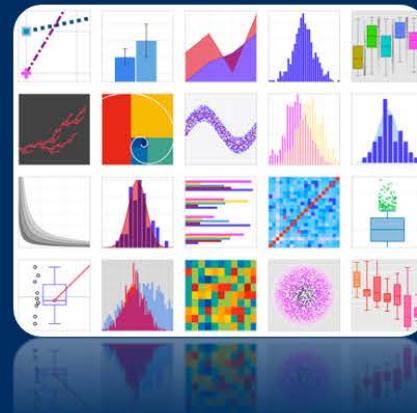


Introducción al Análisis y Visualización de Datos en Python

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)

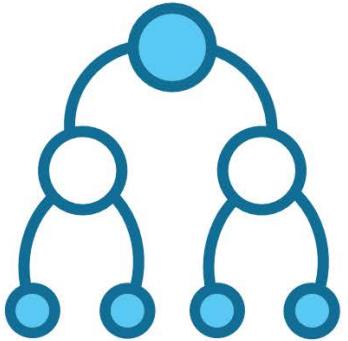
Introducción y antecedentes
Entorno de programación
Variables
Operadores
Break (15 minutos)
Instrucciones de control
Funciones
Librerías



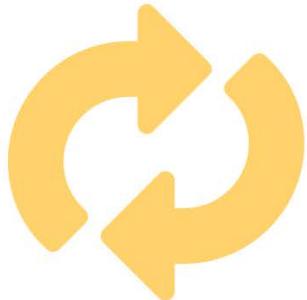
2. Proyecto (Equipos) (30 ~45 minutos)



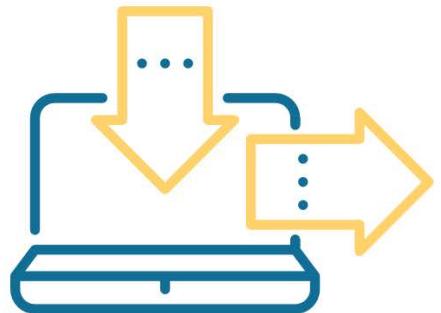
Sequence



Conditional
Statements



Loops



Input / Output



Functions

Conceptos Clave

¿Qué es un lenguaje de programación?

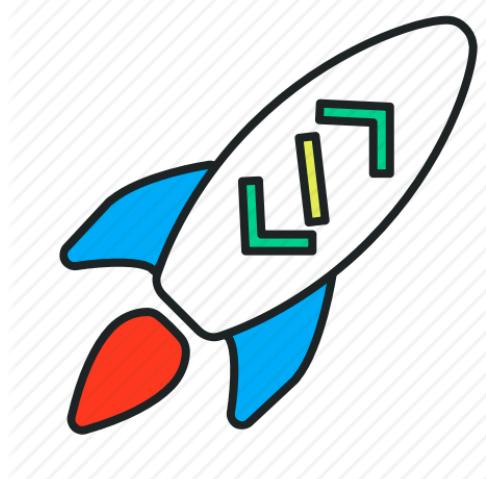
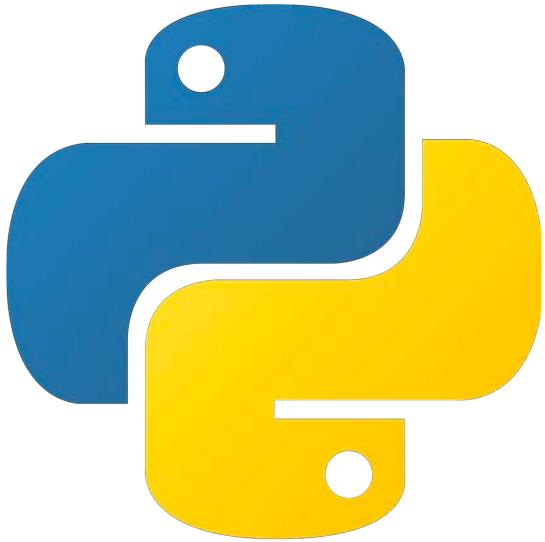
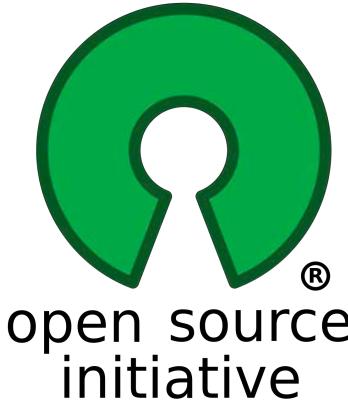
Un lenguaje de programación es un **lenguaje formal** que comprende un conjunto de **instrucciones**, las cuales producen varios tipos de **resultados**.

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



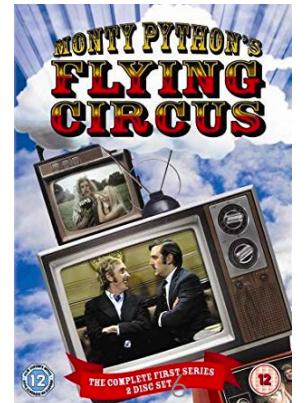
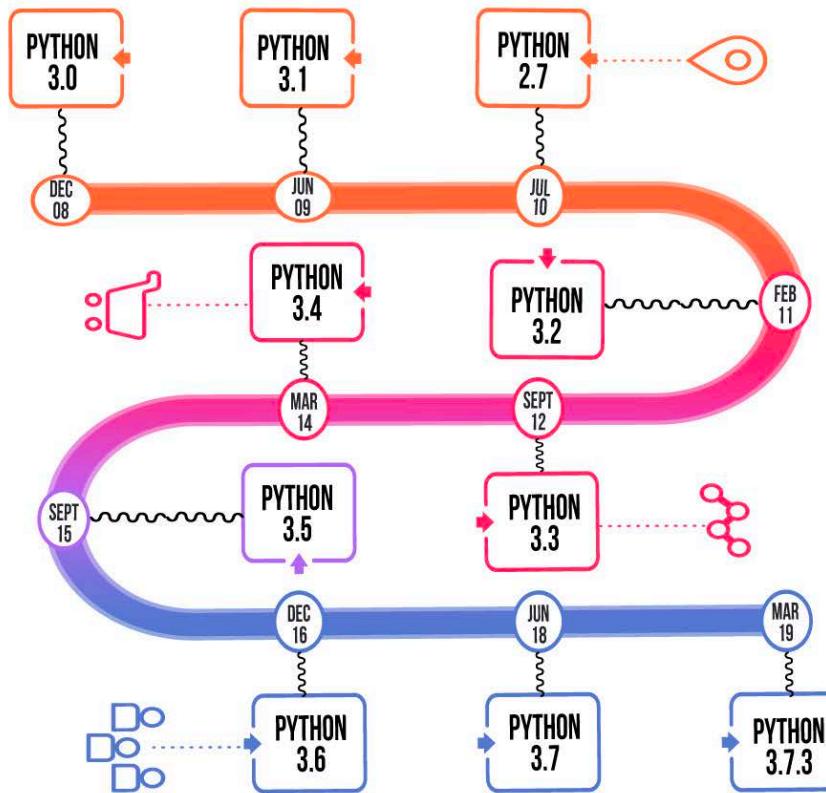
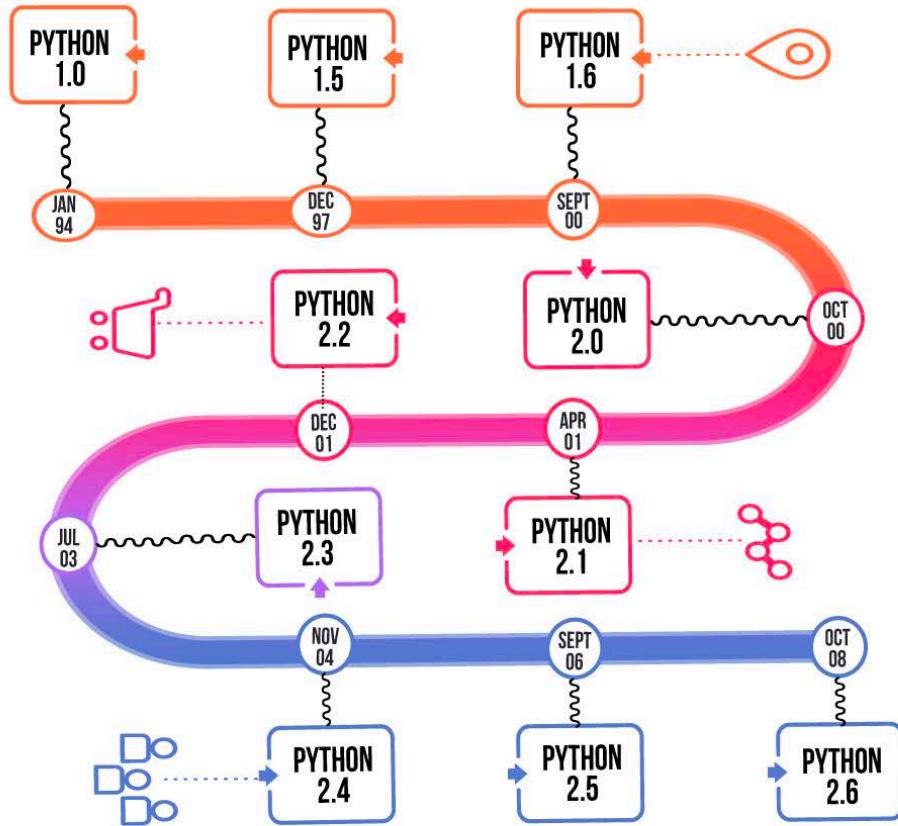
¿Qué es Python?

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](#)

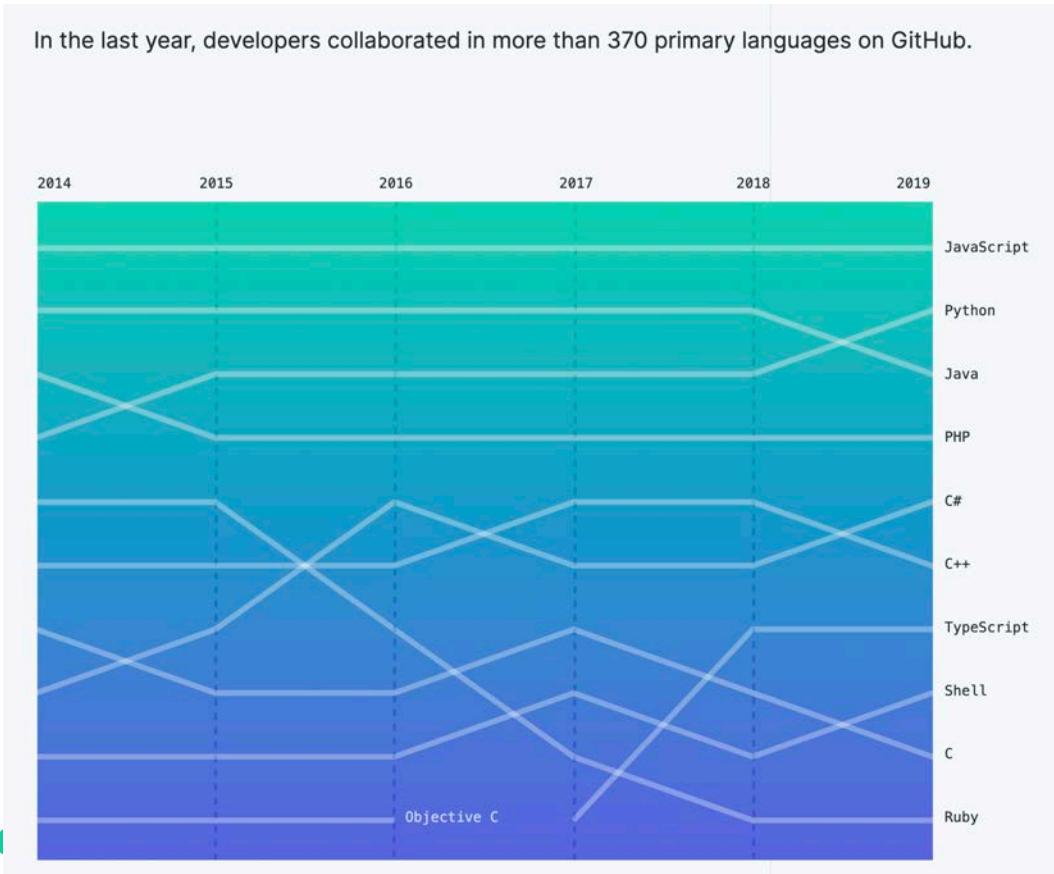
Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#),
[Other](#)

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

Looking for Python 2.7? See below for specific releases

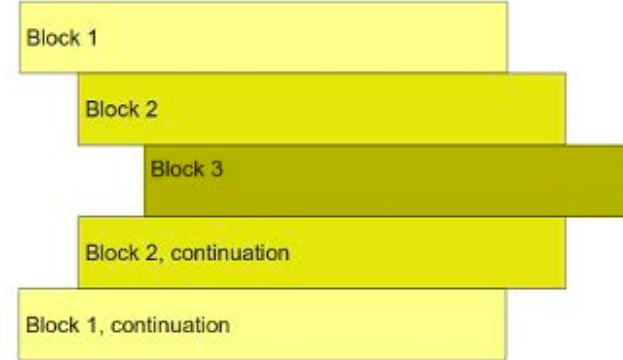


Python es uno de los lenguajes de programación con mayor adopción a nivel mundial.



Características de Python

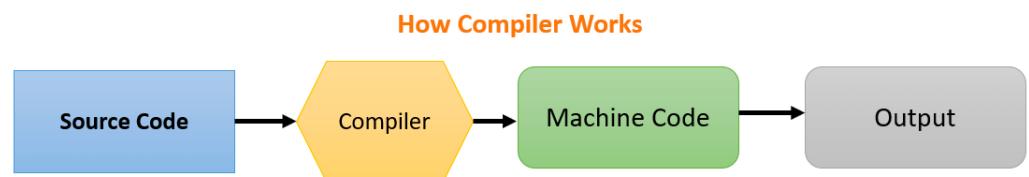
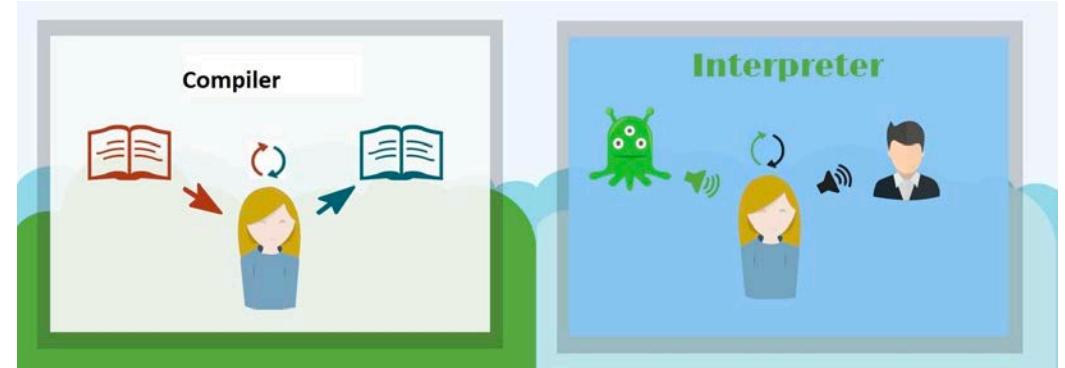
1. Indentación obligatoria
2. Interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)



```
# 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

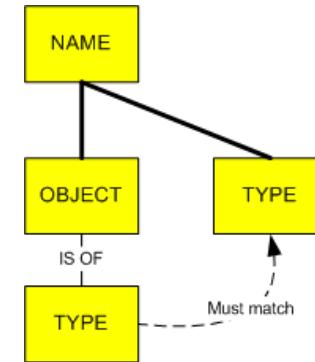
1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Excepciones
5. Librerías (Funcionalidad extendida)



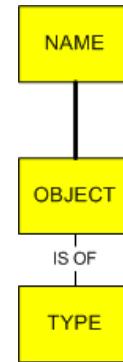
Características de Python

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

Estática



Dinámica

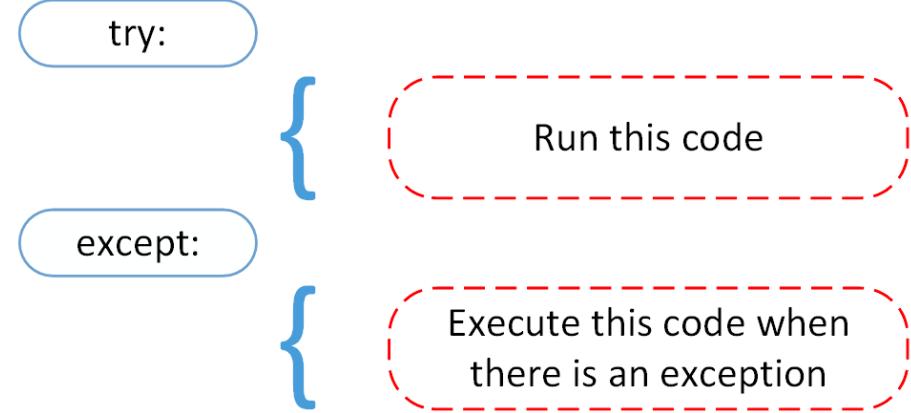


```
my_var = 12  
print(type(my_var))
```

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

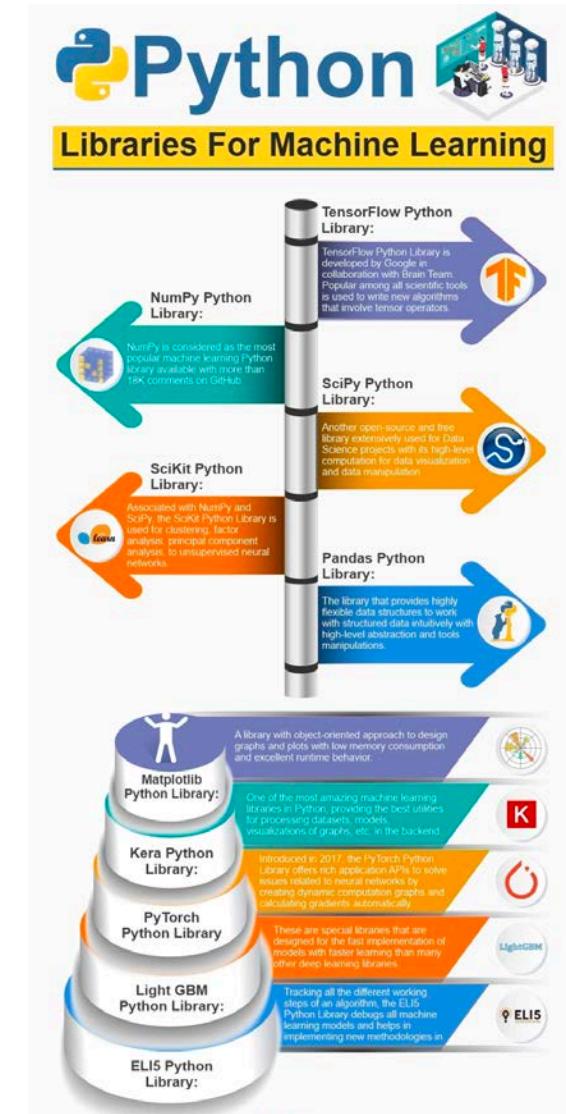
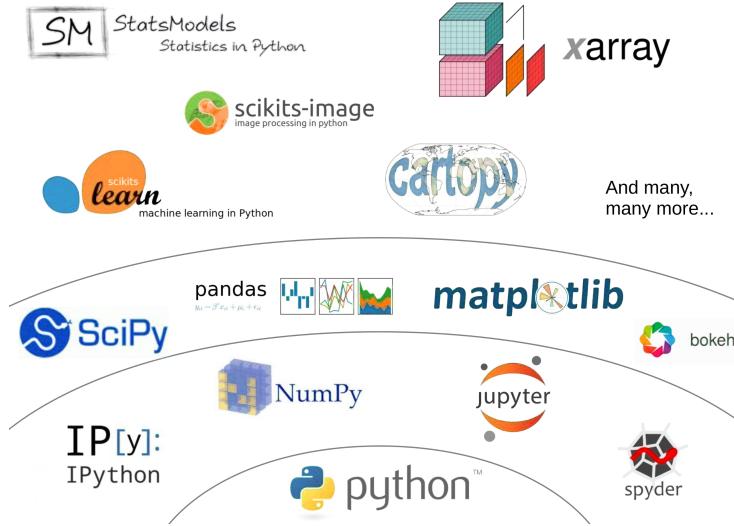
Características de Python

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

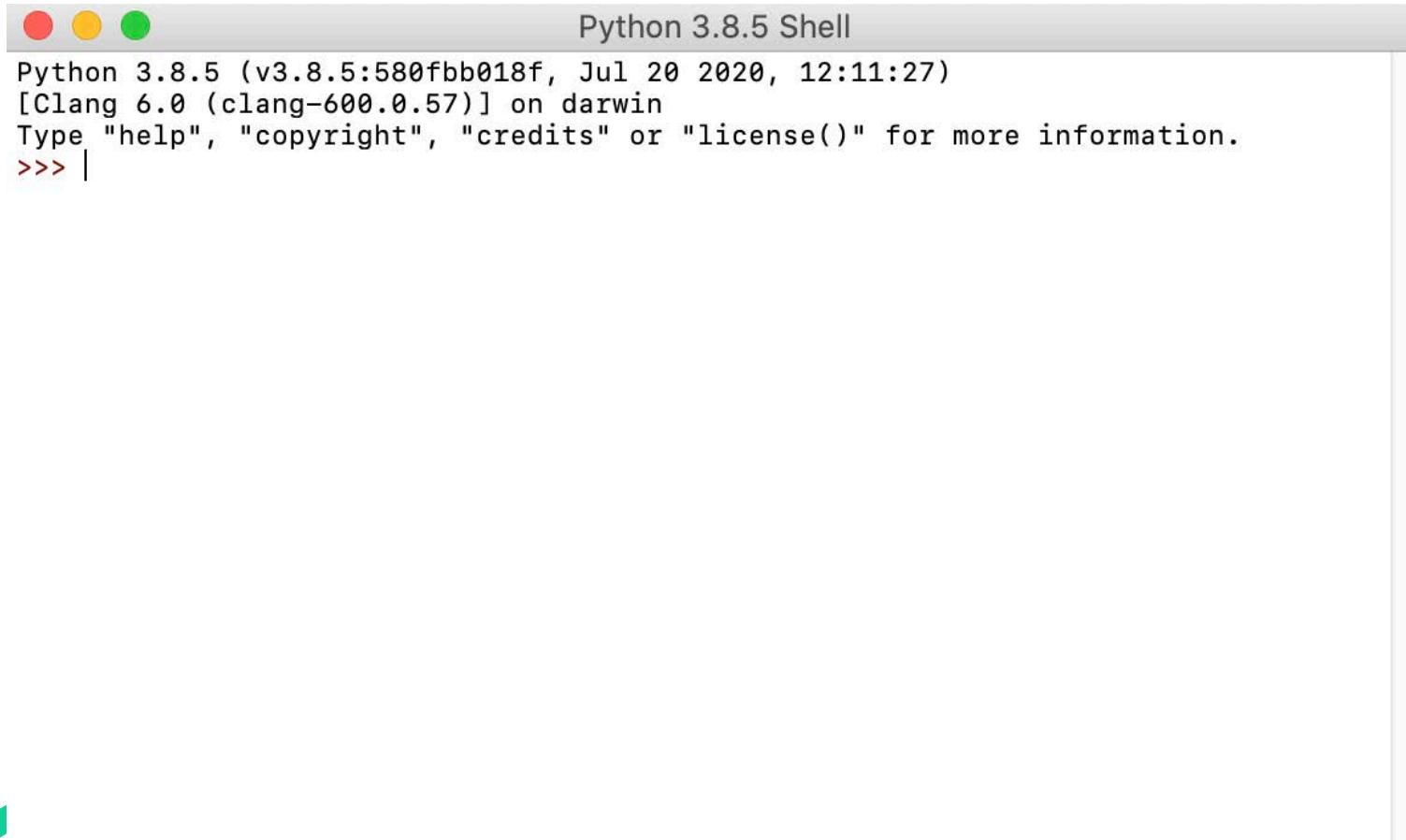


Características de Python

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



Entorno de programación – Consola



```
Python 3.8.5 Shell
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, there's a menu bar with "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Below the menu is a toolbar with icons for file operations (New, Open, Save, etc.), cell navigation (Up, Down), and execution (Run, Cell Kernel Stop, Cell Kernel Restart, Code). A status bar at the top right indicates "Last Checkpoint: 15 minutes ago". The main area contains a code cell with the title "# Introducción a Python - Día 1". The cell's input is:

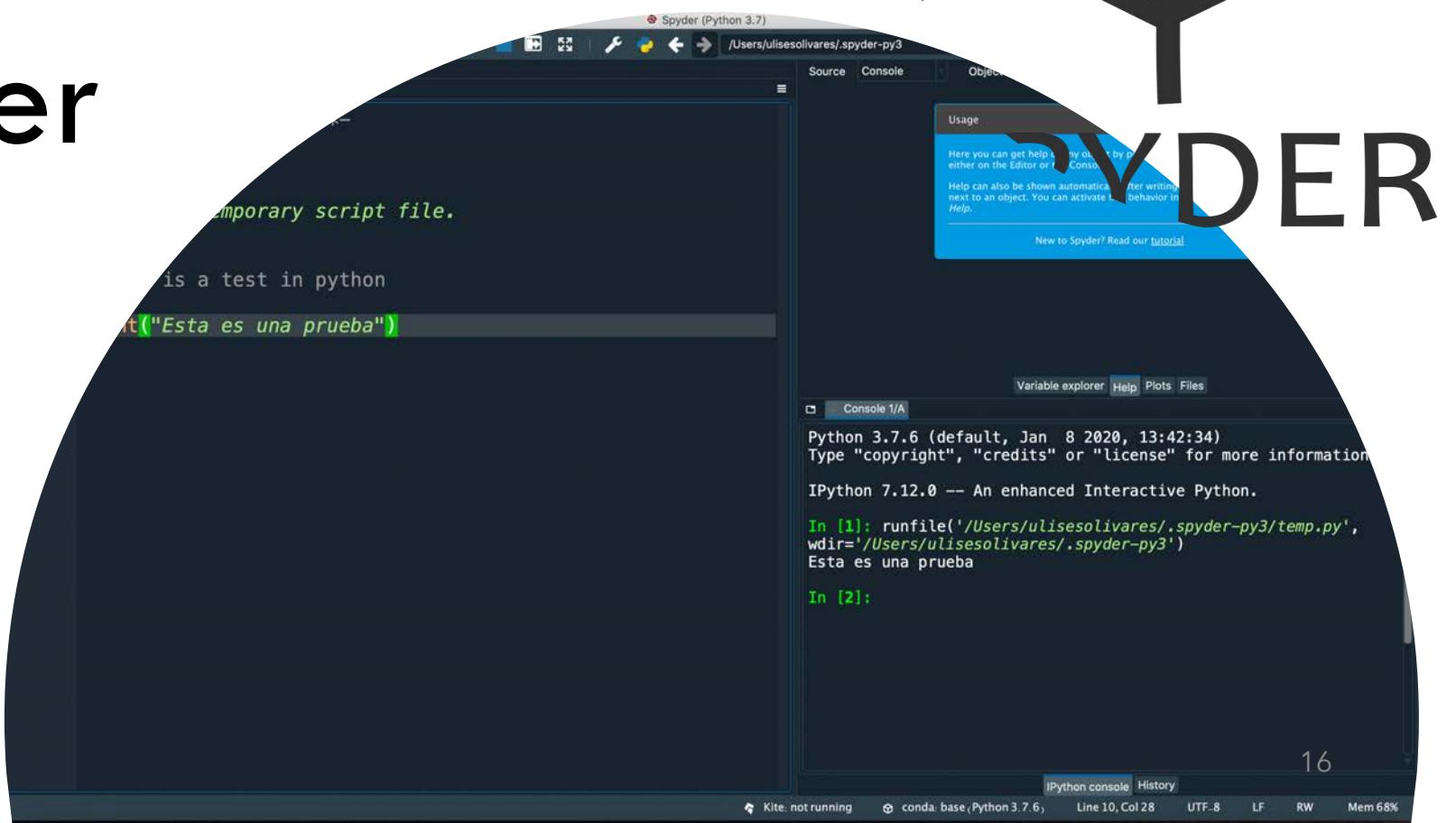
```
In [2]: # Impresión de prueba  
print("Hola mundo, esta es una prueba")
```

The cell's output is:

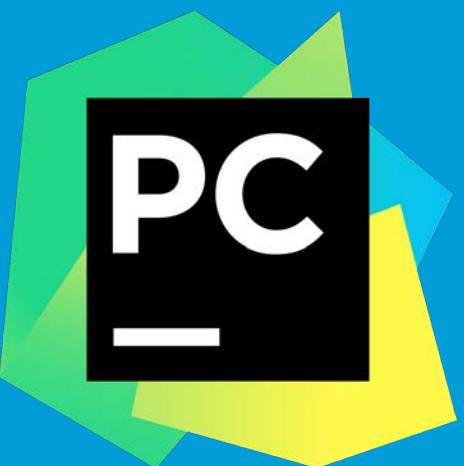
Hola mundo, esta es una prueba

At the bottom, there's another input cell labeled "In []:".

IDE - Spyder



IDE – Pycharm



A screenshot of the PyCharm IDE interface. The window title is "main - main.py". The left sidebar shows a "Project" view with a "main" folder containing "main.py", and "External Libraries" and "Scratches and Consoles" sections. The main code editor area contains the following Python code:

```
print("Hola")
#print("Hola2")
```

The "Run" tool bar at the bottom shows the command: "/Users/ulisesolivares2/serial_acc/venv/bin/python". The status bar at the bottom right indicates the file is "main.py", encoding is "UTF-8", and the Python version is "2.7".

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

>>>

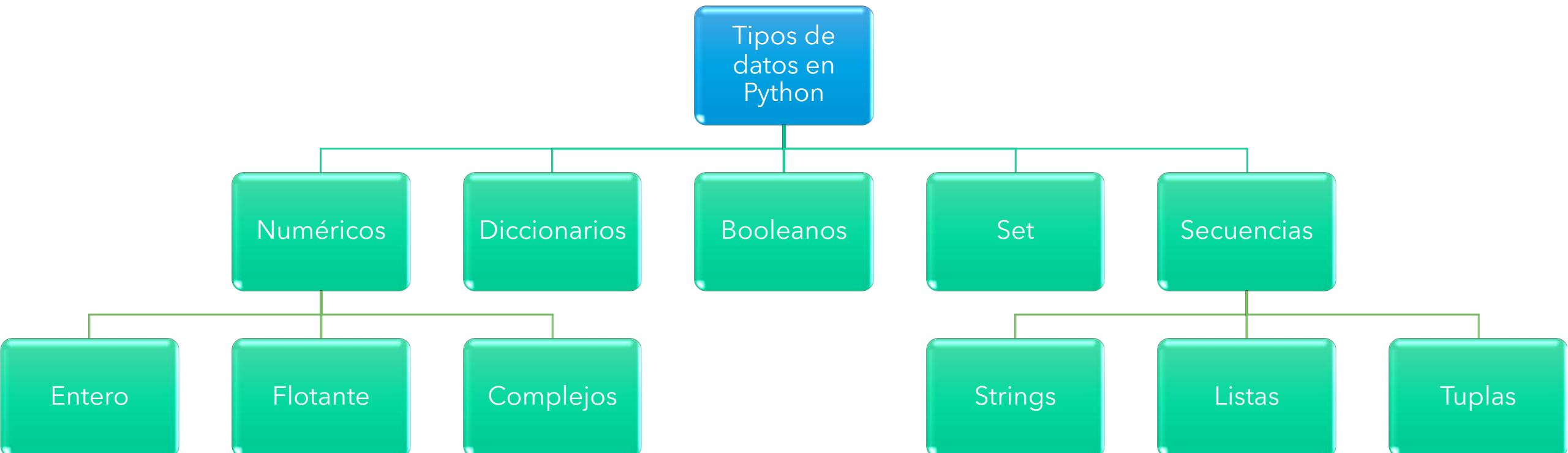
```
>>> 'My age is ' + str(42)
'My age is 42'
```



```
(base) Ulises-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)`



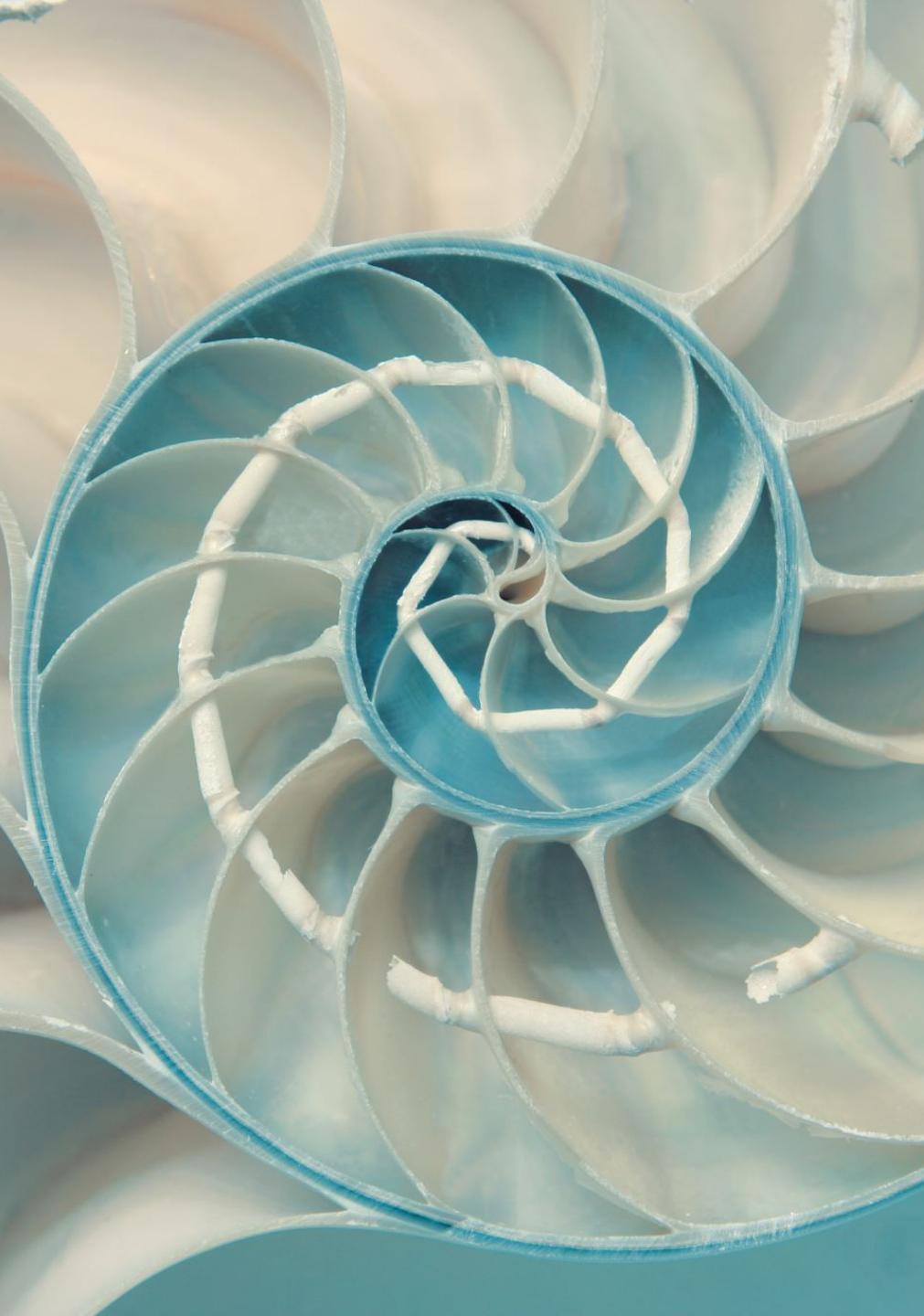
Ejercicio 1- (8 minutos)

- 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
Índice :	0	1	2	3	4	5
Índice inverso :	-6	-5	-4	-3	-2	-1

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 deberán imprimir las iniciales del usuario. Ej. Juan Pérez López (JPL)
 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()`

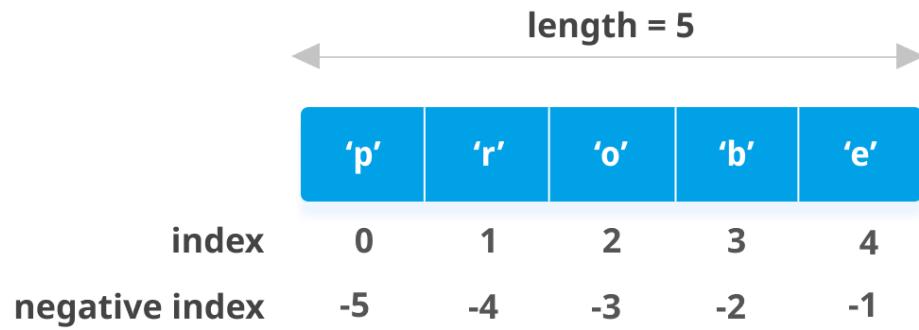
```
>>> vocales = list('aeiou')  
>>> vocales  
['a', 'e', 'i', 'o', 'u']
```

- 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

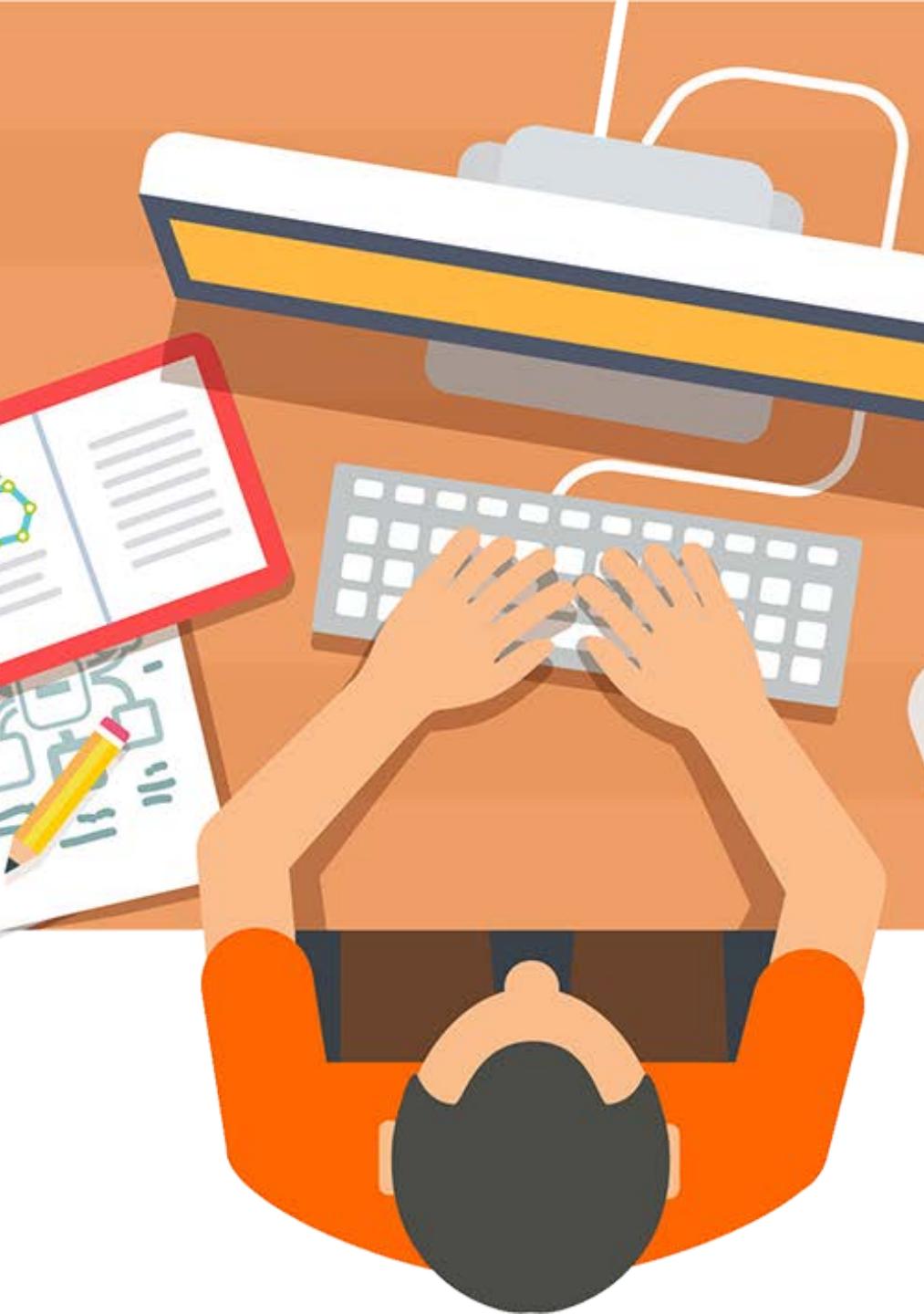
```
# Identidad del objeto numeros
>>> id(numeros)
4475439216
```

```
# 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><</code> .
<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,]
```

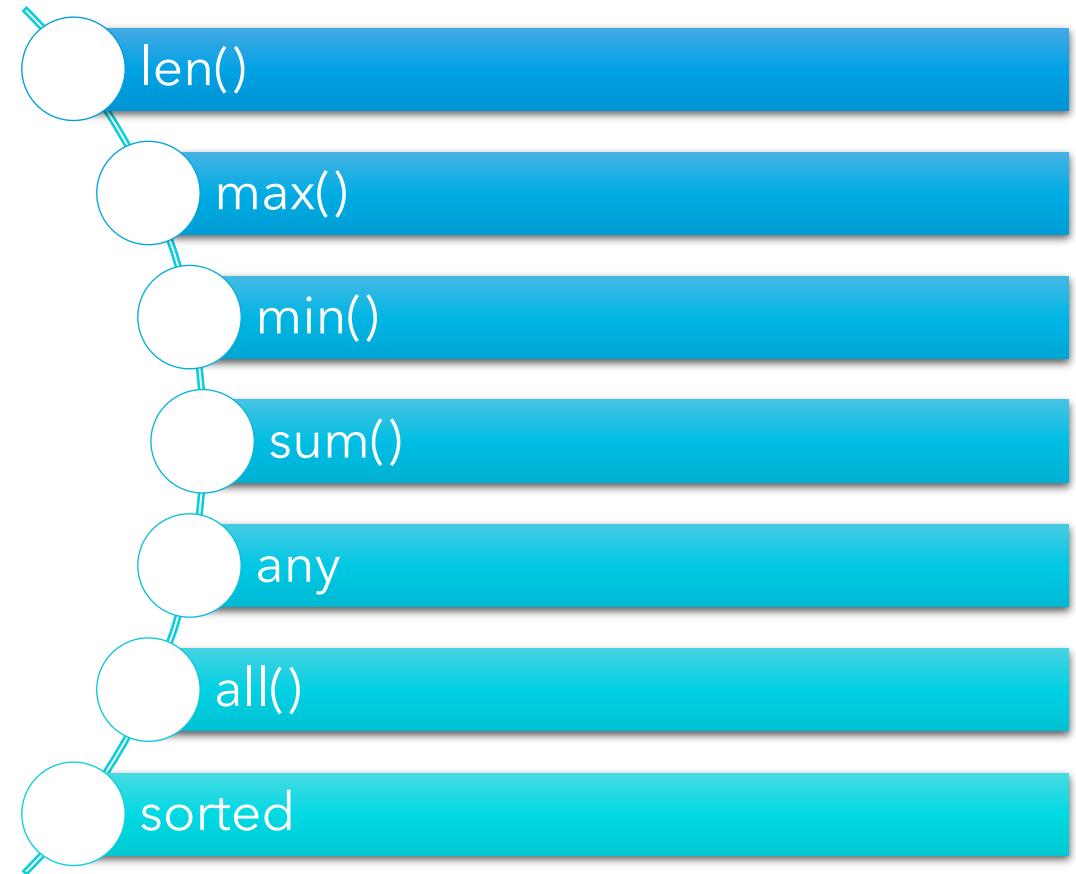
1. Separar los elementos en dos listas:
 - a. Lista numérica
 - b. Lista de caracteres.
2. Ordenar la lista numérica en orden ascendente.
3. Ordenar la lista de strings en orden alfabético (A-Z).

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)
```

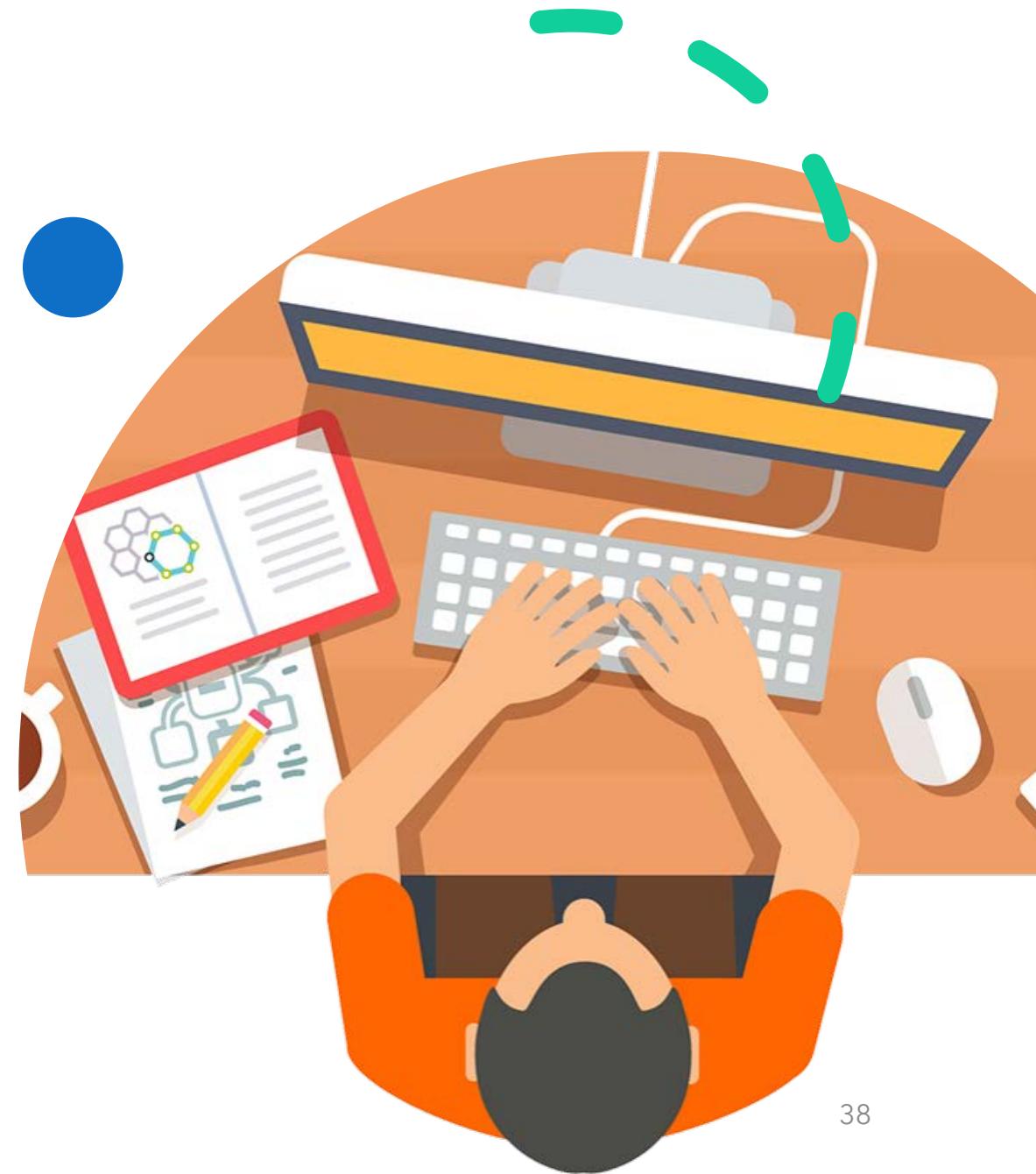
Count

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'>
```

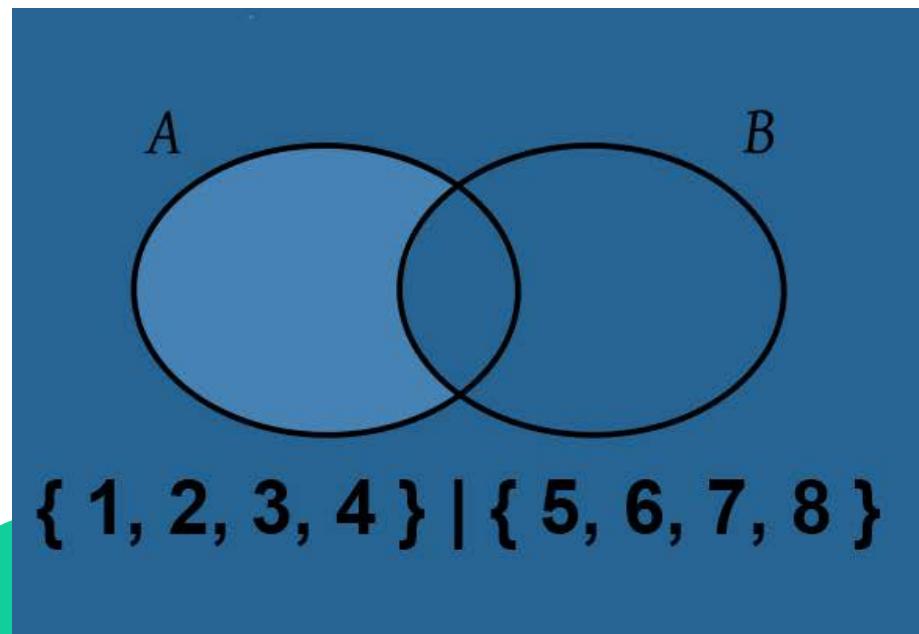


Booleanos en Python

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.

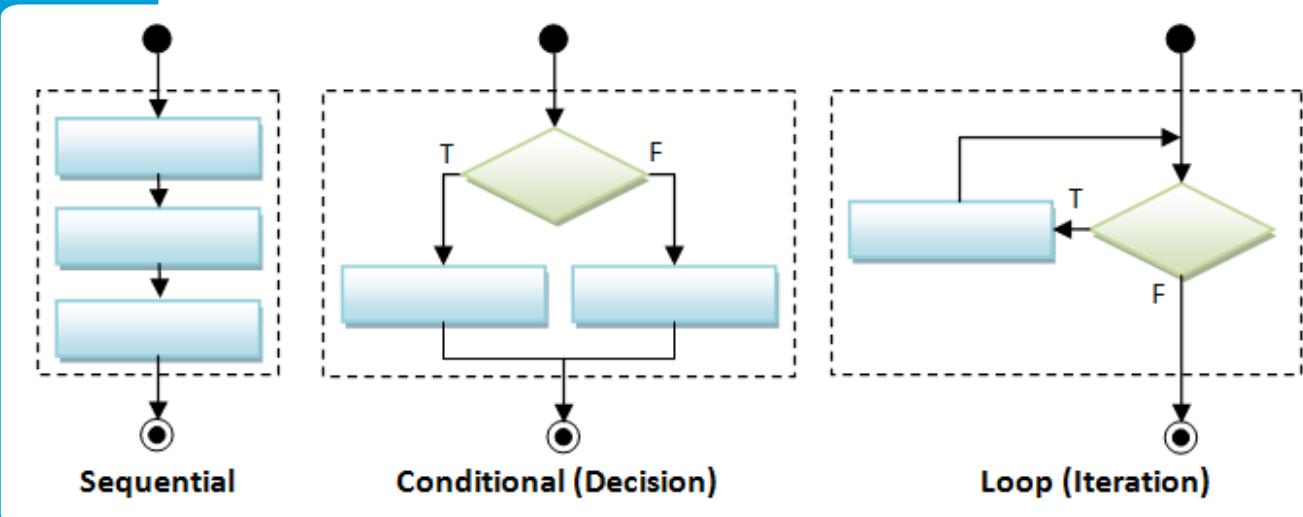


Instrucciones de Control en Python



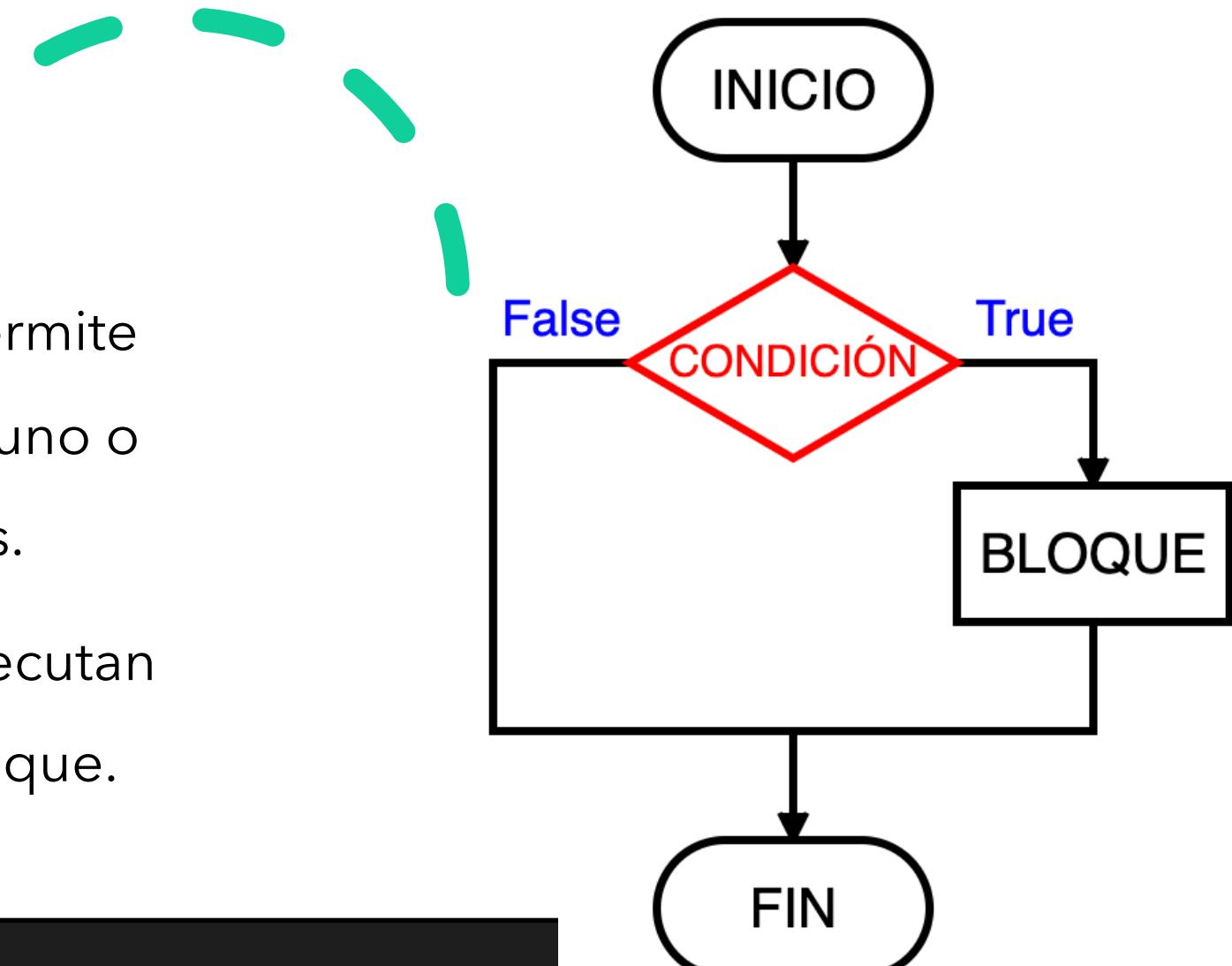
```
34  
35     self.debug = debug  
36     self.logger = logger  
37     if path:  
38         self._path = path  
39     else:  
40         self._path = None  
41  
42     @classmethod  
43     def from_set(cls, settings, path=None):  
44         debug = settings.get('logger.debug', False)  
45         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.

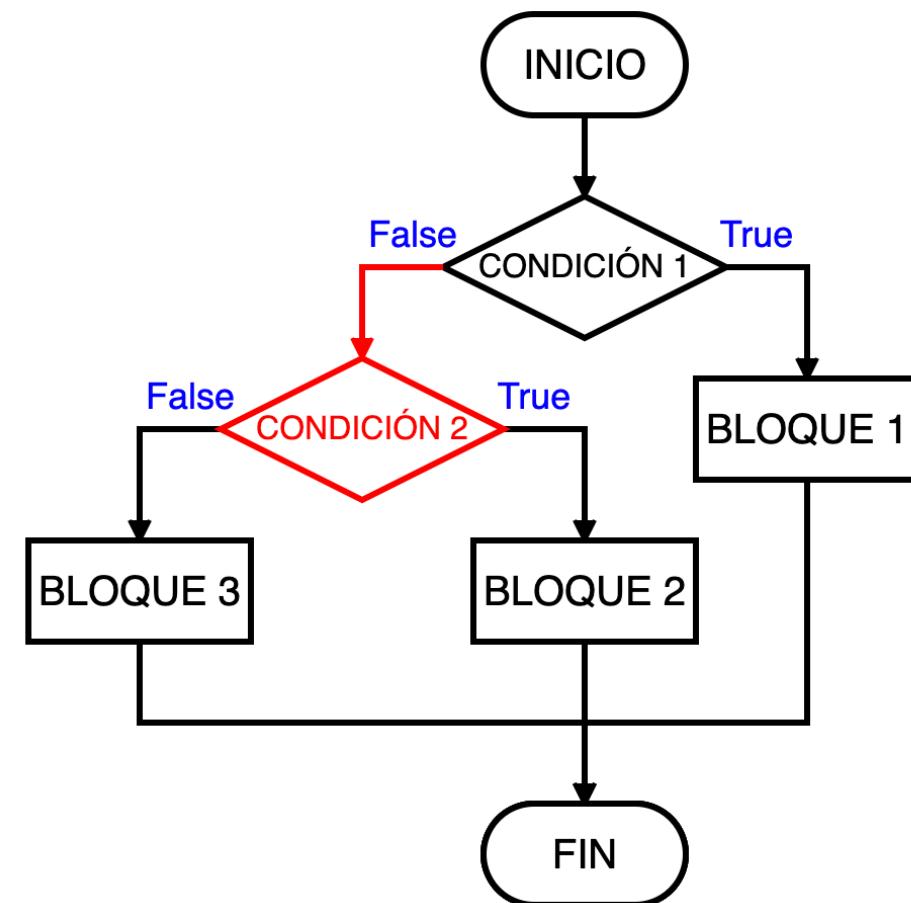


```
if condición:  
    aquí van las órdenes que se ejecutan si la condición es cierta  
    y que pueden ocupar varias líneas
```

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

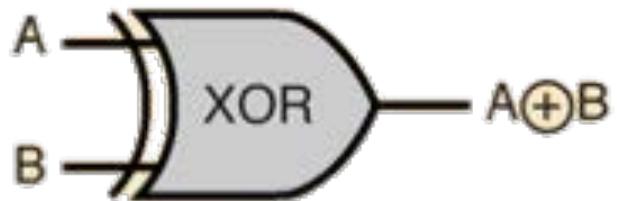
```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

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

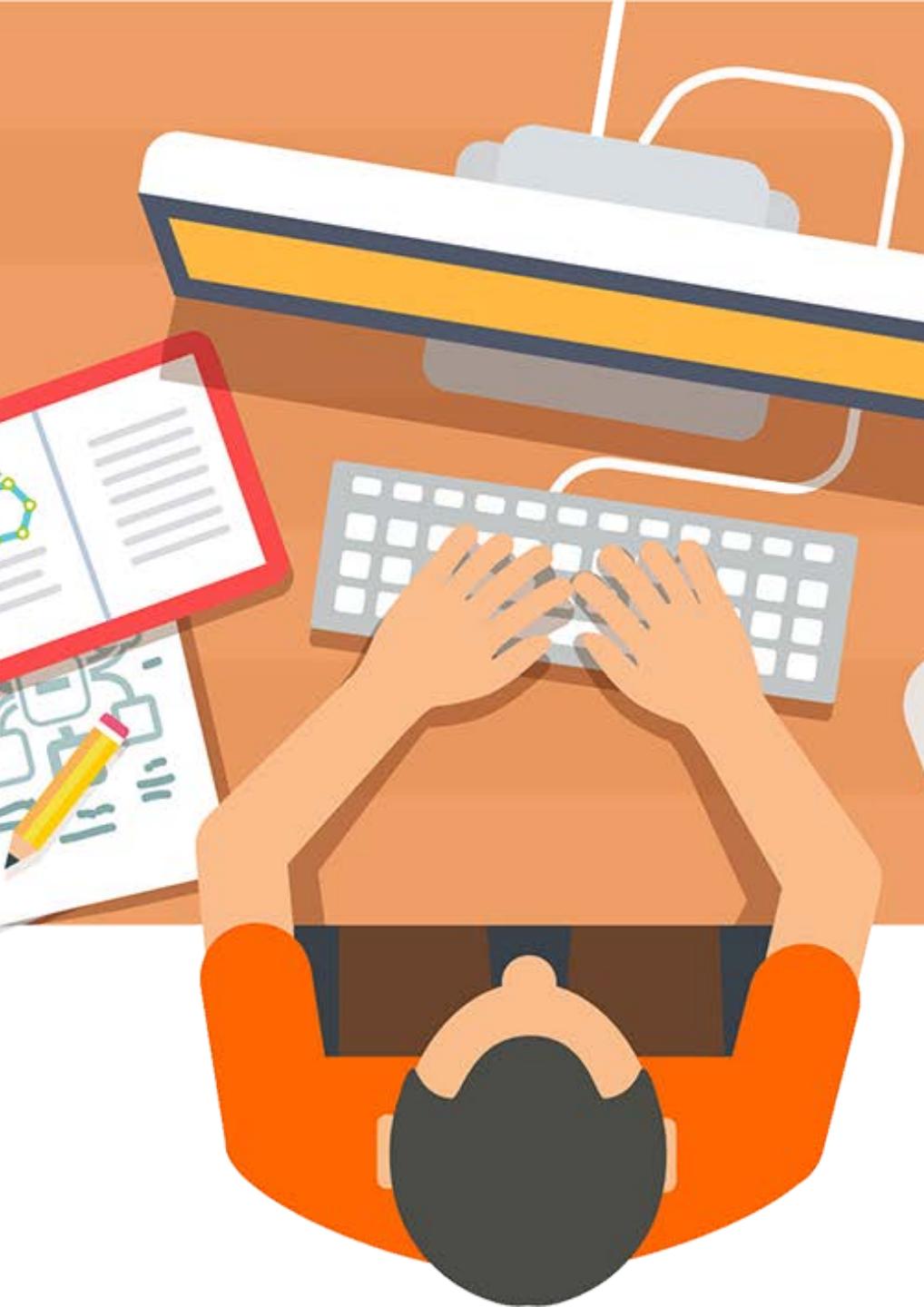


Operadores Lógicos

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

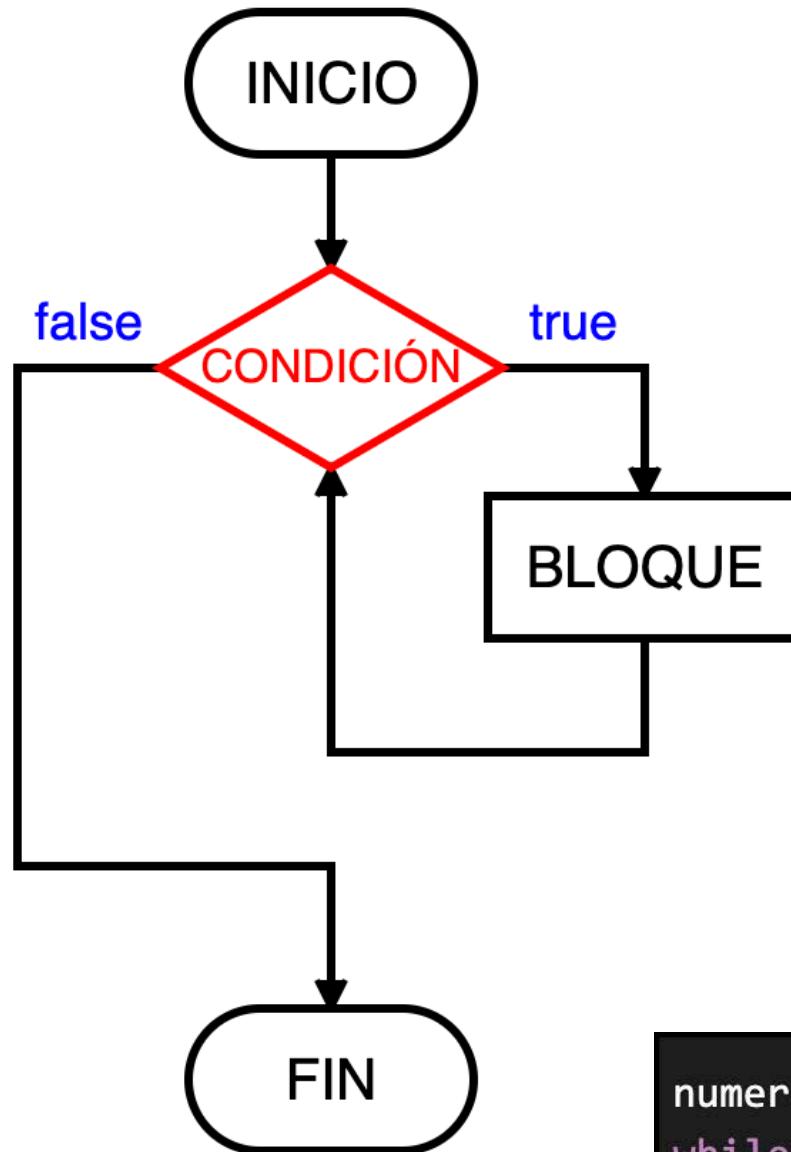


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



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

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

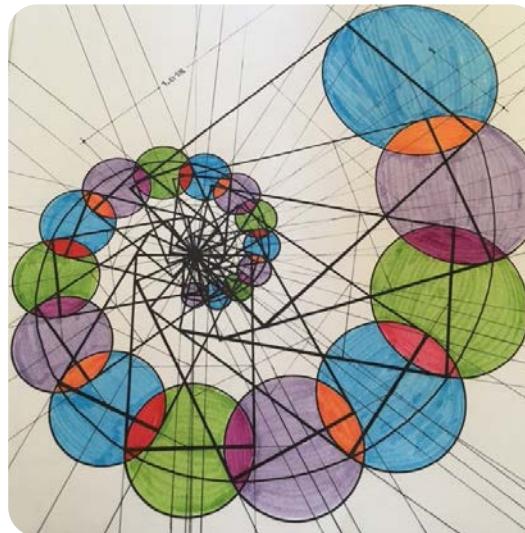
$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

...



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.