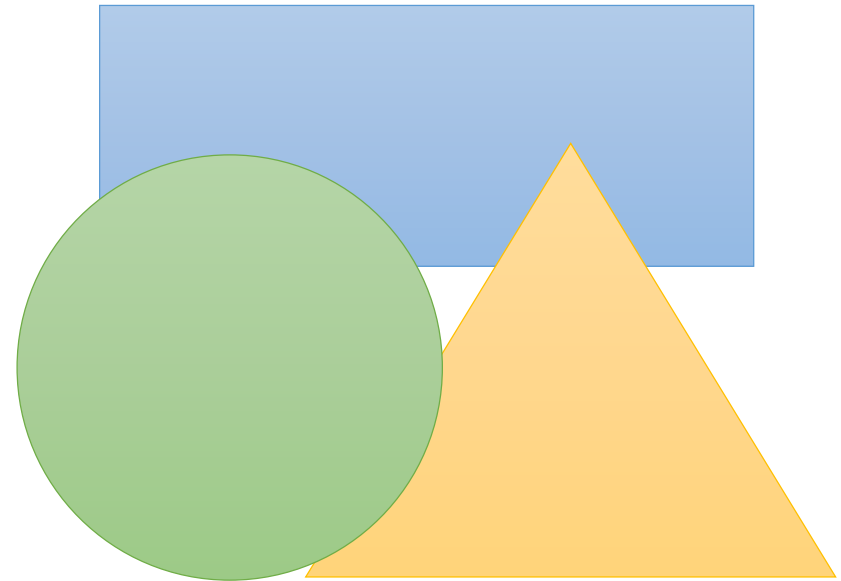
# Ejercicio – Día de la Semana

- Si los días están enumerados del 1 al 7 empezando desde el Lunes. Escriba un programa que dado un número imprima el nombre del día de la semana correspondiente.



# Ejercicio – Menú

- Escriba un programa que imprima un menú de opciones y le pregunte al usuario cuál opción desea. Las opciones del menú son (1) calcular el área de un rectángulo, (2) calcular el perímetro de un triángulo y (3) calcular el área de una circunferencia.

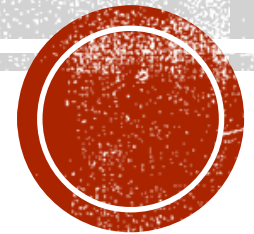


# Ejercicio – Retención en la Fuente

- Escriba un programa que calcule la retención en la fuente que debe descontarse dado el salario de un empleado.
- La retención en la fuente se calcula con base en la tabla de la derecha teniendo en cuenta un valor llamado UVT y que se fija cada año.
- La DIAN fijó el UVT para 2019 en 34.270

Rangos en Uvt	Tarifa marginal	Retención en la fuente	
Desde	hasta		
0	87	0%	0
> 87	145	19%	(Ingreso laboral gravado expresado en UVT menos 87 UVT) x 19%
> 145	335	28%	(Ingreso laboral gravado expresado en UVT menos 145 UVT) x 28% mas 11 UVT
> 335	640	33%	(Ingreso laboral gravado expresado en UVT menos 335 UVT) x 33% mas 64 UVT
> 640	945	35%	(Ingreso laboral gravado expresado en UVT menos 640 UVT) x 35% mas 165 UVT
> 945	2.300	37%	(Ingreso laboral gravado expresado en UVT menos 945 UVT) x 37% mas 272 UVT
> 2.300	En adelante	39%	(Ingreso laboral gravado expresado en UVT menos 2300 UVT) x 39% mas 773 UVT.

# CICLOS Y LISTAS



# ESTRUCTURAS DE CONTROL REPETITIVAS

- Los ciclos permiten repetir un conjunto de instrucciones mientras una condición (expresión que evalúe un valor booleano) sea verdadera
- Son similares a los condicionales (if) en tanto que el bloque de código dentro de ambos (if y ciclos) se ejecuta si la condición es verdadera
- La gran diferencia es que los condicionales evalúan la condición solamente una (1) vez, mientras que los ciclos la evaluarán hasta que sea falsa ejecutando las instrucciones en su interior mientras sea verdadera

```
i = 1
while i<10:
    print ("Python es lo mejor")
    i = i + 1
```

```
from datetime import datetime
opcion = 0
while opcion!=2:
    print ("Menú de opciones")
    print ("1. Hora actual")
    print ("2. Salir")
    opcion = int(input("Digite la opción:"))
    if opcion==1:
        print(datetime.now())
```



# EJERCICIOS

- Escriba un programa que pregunte la cantidad de lados de un polígono, luego pregunte la longitud de cada uno de sus lados e imprima al final su perímetro
- Escriba un programa que pregunte la cantidad de notas de un estudiante en el semestre, luego pregunte por cada una ellas, las promedie e imprima la nota final

- Escriba un programa que pregunte el valor de n y calcule la siguiente sumatoria:

$$\sum_{k=1}^n \frac{1}{k}$$

- Escriba un programa que lea los nombres de la lista de participantes de una reunión e imprima el número total de caracteres (letras) de los nombres de todos los participantes





# MAS EJERCICIOS

- Escriba un programa que calcule la siguiente sumatoria:

$$1+3+5+7+\dots+n$$

- Escriba un programa que indique si un número dado es primo o no

- Escriba un programa que pregunte la cantidad de empleados de una empresa, luego pregunte el nombre y salario de cada uno. El programa deberá imprimir ir imprimiendo el nombre y al lado si debe pagar o no retención en la fuente.



# LISTAS

- Existen varios tipos de datos compuestos, los cuales son utilizados para agrupar varios valores en un solo lugar.
- En Python el más versátil es la Lista, que puede escribirse como una secuencia de valores **separados por coma** encerrados por **corchetes**.

```
cuadrados = [1, 4, 9, 16, 25]  
print (cuadrados)
```

- Las listas están indexadas, es decir, se puede acceder a cada posición utilizando un índice.

```
print (cuadrados[3])  
print (cuadrados[0])
```

- Las listas pueden ser rebanadas (sliced), se puede obtener una porción de ellas con el operador :

```
print (cuadrados[1:3])
```



# OPERACIONES SOBRE LISTAS

- Cuando se rebana una lista, no son obligatorios los índice inicial ni final

```
print (cuadrados[2:])
```

```
print (cuadrados[:3])
```

```
print (cuadrados[:])
```

- Concatenación

```
cuadrados = cuadrados + [36, 49, 63]
```

```
print (cuadrados)
```

- Modificación de valores en una posición

```
cuadrados [7] = 64
```

- Agregar nuevos valores a la lista con la operación append

```
cuadrados.append(81)
```

```
cuadrados.append(100)
```

```
print (cuadrados)
```



# MAS OPERACIONES SOBRE LISTAS

- Es posible cambiar valores de porciones de una lista

```
letras = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

```
print (letras)
```

```
letras[2:5] = ['C', 'D', 'E']
```

```
print (letras)
```

```
letras[1:4] = []
```

```
print (letras)
```

```
letras[:] = []
```

```
print (letras)
```

- De manera similar a las cadenas de texto (strings), podemos conocer la cantidad de elementos con la función **len**

```
print (len(letters))
```

```
print (len(cuadrados))
```

- También es posible anidar listas

```
a = [0,1,2,3,4]
```

```
b = [5,6,7,8,9]
```

```
numbers = [a,b]
```

```
print (numbers)
```



# EL CICLO FOR

- La clausula for es utilizada en Python para iterar sobre cualquier secuencia (lista o cadena de texto, por ejemplo)

```
animales = ["gato","perro","loro"]  
for animal in animales:  
    print (animal, len(animal))
```

- Se puede combinar su uso con la función **range** que genera progresiones aritméticas

```
for i in range(6):  
    print (i)
```

```
for i in range (5,10):  
    print (i)
```

```
print (list(range(3,2,12)))
```



# EJERCICIOS

- Escriba un programa que permita leer  $n$  números enteros, luego lea un (1) número y diga cuántas veces se encuentra dicho número en el listado inicial
- Escriba un programa que dada una cadena de texto, imprima la misma cadena pero al revés

- Escriba un programa que imprima la siguiente sucesión de números:

0, 1, 1, 2, 3, 5, 8, 13, ...,  $n$

Para todos los números de la sucesión menor que  $n$

- Escriba un programa que dado un número  $n$ , cree una matriz (lista de listas) con los valores de 1 a  $n^2$

Por ejemplo si se lee  $n = 3$

$[[1, 2, 3], [4, 5, 6], [7, 8, 9]]$

