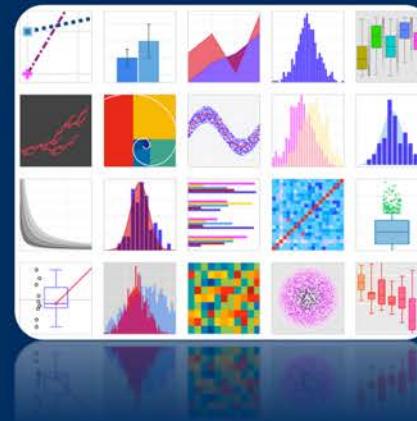


Introducción al análisis y visualización de datos en Python

Día 1 – Introducción a Python



Presentan:
Dr. Ulises Olivares Pinto
Walter André Rosales Reyes
Escuela Nacional de Estudios Superiores Unidad Juriquilla



UNAM
La Universidad
de la Nación



Contenido



1. Introducción a Python (~3 horas - 2 bloques)

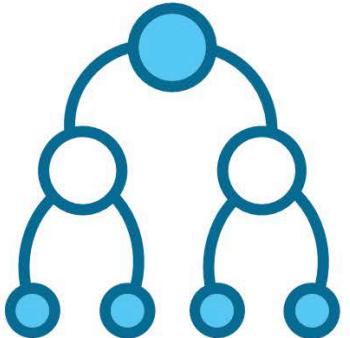
Introducción y antecedentes
Entorno de programación
Variables
Operadores
Break (15 minutos)
Instrucciones de control
Funciones
Librerías



2. Proyecto (Equipos) (30 ~45 minutos)



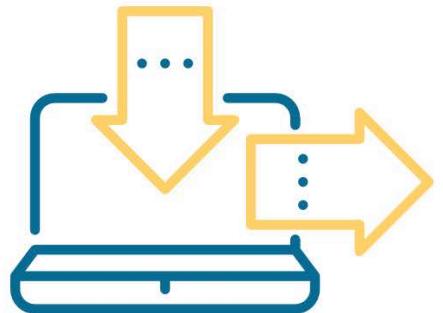
Sequence



Conditional
Statements



Loops



Input / Output



Functions

Conceptos Clave



¿Qué es un lenguaje de programación?

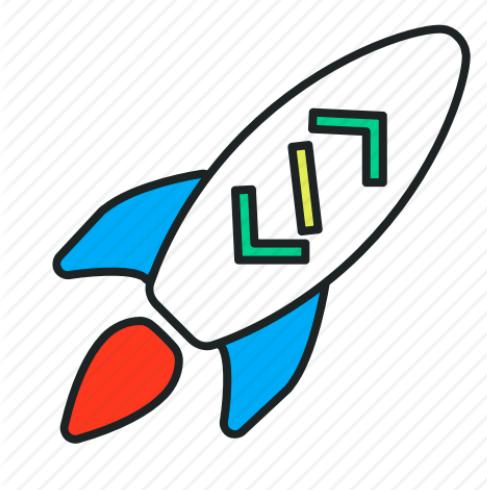
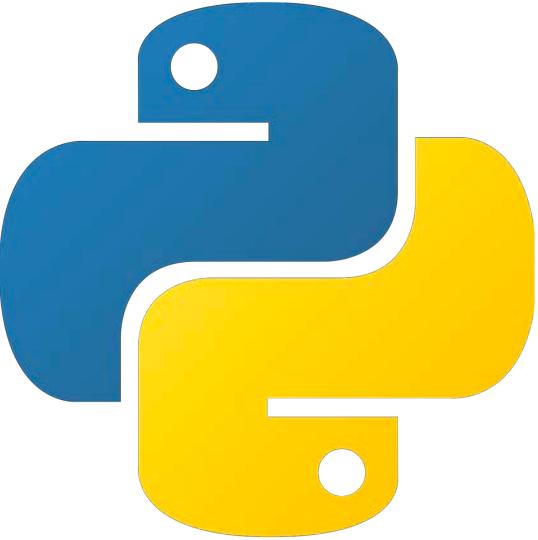
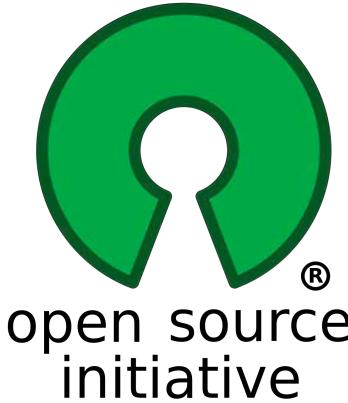
Un lenguaje de programación es un **lenguaje formal** que comprende un conjunto de **instrucciones**, las cuales producen varios tipos de **resultados**.

Un **lenguaje formal** es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados.



¿Qué es Python?

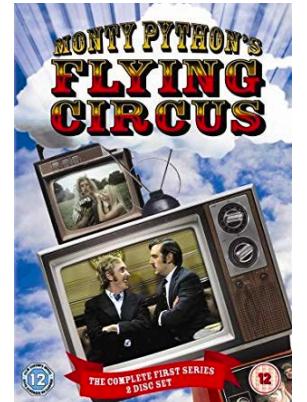
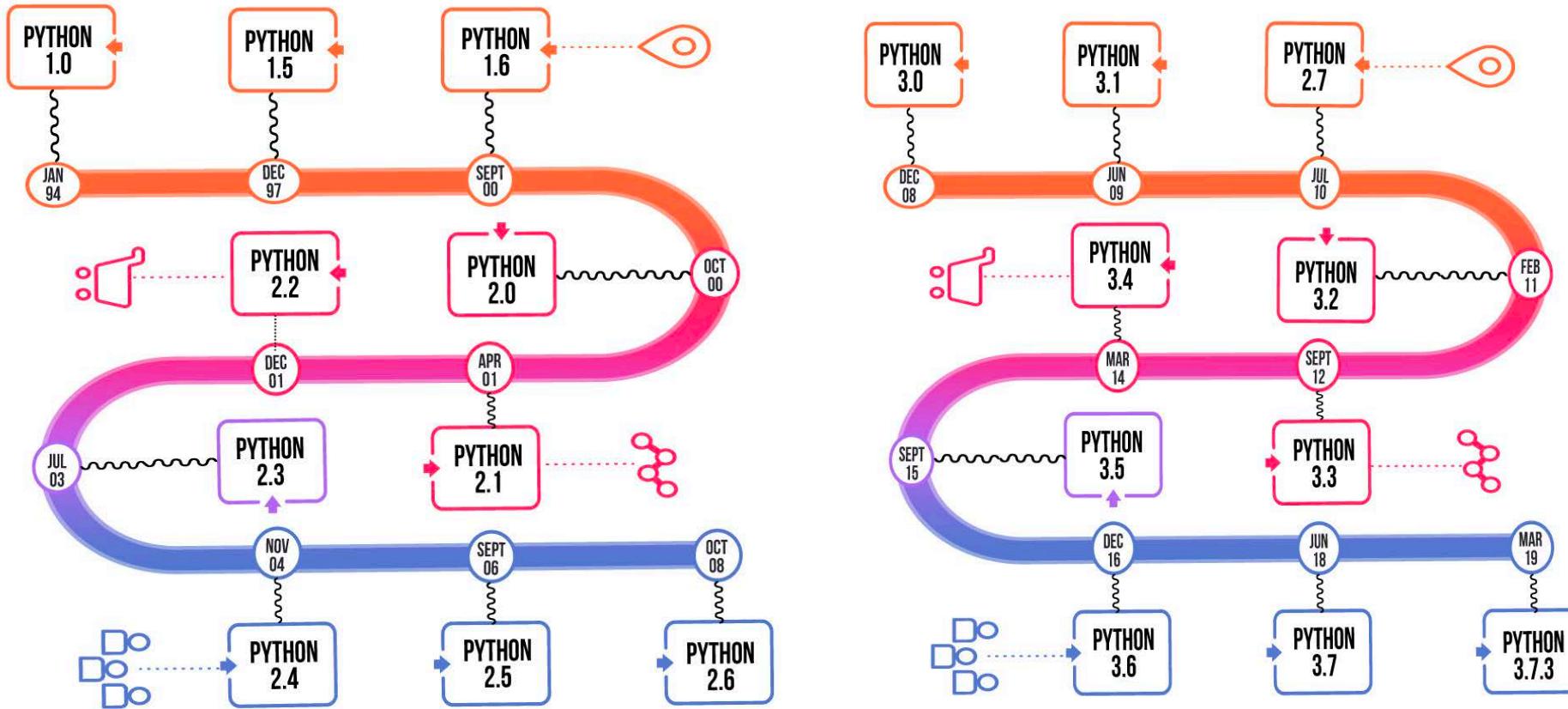
1. Lenguaje Orientado a Objetos
2. Independiente de plataforma
3. Centrado en el tiempo de desarrollo
4. Sintaxis simple y fácil
5. Lenguaje de alto nivel
6. Gestión automática de la memoria
7. ¡Es gratis (código abierto)!



Historia de Python



- Se crea como lenguaje en 1989 por Guido Van Rossum



[Donate](#)

Search

[About](#)[Downloads](#)[Documentation](#)[Community](#)[Success Stories](#)[News](#)[Events](#)

Download the latest version for Mac OS X

[Download Python 3.8.5](#)

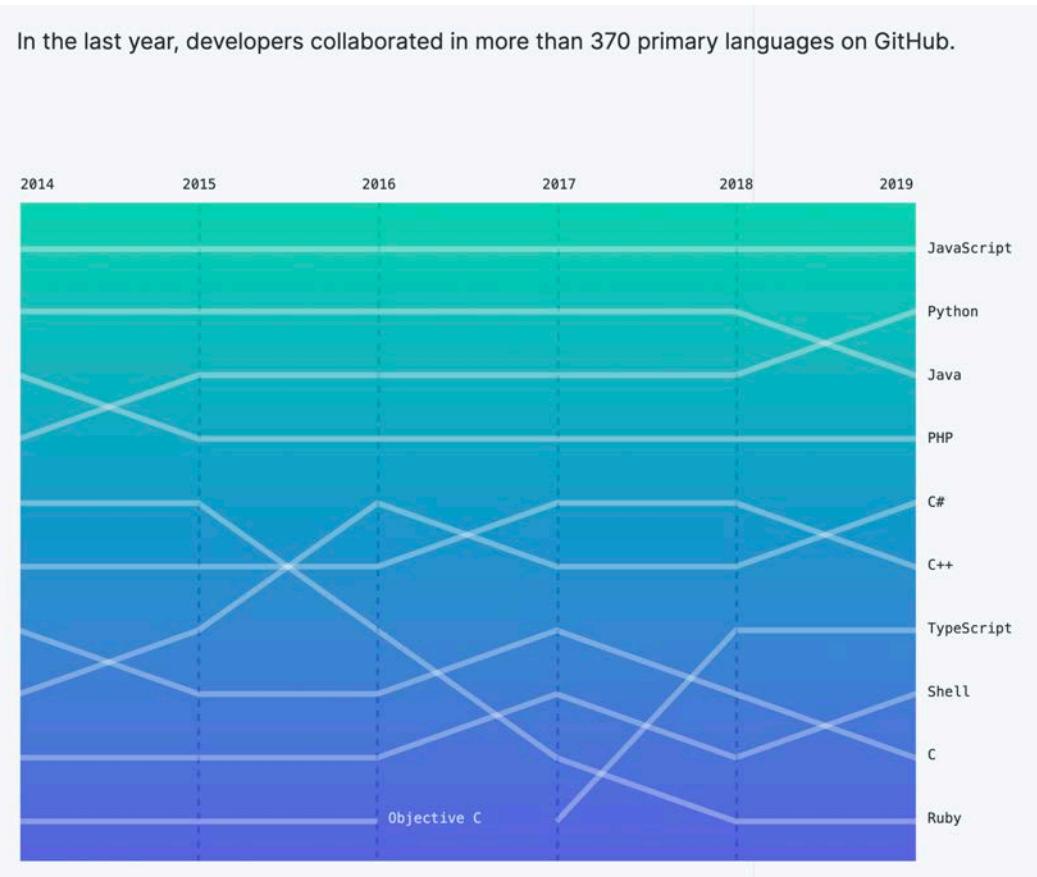
Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#),
[Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

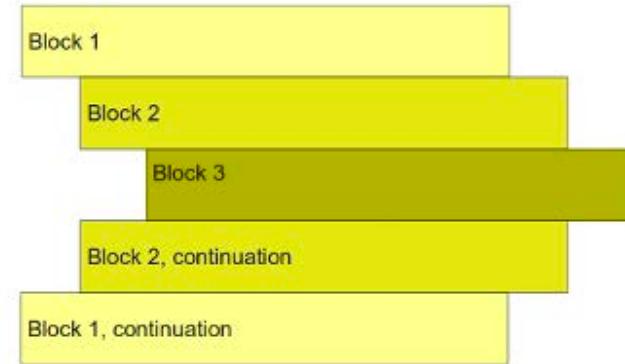


Python es uno de los lenguajes de programación con mayor adopción a nivel mundial.



Características de Python

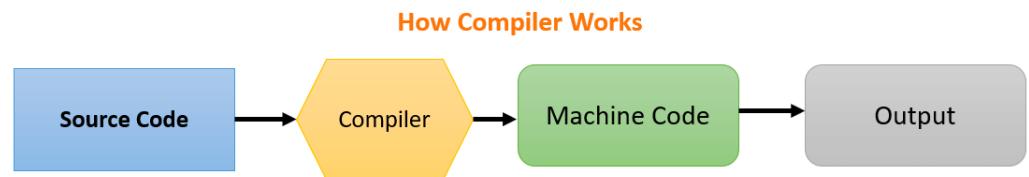
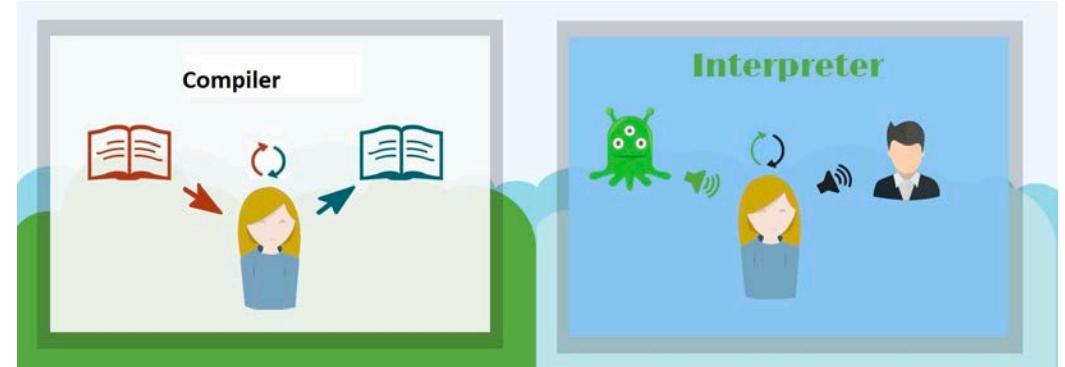
1. Indentación obligatoria
2. Interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
```

Características de Python

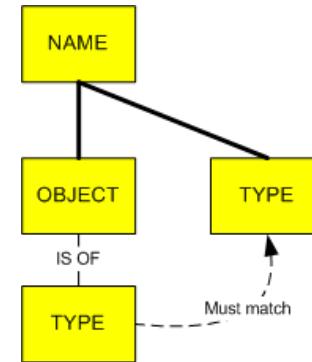
1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)



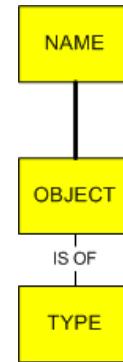
Características de Python

1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)

Estática



Dinámica

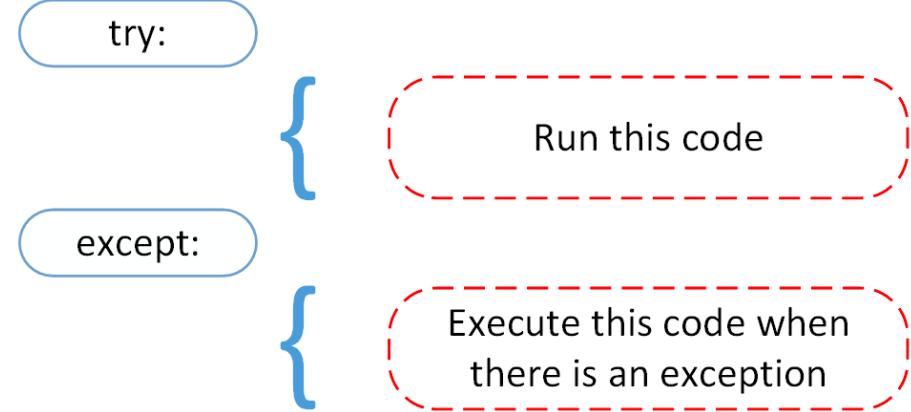


```
my_var = 12  
print(type(my_var))
```

```
my_var = "Hello"  
print(isinstance(my_var,str))
```

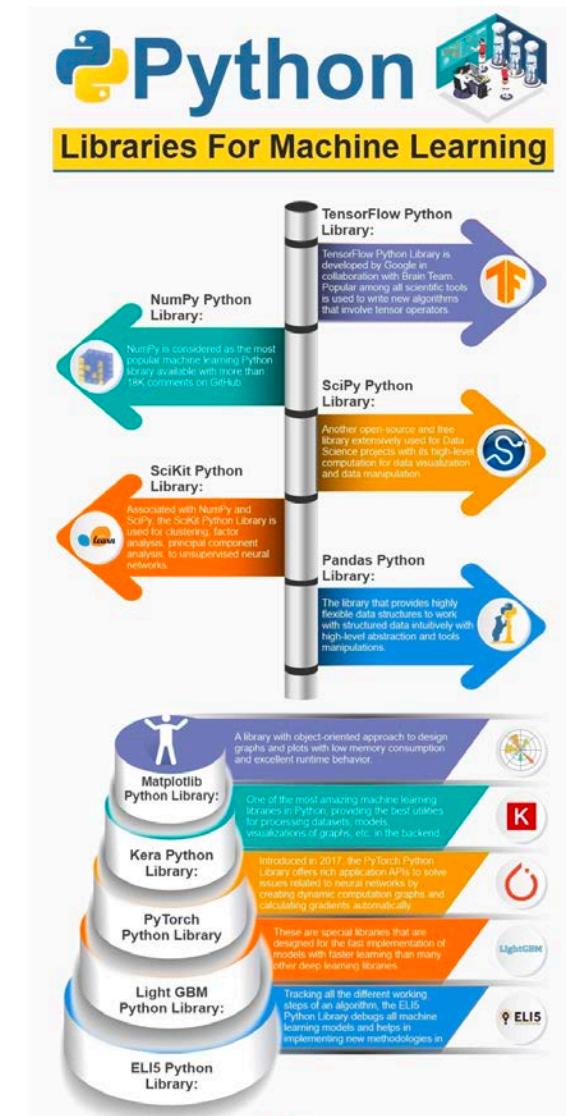
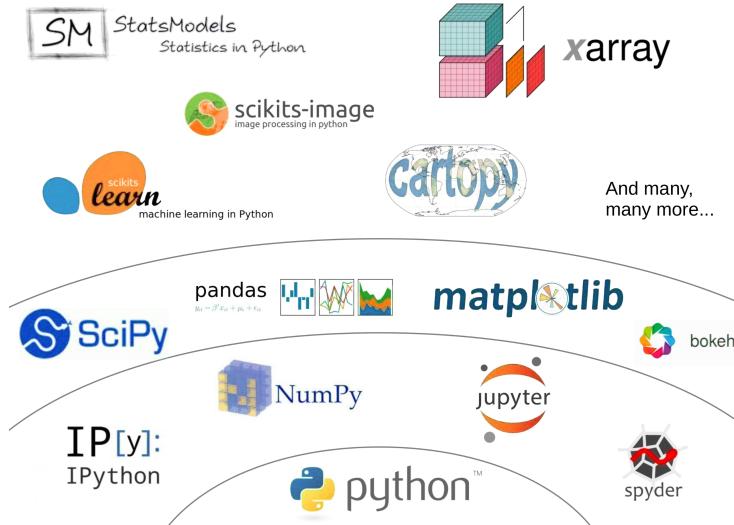
Características de Python

1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)

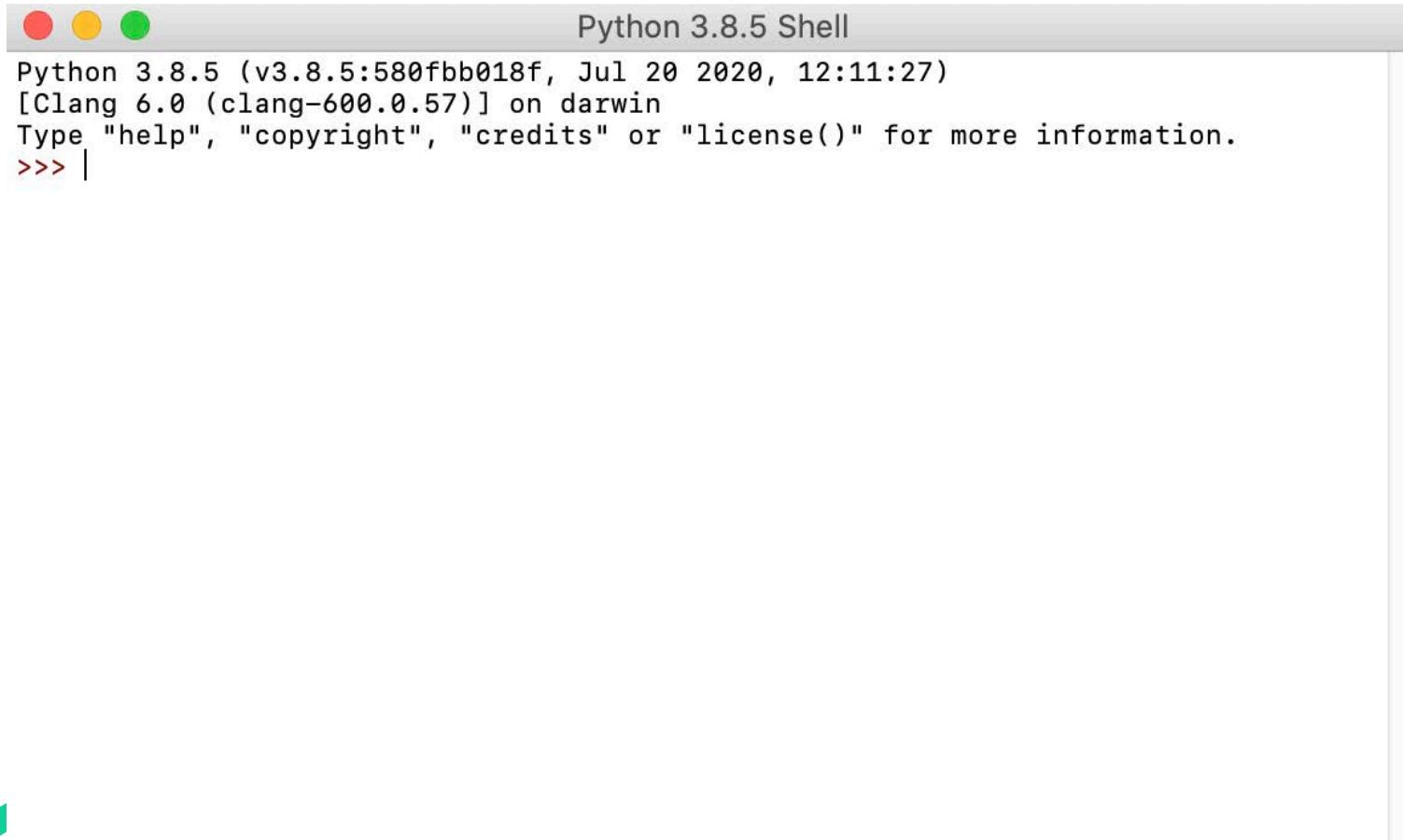


Características de Python

1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)



Entorno de programación – Consola



A screenshot of a Python 3.8.5 Shell window. The title bar reads "Python 3.8.5 Shell". The window contains the following text:

```
Python 3.8.5 (v3.8.5:580fb018f, Jul 20 2020, 12:11:27)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```



IDE – Jupyter

The image shows a Jupyter Notebook interface. At the top right is a large orange circular logo with the word "jupyter" in black. Above the logo are several green dashed horizontal bars. The notebook has a light gray header bar with menu items: View, Insert, Cell, Kernel, Widgets, and Help. Below the header is a toolbar with icons for file operations (New, Open, Save, etc.), cell navigation (Up, Down), running code, and other functions. A status message "Last Checkpoint: 15 minutes ago" is displayed above the main content area. The main content area features a title box with the text "# Introducción a Python - Día 1" in blue. Below it is a code cell labeled "In [2]:" containing the following Python code:

```
# Impresión de prueba
print("Hola mundo, esta es una prueba")
```

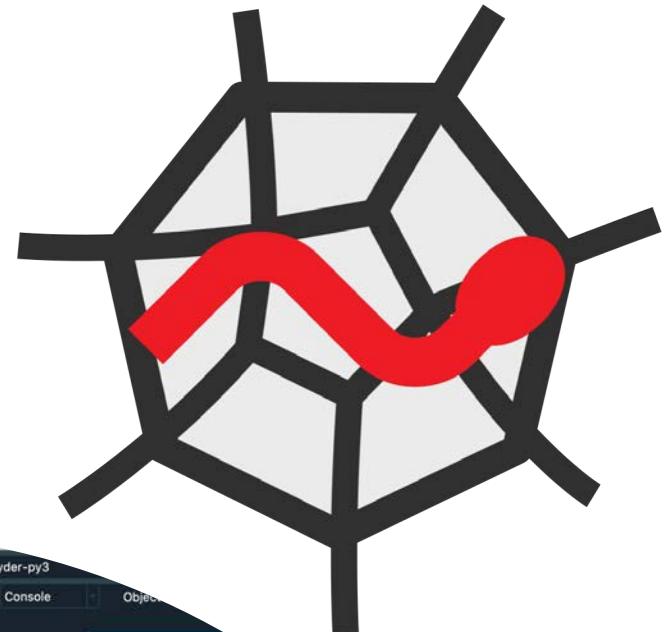
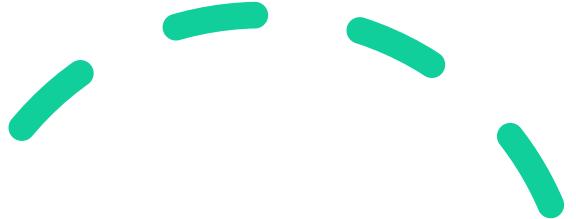
The output of the code is "Hola mundo, esta es una prueba". At the bottom of the screen, there is a partial view of another code cell starting with "In []:". The background of the slide features a white space-themed illustration with stars and a planet.

```
In [2]: # Impresión de prueba
print("Hola mundo, esta es una prueba")
```

Hola mundo, esta es una prueba

In []:

IDE - Spyder



Spyder (Python 3.7)

/Users/ulisesolivares/.spyder-py3

Source Console Object

Usage

Here you can get help on any object by either on the Editor or on the Console.

Help can also be shown automatically after writing next to an object. You can activate this behavior in Help.

New to Spyder? Read our tutorial

Temporary script file.

This is a test in python

```
print("Esta es una prueba")
```

Variable explorer Help Plots Files

Console 1/A

Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type "copyright", "credits" or "license" for more information
IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/ulisesolivares/.spyder-py3/temp.py', wdir='/Users/ulisesolivares/.spyder-py3')
Esta es una prueba

In [2]:

Kite: not running conda: base, Python 3.7.6 Line 10, Col 28 UTF-8 LF RW Mem 68%

IDE – Clion



Flujos de Salida

- Función print

```
Python
```

```
>>>
```

```
>>> print()
```

- '\n' - Línea en blanco (Salto de línea)
 - '\t' - Tabulador
 - '' - Línea vacía
-
- print("Hola Mundo, Bienvenido a Python")
 - mensaje = "Hola Mundo, Bienvenido a Python"
 - print(mensaje)

Flujos de Salida

Python

>>>

```
>>> 'My age is ' + 42
Traceback (most recent call last):
  File "<input>", line 1, in <module>
    'My age is ' + 42
TypeError: can only concatenate str (not "int") to str
```



Python

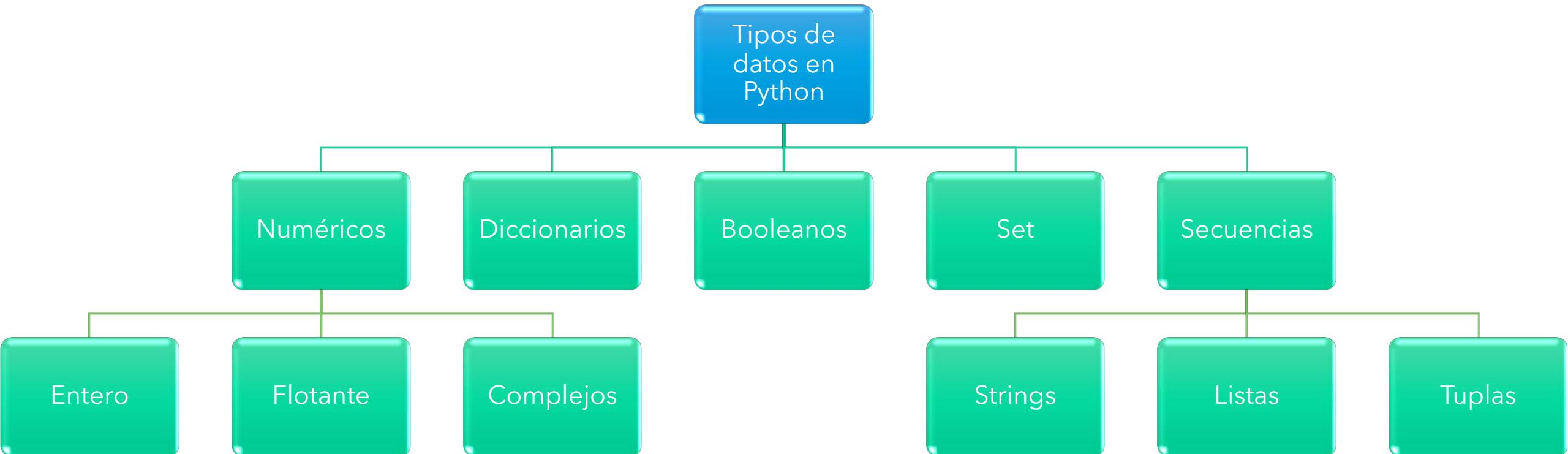
>>>

```
>>> 'My age is ' + str(42)
'My age is 42'
```

```
ulisesolivares2 — python — 80x24
(base) Ulisess-iMac:~ ulisesolivares2$ python
Python 3.7.6 (default, Jan  8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print("Bienvenid@ al curso de Python")
Bienvenid@ al curso de Python
>>>
```

Primer programa en Python

Variables, Tipos y Operadores



Numéricos

Enteros

- $e1 = 1$
- $e2 = 2$

Flotantes

- $f1 = 1.2$
- $f1 = 3.5$

Complejos

- $c1 = (1, 2j)$



Operadores Aritméticos

- + (Suma)
- - (Resta)
- * (Multiplicación)
- / (División)
- % (modulo)



Flujos de Entrada

- `input(<mensaje>)`
- `input("Ingresa un número entero: ")`
- `num =input("Ingresa un número entero:
")`
- `type(num)`



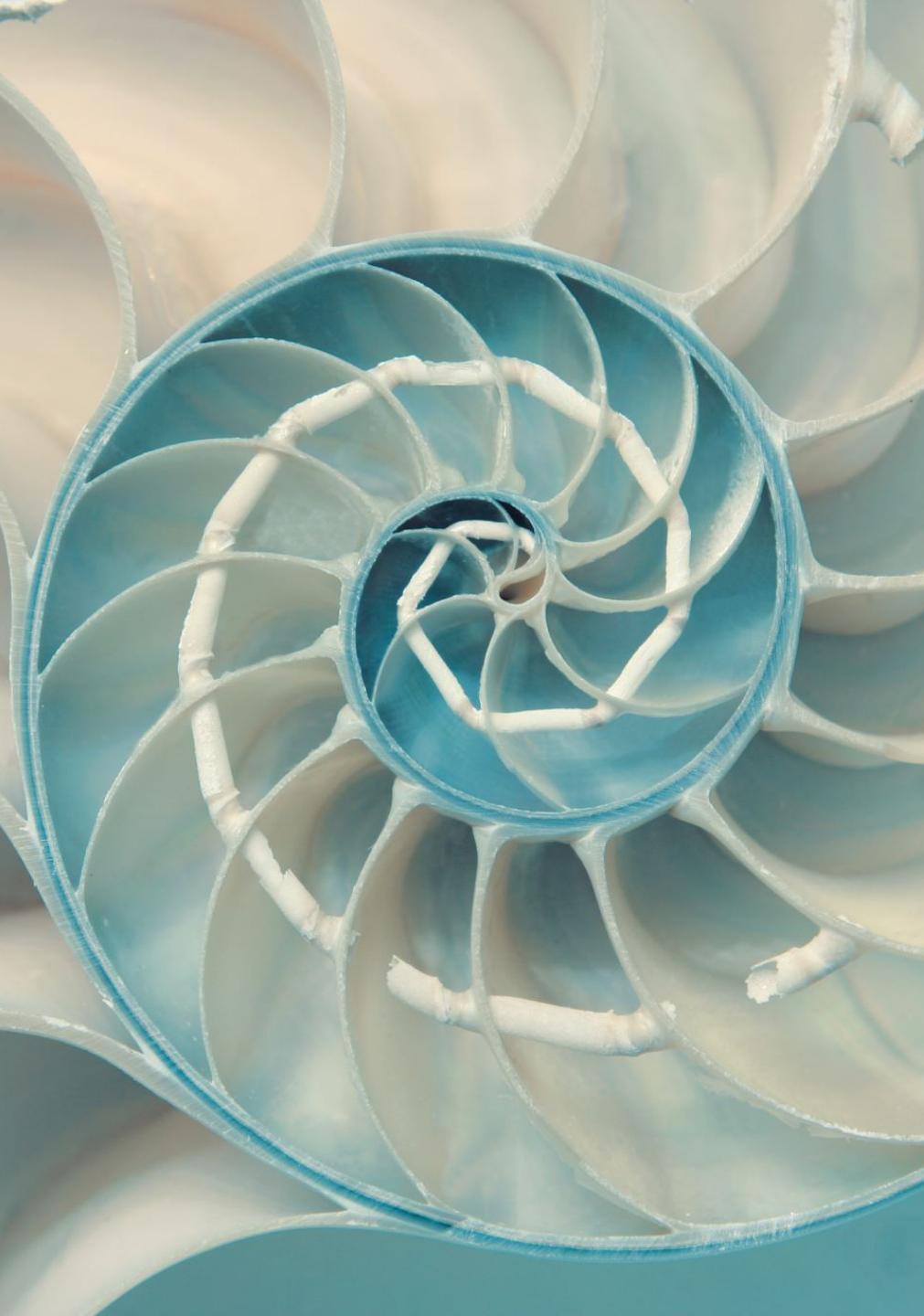
Ejercicio 1- (8 minutos)

- Pedir dos números enteros al usuario.
- Pedir al usuario que seleccione una operación.
 - +, -, *, /, %
- Efectuar la operación sobre ambos números.
- Almacenar el resultado en una tercer variable.

¿Cómo modificar el ejercicio anterior para soportar flotantes?

- ¿Es necesario realizar alguna modificación al código anterior?
- Tipificación dinámica
- Consultar tipo de datos función `type()`





Secuencias



Strings

- s1= "Hola Mundo"

Listas

- l1 = [1, 2, "tres", "cuatro", 5.0, 6.2]

Tuplas

- t1 = (1, "1987")

Strings

```
>>> cadena = "Programa en Python"  
>>> type(cadena)  
<class 'str'>
```

- Dimensión de una cadena `len()`

```
>>> cadena = "Programa en Python"  
>>> len(cadena)  
18
```

- Indexación []

```
>>> cadena = "Programa en Python"  
>>> cadena[:8]  
'Programa'  
>>> cadena[12:]  
'Python'
```

Caracteres :

P	y	t	h	o	n
0	1	2	3	4	5

Índice :

-6	-5	-4	-3	-2	-1

Índice inverso :

Strings

- Mayúsculas `x.upper()`

```
>>> x = "Programa en Python 3"  
>>> x.upper()  
'PROGRAMA EN PYTHON 3'
```

- Minúsculas `x.lower()`

```
>>> x = "PROGRAMA EN PYTHON 3"  
>>> x.lower()  
'programa en python 3'
```

- Tipo Oración `x.title()`

```
>>> x = "programa en python 3"  
>>> x.title()  
'Programa En Python 3'
```

- Reemplazar `x.replace()`

```
>>> x = "Programa En Python 3"  
>> x.replace("E", "e")  
'Programa en Python 3'
```

Strings

- Eliminar espacios al inicio
`x.lstrip()`
- Separar una oración en palabras `x.split()`

```
>>> x = "      Programa en Python      "
>>> x.lstrip()
'Programa en Python      '
```

- Eliminar espacios al final
`x.rstrip()`

```
>>> x.rstrip()
'      Programa en Python'
```

```
>>> x = "Programa en Python"
>>> x.split()
['Programa', 'en', 'Python']
>>> email = "nombre.apellido@ejemplo.tld"
>>> email.split('@')
['nombre.apellido', 'ejemplo.tld']
```



Ejercicio 2 - (8 minutos)

- Se deberán solicitar los siguientes datos a un usuario a través de la terminal:
 - **Nombre completo (Nombre(s), Apellido Paterno, Apellido Materno)**
 - **Edad (Masculino/Femenino)**
 - **Sexo**
1. Se deberan imprimir las iniciales del usuario. Ej. Juan Pérez López (JPLL)
 2. Se deberá reemplazar el sexo solo por M/F
 3. Se deberá imprimir el nombre completo en el siguiente orden: Apellido Paterno, Apellido Materno Nombre(s)
 4. Asegurar que la impresión se realice en forma de Oración (Nombre Propio)

Listas

```
>>> numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> elementos = [3, 'a', 8, 7.2, 'hola']
```

- Función `list()`

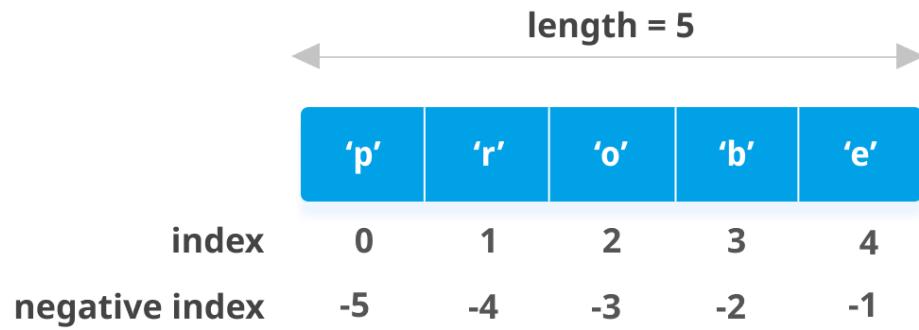
```
>>> vocales = list('aeiou')  
>>> vocales  
['a', 'e', 'i', 'o', 'u']
```

- Lista vacía

```
>>> lista_1 = [] # Opción 1  
>>> lista_2 = list() # Opción 2
```



Listas



```
>>> vocales = ['a', 'e', 'i', 'o', 'u']
>>> len(vocales)
5
```

- Indexación

```
>>> vocales = ['a', 'e', 'i', 'o', 'u']
>>> vocales[-1]
'u'
>>> vocales[-4]
'e'
```

- Subconjuntos

```
>>> vocales = ['a', 'e', 'i', 'o', 'u']
>>> vocales[2:3] # Elementos desde el índice 2 hasta el índice 3-1
['i']
>>> vocales[2:4] # Elementos desde el 2 hasta el índice 4-1
['i', 'o']
>>> vocales[:] # Todos los elementos
['a', 'e', 'i', 'o', 'u']
>>> vocales[1:] # Elementos desde el índice 1
['e', 'i', 'o', 'u']
>>> vocales[:3] # Elementos hasta el índice 3-1
['a', 'e', 'i']
```

Métodos para la clase List

- Pertenencia de un elemento a una lista función `in()`

```
# Lista desordenada de números enteros
>>> numeros = [3, 2, 6, 1, 7, 4]
```

```
>>> 3 in numeros
```

- Ordenamientos

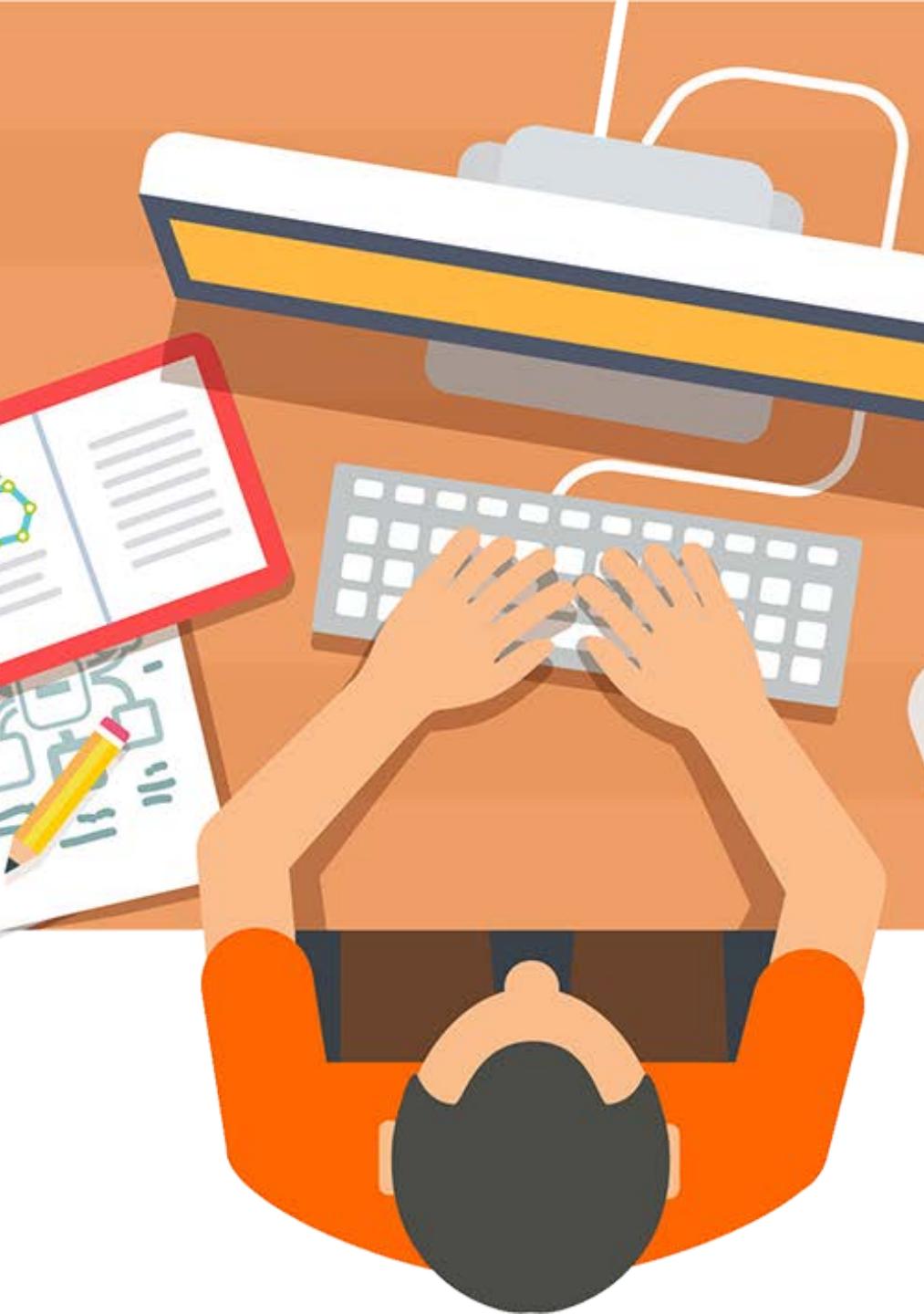
```
# Identidad del objeto numeros
>>> id(numeros)
4475439216
```

```
# Se llama al método sort() para ordenar los elementos de la lista
>>> numeros.sort()
>>> numeros
[1, 2, 3, 4, 6, 7]
```

```
# Se comprueba que la identidad del objeto numeros es la misma
>>> id(numeros)
4475439216
```



Método	Descripción
<code>append()</code>	Añade un nuevo elemento al final de la lista.
<code>extend()</code>	Añade un grupo de elementos (iterables) al final de la lista.
<code>insert(indice, elemento)</code>	Inserta un elemento en una posición concreta de la lista.
<code>remove(elemento)</code>	Elimina la primera ocurrencia del elemento en la lista.
<code>pop([i])</code>	Obtiene y elimina el elemento de la lista en la posición i. Si no se especifica, obtiene y elimina el último elemento.
<code>clear()</code>	Borra todos los elementos de la lista.
<code>index(elemento)</code>	Obtiene el índice de la primera ocurrencia del elemento en la lista. Si el elemento no se encuentra, se lanza la excepción <code>ValueError</code> .
<code>count(elemento)</code>	Devuelve el número de ocurrencias del elemento en la lista.
<code>sort()</code>	Ordena los elementos de la lista utilizando el operador <.
<code>reverse()</code>	Obtiene los elementos de la lista en orden inverso.
<code>copy()</code>	Devuelve una copia poco profunda de la lista.



Ejercicio 3 - (8 -10 minutos)

- Inicializar la siguiente lista

```
objetos = ["celular", "laptop", "cámara", "televisión",  
          "bocinas", 100, 310.28, 27.00, 1000, 120.2,  
          300,]
```

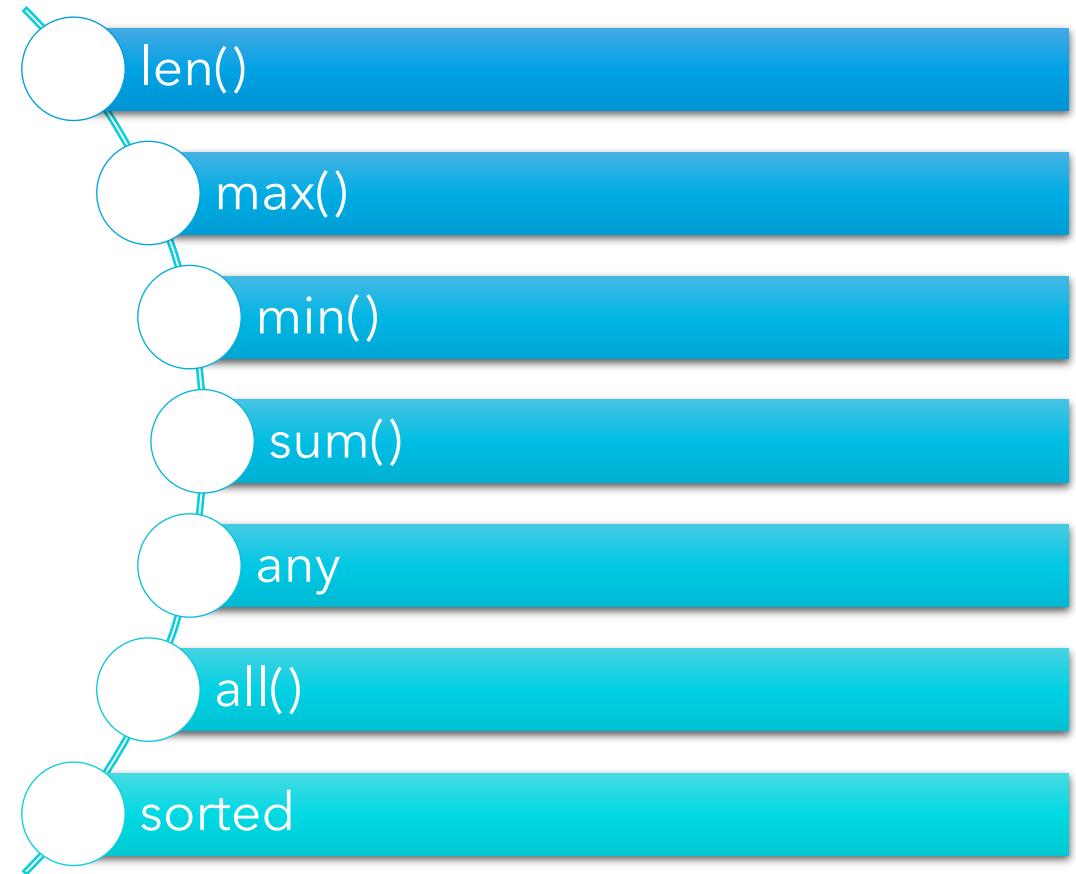
1. Separar los elementos en dos listas:
 - a. Lista numérica
 - b. Lista de caracteres.
2. Ordenar la lista numérica en orden ascendente y descendente.
3. Ordenar la lista de strings en orden alfabético (A-Z) y (Z-A).

Tuplas

- Una tupla es un conjunto **inmutable** de elementos del mismo o distinto tipo.

```
>>> a = ()  
>>> b = tuple()
```

```
>>> t=(31, "Agosto", 2020)  
>>> t[0] 31  
>>> t[1] 'Agosto'  
>>> t[2] 2020
```



Tuplas

Métodos propios de tuplas

Index

Retorna el índice del elemento que se proporciona.

```
>>> a=(1,2,3,2,4,5,2)  
>>> a.index(2)
```

Count

Retorna el número de ocurrencias del elemento que se proporciona.

```
>>> a.count(2)
```

Ejercicio 4 - (5 minutos)

- Inicializar una tupla que contenga las calificaciones de una asignatura para 10 estudiantes.
 - Ej. cal = (8, 9 , 5, 10, 8, 7.2, 9, 8, 8, 10)
1. Se deberá obtener:
 - a. Promedio
 - b. Calificación máxima
 - c. Calificación mínima
 - d. Moda
 - e. Ordenar calificaciones



Booleanos

El tipo booleano sólo puede tener dos valores: **True** (verdadero) y **False** (falso). Estos valores son especialmente importantes para las expresiones condicionales y los ciclos.

Valor booleano falso.	<code>False</code>
Valor booleano verdadero.	<code>True</code>

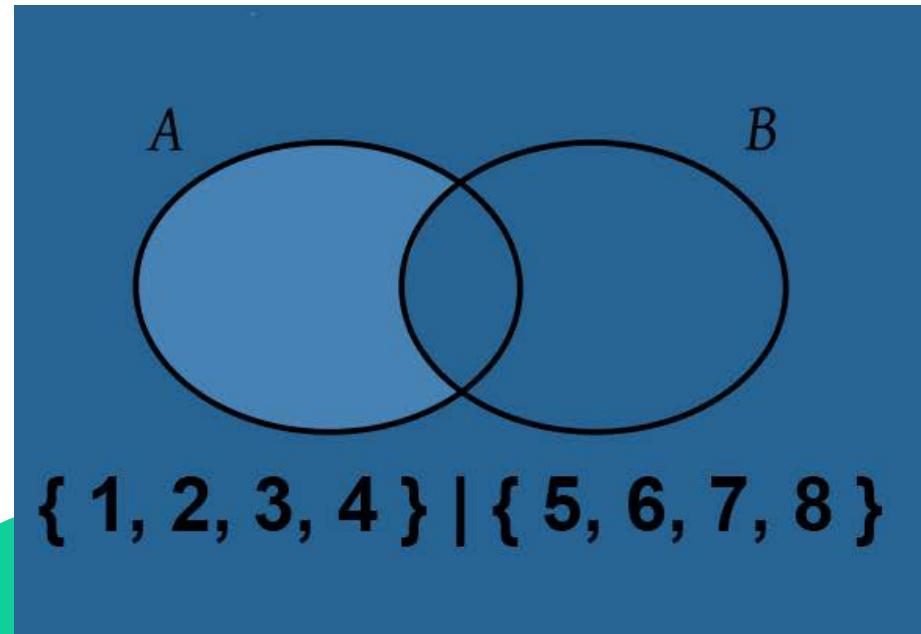
```
1 >>> type(True)  
2 <class 'bool'>
```



Operadores Relacionales

> >= < <= == !=
mayor mayor-igual menor menor-igual igual distinto

Operador	Descripción	Ejemplo
a == b	¿Son iguales a y b?	5 == 5 # verdadero 6 == 8 # falso
a != b	¿Son distintos a y b?	21 != 5 # verdadero 6 != 6 # falso
a < b	¿Es a menor que b?	5 < 9 # verdadero 6 < 2 # falso
a > b	¿Es a mayor que b?	6 > 5 # verdadero 6 > 6 # falso
a <= b	¿Es a menor o igual que b?	5 <= 5 # verdadero 6 <= -10 # falso
a >= b	¿Es a mayor o igual que b?	5 >= 5 # verdadero 6 >= 8 # falso



Set (Conjuntos)

Un conjunto es una colección **no ordenada** de **objetos únicos**. Python provee este tipo de datos «por defecto». Los conjuntos son ampliamente utilizados en **lógica** y **matemáticas**.

```
>>> s1 = {1, 2, 3, 4}
```

```
>>> s2 = {True, 3.14, None, False, "Hola mundo", (1, 2)}
```

Set

Un conjunto **no puede incluir objetos mutables** como listas, diccionarios, e incluso otros conjuntos.

```
1. >>> s = {[1, 2]}\n2. Traceback (most recent call last):\n3. ...\n4. TypeError: unhashable type: 'list'
```

Sets - Operaciones Básicas

- Unión |

Lista todos los elementos de ambos conjuntos.

```
1. >>> a = {1, 2, 3, 4}  
2. >>> b = {3, 4, 5, 6}  
3. >>> a | b  
4. {1, 2, 3, 4, 5, 6}
```

- Intersección &

Lista los elementos en común entre ambos conjuntos.

```
1. >>> a & b  
2. {3, 4}
```

- Diferencia -

```
1. >>> a = {1, 2, 3, 4}  
2. >>> b = {2, 3}  
3. >>> a - b  
4. {1, 4}
```

Set (Métodos)

Método	Descripción		
<code>add(e)</code>	Añade un elemento al conjunto.	<code>isdisjoint(iterable)</code>	Devuelve <code>True</code> si dos conjuntos son disjuntos.
<code>clear()</code>	Elimina todos los elementos del conjunto.	<code>issubset(iterable)</code>	Devuelve <code>True</code> si el conjunto es subconjunto del <code>iterable</code> .
<code>copy()</code>	Devuelve una copia superficial del conjunto.	<code>issuperset(iterable)</code>	Devuelve <code>True</code> si el conjunto es superconjunto del <code>iterable</code> .
<code>difference(iterable)</code>	Devuelve la diferencia del conjunto con el <code>iterable</code> como un conjunto nuevo.	<code>pop()</code>	Obtiene y elimina un elemento de forma aleatoria del conjunto.
<code>difference_update(iterable)</code>	Actualiza el conjunto tras realizar la diferencia con el <code>iterable</code> .	<code>remove(e)</code>	Elimina el elemento del conjunto. Si no existe lanza un error.
<code>discard(e)</code>	Elimina, si existe, el elemento del conjunto.	<code>symmetric_difference(iterable)</code>	Devuelve la diferencia simétrica del conjunto con el <code>iterable</code> como un conjunto nuevo.
<code>intersection(iterable)</code>	Devuelve la intersección del conjunto con el <code>iterable</code> como un conjunto nuevo.	<code>symmetric_difference_update(iterable)</code>	Actualiza el conjunto tras realizar la diferencia simétrica con el <code>iterable</code> .
<code>intersection_update(iterable)</code>	Actualiza el conjunto tras realizar la intersección con el <code>iterable</code> .	<code>union(iterable)</code>	Devuelve la unión del conjunto con el <code>iterable</code> como un conjunto nuevo.
		<code>update(iterable)</code>	Actualiza el conjunto tras realizar la unión con el <code>iterable</code> .

Sets Mutables vs Inmutables

Clase	Tipo	Notas	Ejemplo
<code>set</code>	Conjuntos	Mutable, sin orden, no contiene duplicados.	<code>set([4.0, 'Carro', True])</code>
<code>frozenset</code>	Conjuntos	Inmutable, sin orden, no contiene duplicados.	<code>frozenset([4.0, 'Carro', True])</code>

Ejercicio 5 – (10 minutos)

- Dados los siguientes conjuntos:
 - `set1 = {10, 20, 30, 40, 50}`
 - `set2 = {60, 70, 80, 90, 10}`
1. Se deberán imprimir los **elementos en común**.
 2. Se deberán imprimir **todos los elementos sin repeticiones** en ambos conjuntos.
 3. Se deberá imprimir el primer conjunto **sin los elementos {10, 20, 30}**, se deberá usar una sola instrucción.

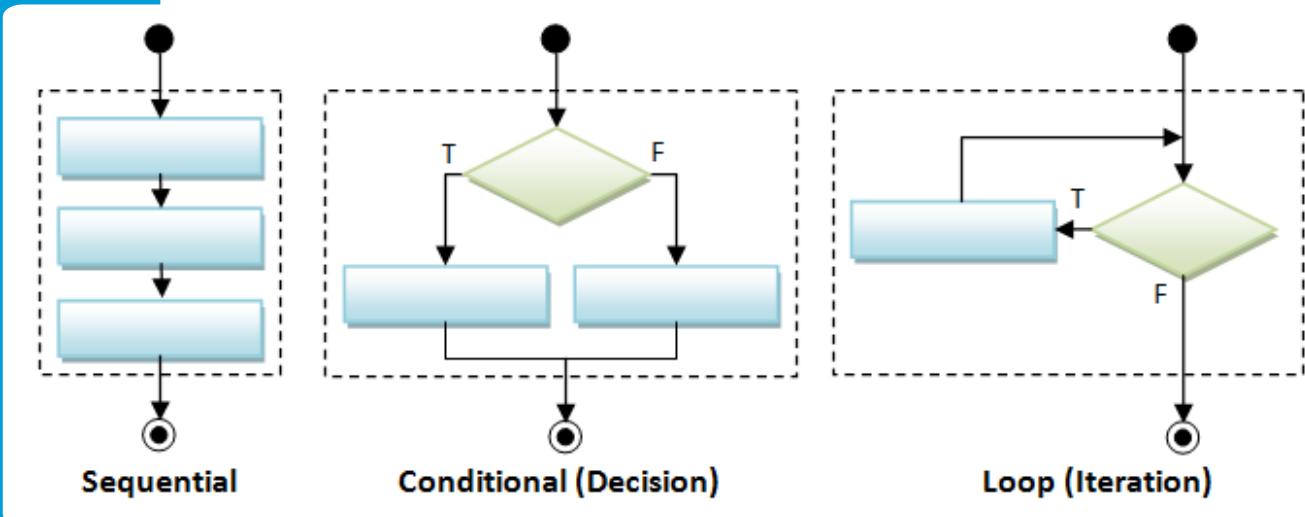


Instrucciones de Control en Python



```
34  
35     self.debug = debug  
36     self.logger = logger  
37     if path:  
38         self._path = path  
39     else:  
40         self._path = None  
41  
42     @classmethod  
43     def from_set(cls, settings, path=None):  
44         debug = settings.get("logger.debug", False)  
45         return cls(path, debug=debug)  
46  
47     def request_seen(self, request):  
48         fp = self.request_fingerprint(request)  
49         return fp in self._seen_requests
```

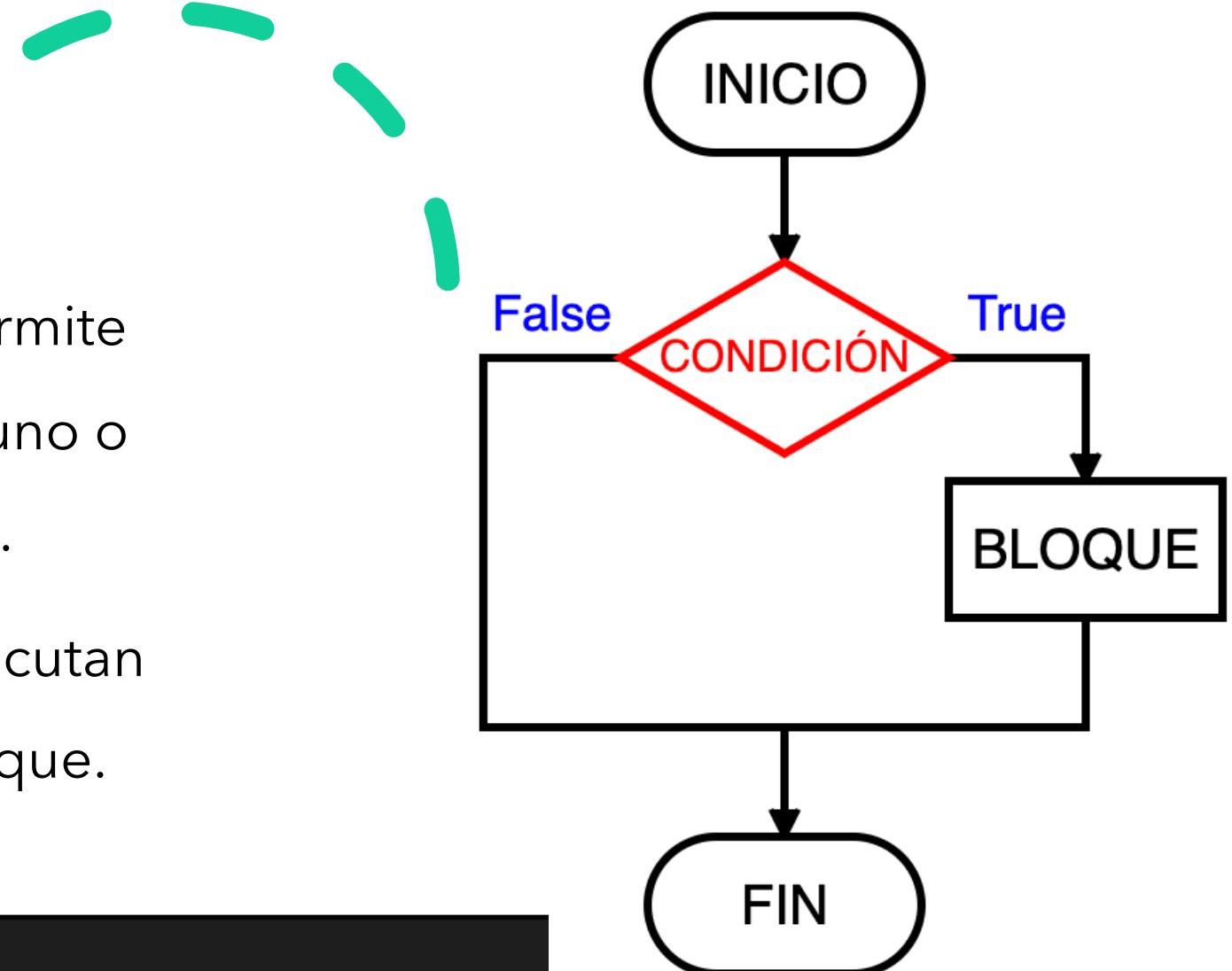
Estructuras de control de flujo



Condicional (IF)

- Esta secuencia de control permite **condicionar** la ejecución de uno o varios bloques de sentencias.
- Si la condición es **True**, se ejecutan las sentencias dentro del bloque.

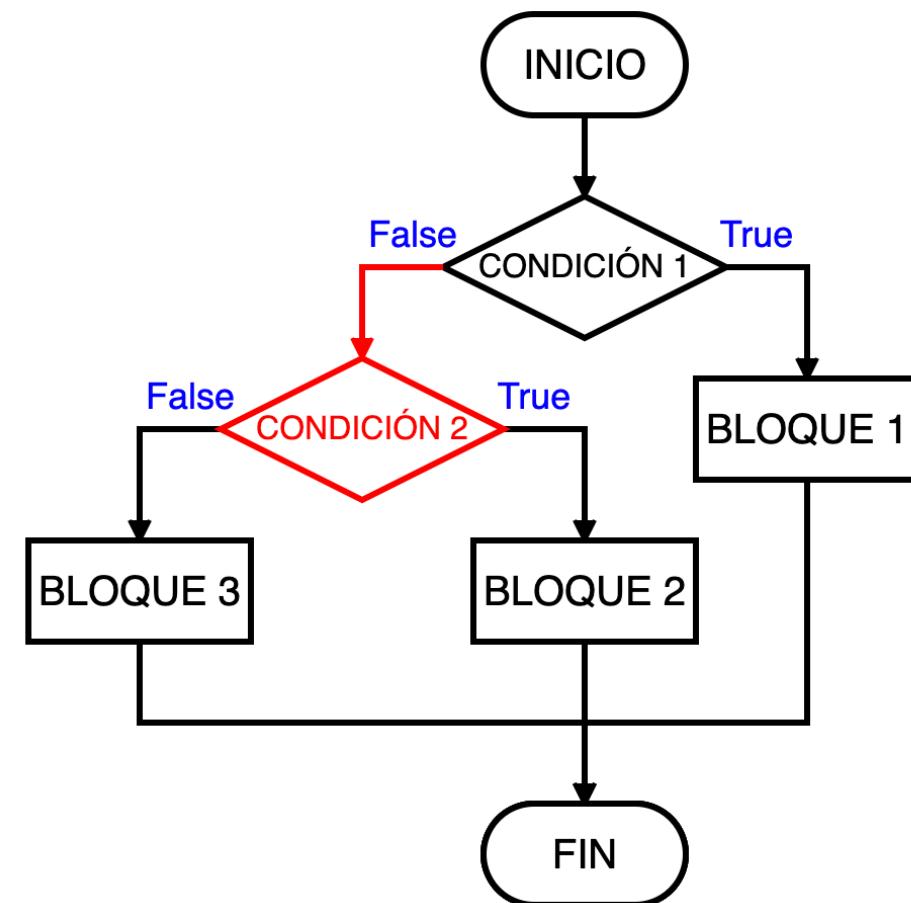
```
if condición:  
    aquí van las órdenes que se ejecutan si la condición es cierta  
    y que pueden ocupar varias líneas
```



```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
```

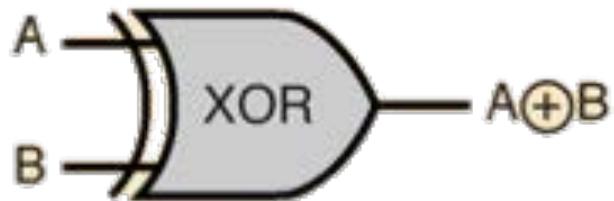
```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

Condicional (if...elif... else)

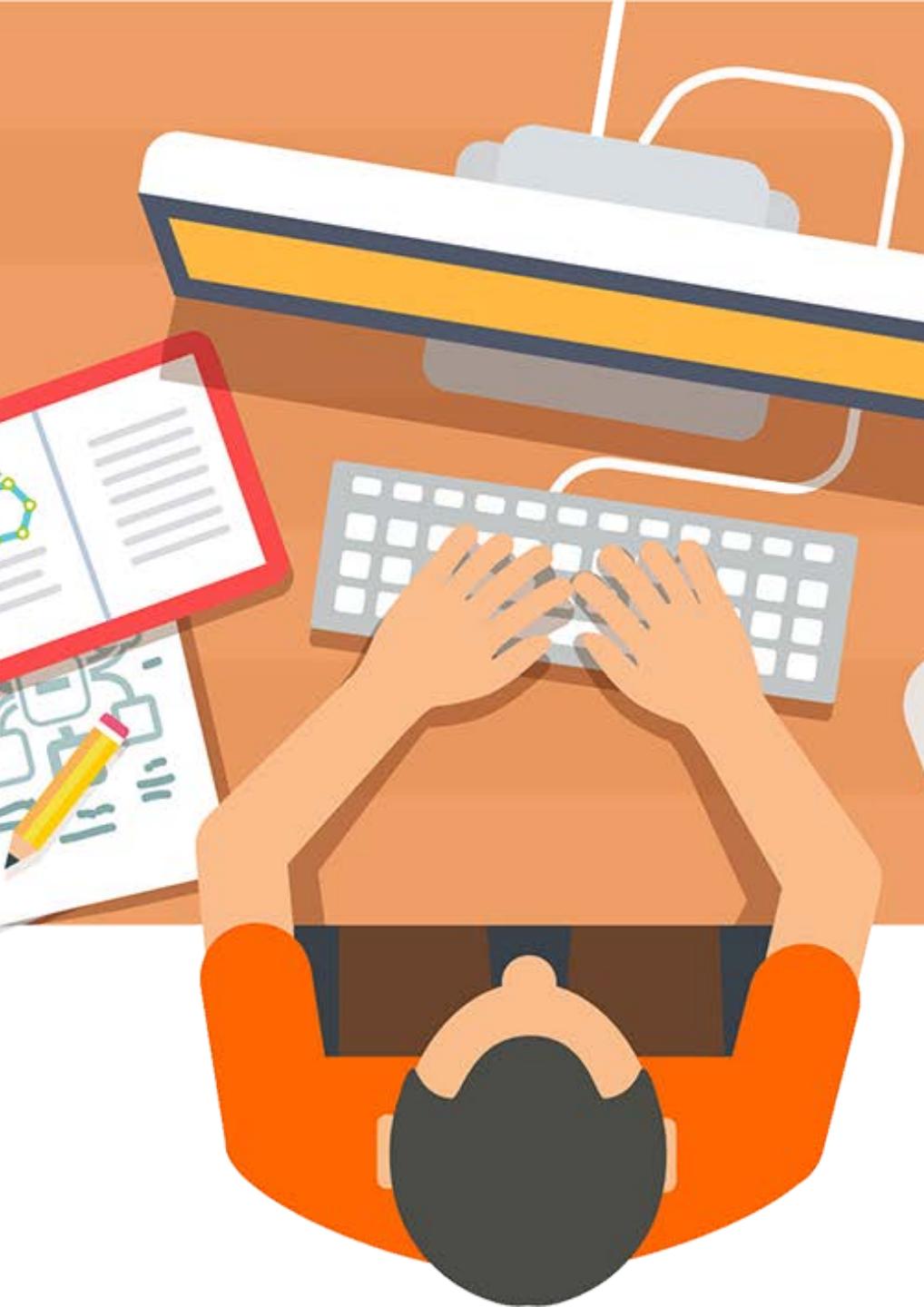


Operadores Lógicos

Operador	Ejemplo	Explicación	Resultado
and	<code>5 == 7 and 7 < 12</code>	False and False	False
and	<code>9 < 12 and 12 > 7</code>	True and True	True
and	<code>9 < 12 and 12 > 15</code>	True and False	False
or	<code>12 == 12 or 15 < 7</code>	True or False	True
or	<code>7 > 5 or 9 < 12</code>	True or True	True
xor	<code>4 == 4 xor 9 > 3</code>	True o True	False
xor	<code>4 == 4 xor 9 < 3</code>	True o False	True

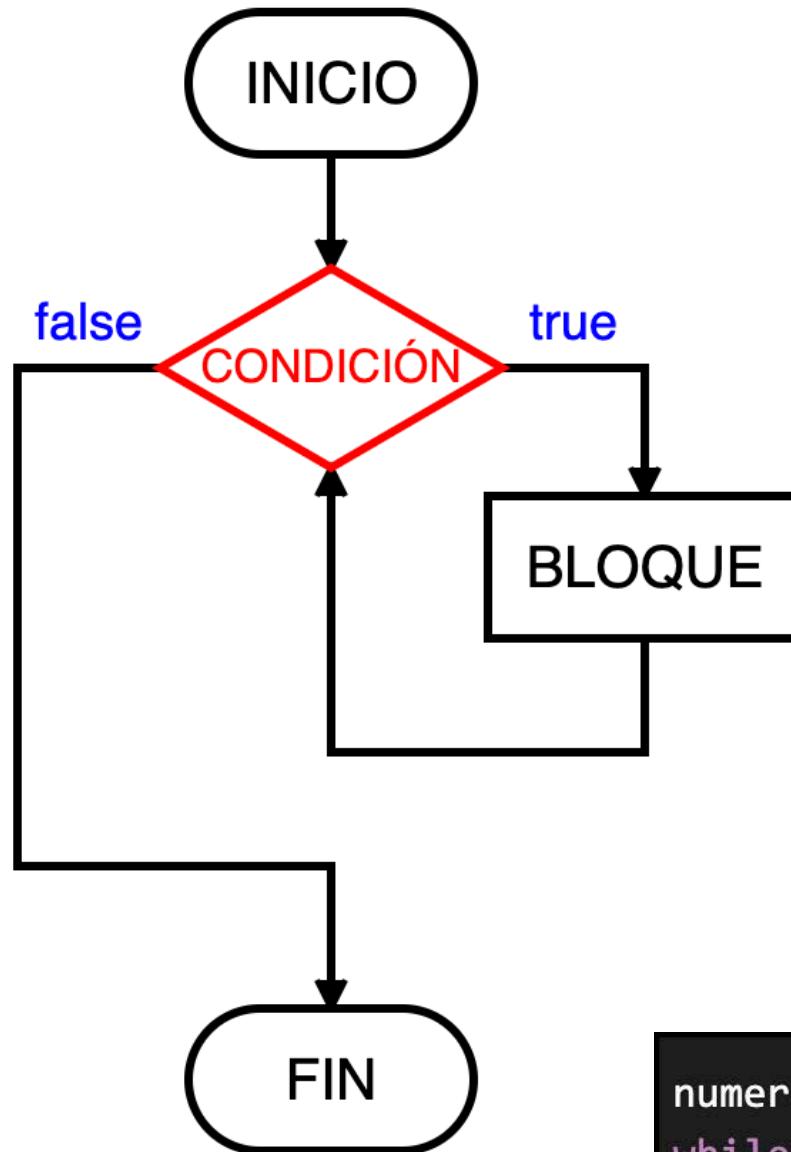


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



Ejercicio 6 (10-15 minutos)

1. Se deberá determinar si un número ingresado por el usuario es **par** o **impar**.
2. Escribir un código para determinar el **mayor** de **dos números**.
3. Escribir un código para determinar el **mayor** de **tres números**.
4. Escribir un código para determinar si un número se encuentra en un intervalo $[0, 10]$.



Ciclos (while)

- Permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (La evaluación sea verdadera).

```

i = 1
while i <= 3:
    print(i)
    i += 1
  
```

```

numero = int(input("Escriba un número positivo: "))
while numero < 0:
    print("¡Ha escrito un número negativo! Inténtelo de nuevo")
    numero = int(input("Escriba un número positivo: "))
  
```

Ciclos Infinitos (while True:)

```
import time

segundero = 0
maximo = 5

while True:
    time.sleep(1)      # esperamos un segundo
    segundero += 1    # segundero = segundero + 1
    print(segundero)

    if segundero == maximo:
        print("Se ha llegado al límite, rompiendo el bucle infinito")
        break
```

La serie de Fibonacci

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

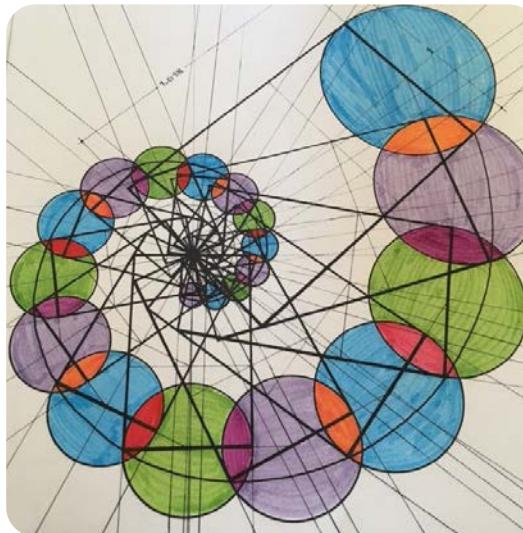
$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

...



Ejercicio 7 (10 -15 minutos)

1. Se deberá calcular la serie de Fibonacci hasta $n = 100$ utilizando un ciclo while.

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

...



Ciclos FOR

- Los ciclos For permiten ejecutar una o varias instrucciones de forma iterativa, una vez por elemento en la colección.

Iterar sobre un rango de valores

```
1. >>> for contador in range(1,10):  
2. ...     print contador,  
3. ...  
4. 1 2 3 4 5 6 7 8 9  
5. >>>
```

Iterar sobre una estructura

```
1. >>> numeros = [0, 1, 2, 3, 4, 5]  
2. >>> for numero in numeros:  
3. ...     print numero,  
4. ...  
5. 0 1 2 3 4 5  
6. >>>
```

Ciclos FOR

Iterar sobre un diccionario:

```
1. >>> frutas = {'Fresa':'roja', 'Limon':'verde', 'Papaya':'naranja', 'Manzana':'amarilla', 'Guayaba':'rosa'}
2. >>> for nombre, color in frutas.items():
3. ...     print nombre, "es de color", color
4. ...
5. Fresa es de color roja
6. Limon es de color verde
7. Manzana es de color amarilla
8. Papaya es de color naranja
9. Guayaba es de color rosa
```



Ejercicio 8 – Ciclos For

(10 -15 minutos)

- Se deberán pedir al usuario un total de 10 números enteros.
1. Se deberá realizar una sumatoria de todos los elementos.
 2. Se deberá calcular el promedio.
 3. Se deberá obtener el elemento más pequeño.
 4. Se deberá obtener el elemento más grande.

Funciones

- Una función es un **bloque de código** con un **nombre** asociado, que recibe ningún o más de un **argumento** como entrada, ejecuta una secuencia de **sentencias** y devuelven un **valor**. Estos bloques pueden ser llamados cuantas veces sea necesario.

Sintaxis

```
def NOMBRE(LISTA_DE_PARAMETROS):  
    """DOCSTRING_DE_FUNCION"""  
    SENTENCIAS  
    RETURN [EXPRESION]
```

Advertencia:

Los bloques de `function` deben estar indentado como otros bloques estructuras de control.

Función main()

In [10]:

```
1      """
2          Función suma
3          Entradas: Dos números (enteros o flotantes)
4          Salida: Sumatoria de dos números
5      """
6
7      def suma(num1, num2):
8          suma = num1 + num2
9          return suma
10
11     # Función principal
12     def main():
13         a = 2
14         b = 3
15         print("La suma de "+str(a)+"+"+str(b)+" Es: " + str(suma(a,b)) )
16
17     # Función que detecta el interprete de python
18     if __name__ == "__main__":
19         print("Este es el inicio del programa")
20         main()
```

Este es el inicio del programa

La suma de 2+3 Es: 5

Ejercicio 9 – 15 minutos (Funciones)

1. Escribir una función para cada una de las siguientes operaciones.
 - a) Suma
 - b) Resta
 - c) Multiplicación
 - d) División
 - e) Módulo
2. Escribir una función para que dado un número entero, calcule su factorial.

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times (n - 1) \times n.$$



Ejercicio 10 – 15 minutos

Equipos 4-5 personas

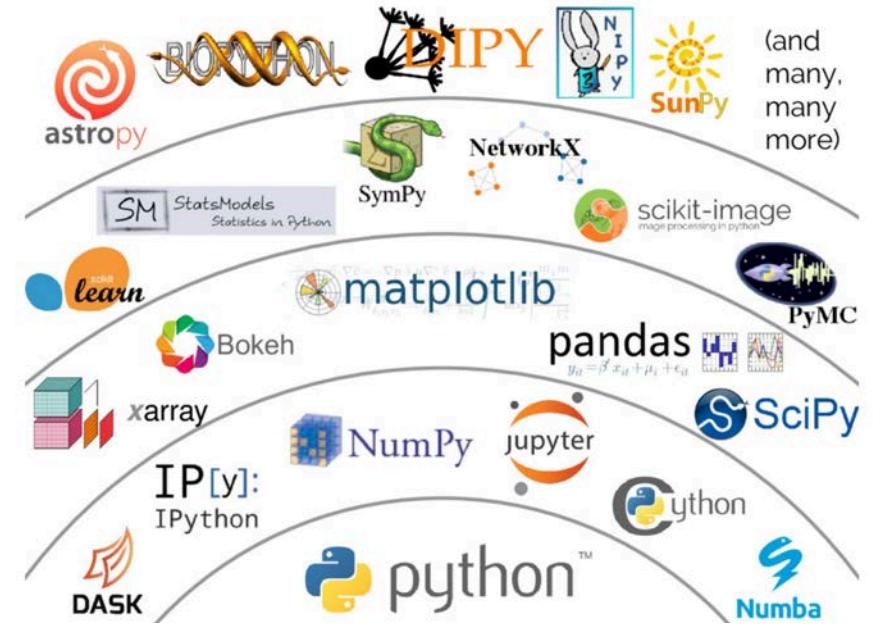
1. Escribir una función para calcular si una cadena es palíndroma.

RECONOCER
RECONOCER



Librerías

- Las librerías (módulos) permiten extender la funcionalidad de Python.
- Un módulo es una porción de programa que realiza una operación específica.
- Una librería contiene un conjunto de módulos.



Librerías - Números Aleatorios

```
[18]: 1 # Importar librería random
2 import random as rn      # Se importa librería con el alias rn
3 rn.seed(20) # Semilla

[19]: 1 print(random.random())      # Número aleatorio x, 0.0 <= x < 1.0
0.9056396761745207

[12]: 1 print(random.random())      # Nú x, 0.0 <= x < 1.0
0.2598274474889769

[13]: 1 rn.uniform(1, 10)          # Número aleatorio x, 1.0 <= x < 10.0
6.7215328264539025

[16]: 1 rn.randint(0, 10)          # Número aleatorio entero x, 0, 10
16: 9

[22]: 1 rn.randrange(0, 100, 2)    # Número entero par x, 0, 100
22: 86

[23]: 1 random.choice('abcdefghij') # Se elige un elemento de forma aleatoria
23: 'b'

[27]: 1 # Mezcla los elementos de forma aleatoria
2 elementos = [1, 2, 3, 4, 5, 6, 7]
3 random.shuffle(elementos)
4 print(elementos)

[4, 1, 2, 5, 3, 7, 6]

[28]: 1 random.sample([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 3) # Escoje tres elementos al azar
28: [9, 8, 7]
```

Python » English 3.8.5 Documentation » The Python Standard Library » Numeric and Mathematical Modules »

random — Generate pseudo-random numbers

Source code: [Lib/random.py](#)

This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range [0.0, 1.0). Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of $2^{19937}-1$. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The functions supplied by this module are actually bound methods of a hidden instance of the `random.Random` class. You can instantiate your own instances of `Random` to get generators that don't share state.

Class `Random` can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the `random()`, `seed()`, `getstate()`, and `setstate()` methods. Optionally, a new generator can supply a `getrandbits()` method — this allows `randrange()` to produce selections over an arbitrarily large range.

The `random` module also provides the `SystemRandom` class which uses the system function `os.urandom()` to generate random numbers from sources provided by the operating system.

Warning: The pseudo-random generators of this module should not be used for security purposes. For security or cryptographic uses, see the `secrets` module.

See also: M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Transactions on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30 1998.

[Complementary-Multiply-with-Carry recipe](#) for a compatible alternative random number generator with a long period and comparatively simple update operations.

Librerías – Funciones Matemáticas

```
[1]: import math  
  
math.pi  
  
[1]: 3.141592653589793  
  
[2]: math.sin(90)  
  
[2]: 0.8939966636005579  
  
[4]: math.sqrt(9)  
  
[4]: 3.0  
  
[5]: math.ceil(2.5)  
  
[5]: 3  
  
[6]: math.floor(2.9)  
  
[6]: 2  
  
[8]: math.cos(90)  
  
[8]: -0.4480736161291701
```

Python » Spanish 3.8.5 Documentation » La Biblioteca Estándar de Python » Numeric and Mathematical Modules »

Tabla de contenido

- [math — Mathematical functions](#)
 - Number-theoretic and representation functions
 - Power and logarithmic functions
 - Trigonometric functions
 - Angular conversion
 - Hyperbolic functions
 - Special functions
 - Constants

Tema anterior

[numbers — Clase base abstracta numérica](#)

Próximo tema

[cmath — Mathematical functions for complex numbers](#)

Esta página

[Reporta un Bug](#)
[Mostrar el código](#)

math — Mathematical functions

This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the [cmath](#) module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

Number-theoretic and representation functions

`math.ceil(x)`

Return the ceiling of x , the smallest integer greater than or equal to x . If x is not a float, delegates to $x.\underline{\text{ceil}}()$, which should return an [Integral](#) value.

`math.comb(n, k)`

Return the number of ways to choose k items from n items without repetition and without order.
Evaluates to $n! / (k! * (n - k)!)$ when $k \leq n$ and evaluates to zero when $k > n$.

Also called the binomial coefficient because it is equivalent to the coefficient of k -th term in polynomial expansion of the expression $(1 + x)^n$.

Raises [TypeError](#) if either of the arguments are not integers. Raises [ValueError](#) if either of the arguments are negative.

Nuevo en la versión 3.8.

`math.copysign(x, y)`

Return a float with the magnitude (absolute value) of x but the sign of y . On platforms that support signed zeros, `copysign(1.0, -0.0)` returns `-1.0`.

`math.fabs(x)`

Return the absolute value of x .

`math.factorial(x)`

Return x factorial as an integer. Raises [ValueError](#) if x is not integral or is negative.

`math.floor(x)`

Return the floor of x , the largest integer less than or equal to x . If x is not a float, delegates to $x.\underline{\text{floor}}()$, which should return an [Integral](#) value.

Ejercicio 11

15 minutos

- Se deberá generar un total de 10,000 números aleatorios en un rango del [1, 100].
 1. Se deberá generar una lista en forma de histograma con 10 clases, se deberá contabilizar la totalidad de elementos en cada clase.

