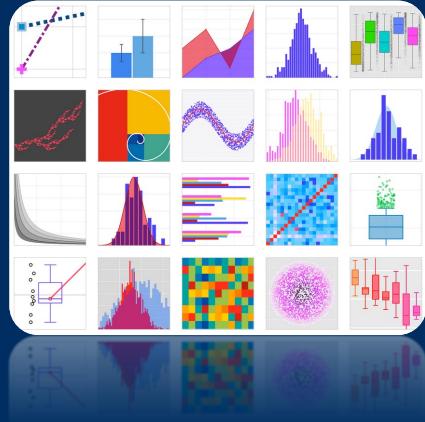


Introducción al aprendizaje de máquinas en Python

Día 1 – Introducción a Python y Dataframes



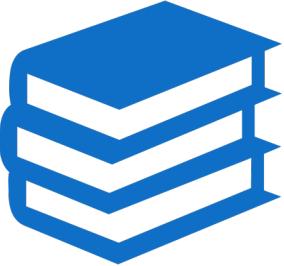
Presentan:
Dr. Ulises Olivares Pinto
Dr. Jesús Emmanuel Solís Pérez
Walter André Rosales Reyes
Escuela Nacional de Estudios Superiores Unidad Juriquila



UNAM
La Universidad
de la Nación



Contenido



1. Introducción a Python **(~3 horas 45 minutos)**

Introducción y antecedentes

Entorno de programación

Variables

Operadores

Break (15 minutos)

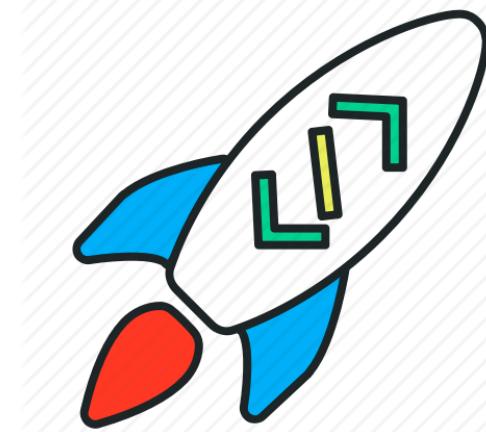
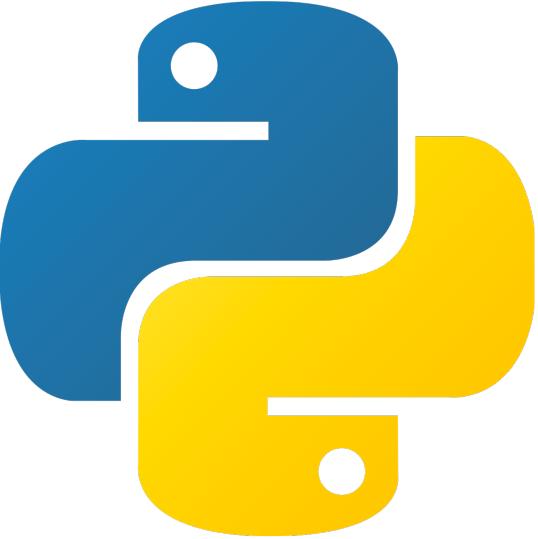
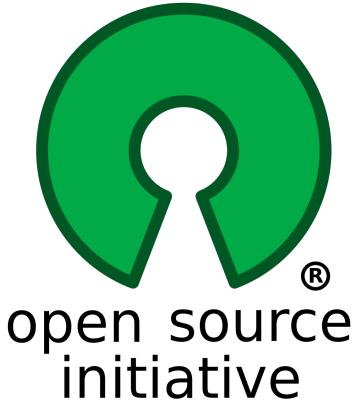
Instrucciones de control

Funciones

Librerías

¿Qué es Python?

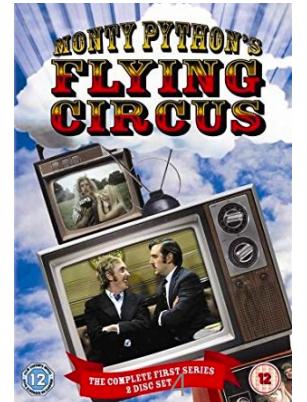
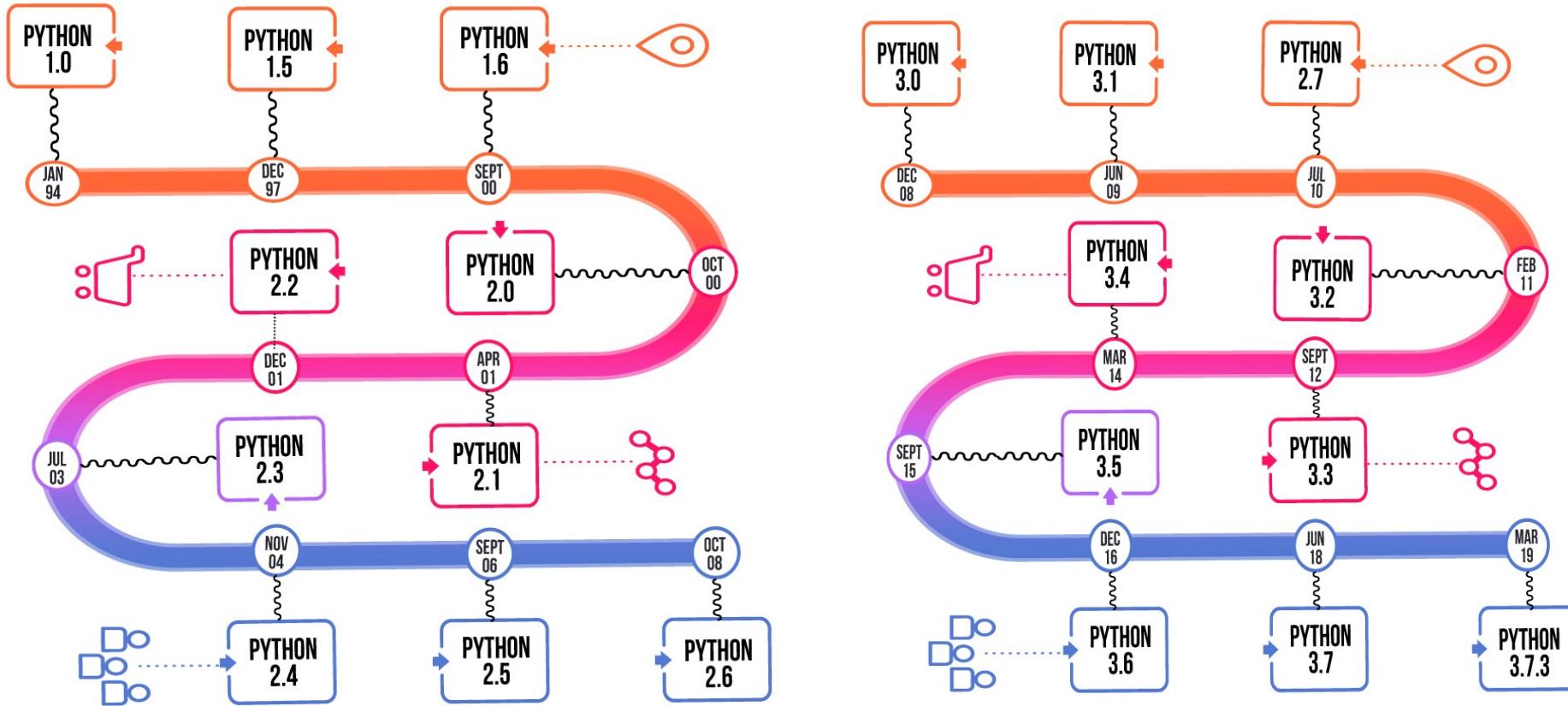
1. Lenguaje Orientado a Objetos
2. Independiente de plataforma
3. Centrado en el tiempo de desarrollo
4. Sintaxis simple y fácil
5. Lenguaje de alto nivel
6. Gestión automática de la memoria
7. ¡Es gratis (código abierto)!



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 macOS

[Download Python 3.10.2](#)

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

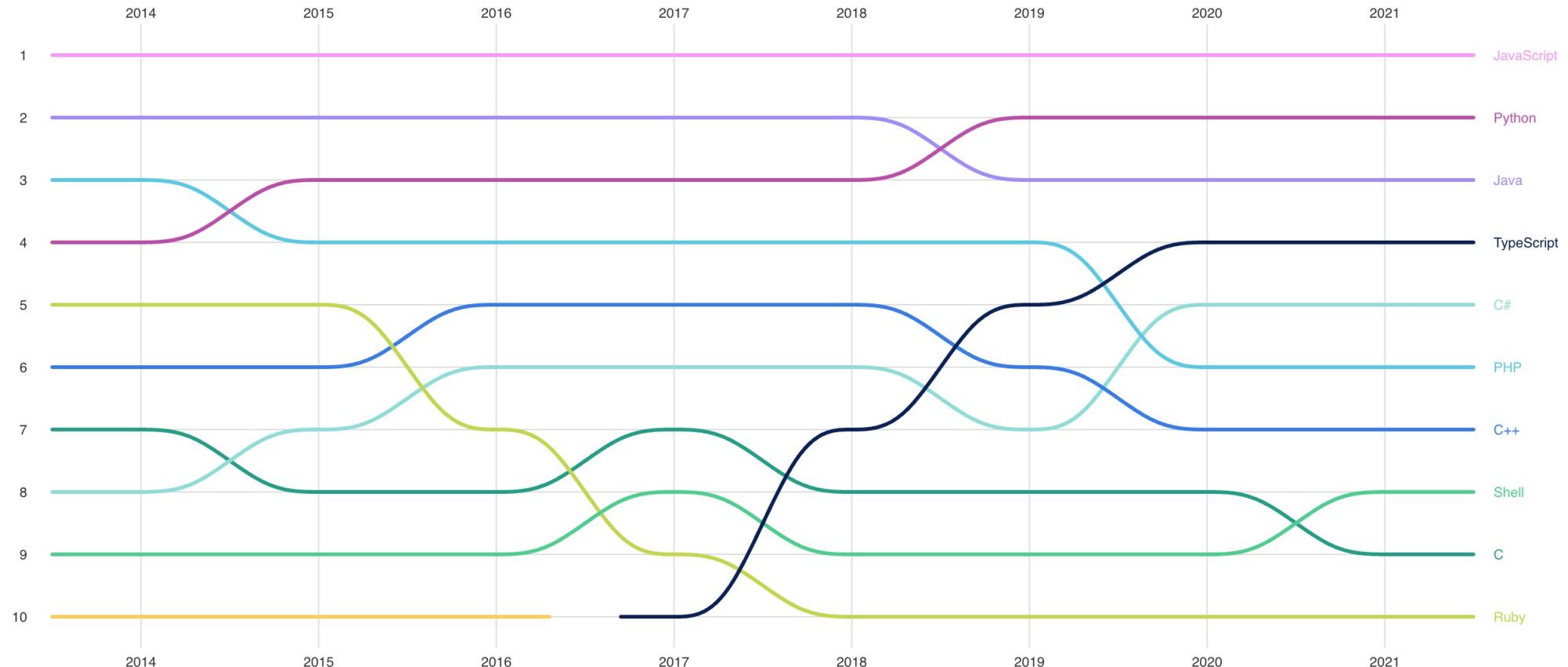
Want to help test development versions of Python? [Prereleases](#),
[Docker images](#)

Looking for Python 2.7? See below for specific releases



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

Top languages over the years

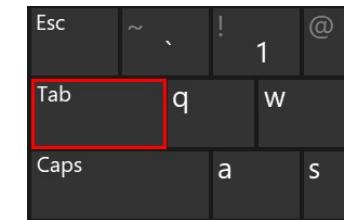
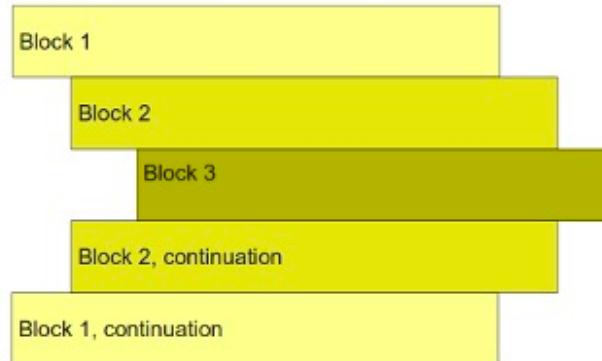


Python es uno de los lenguajes de programación con mayor adopción a nivel mundial.



Características de Python

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

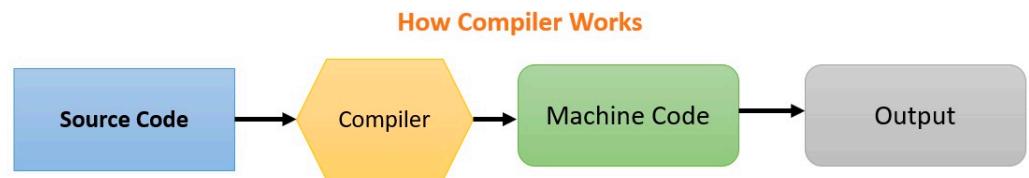
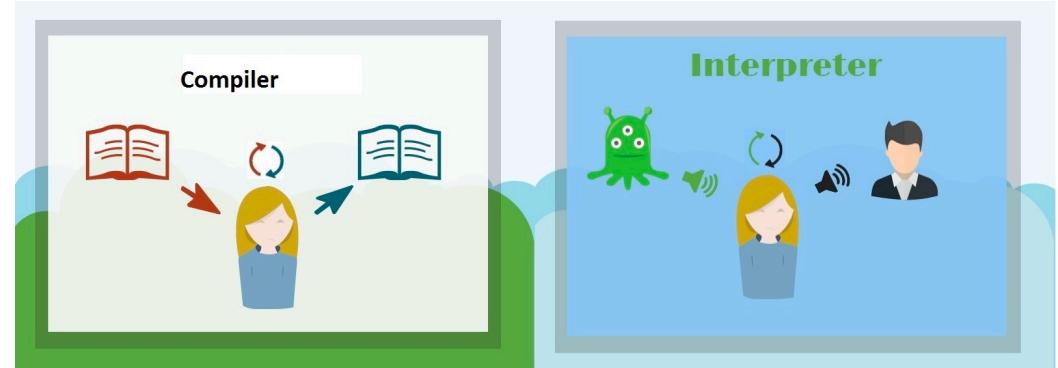


```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
```



Características de Python

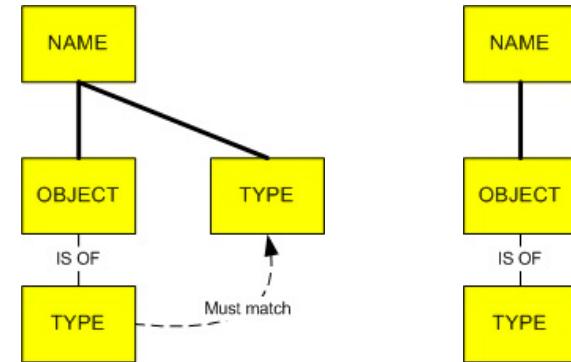
1. Indentación obligatoria
2. Lenguaje interpretado
3. Tipificación dinámica
4. Librerías (Funcionalidad extendida)



Características de Python

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

Estática Dinámica

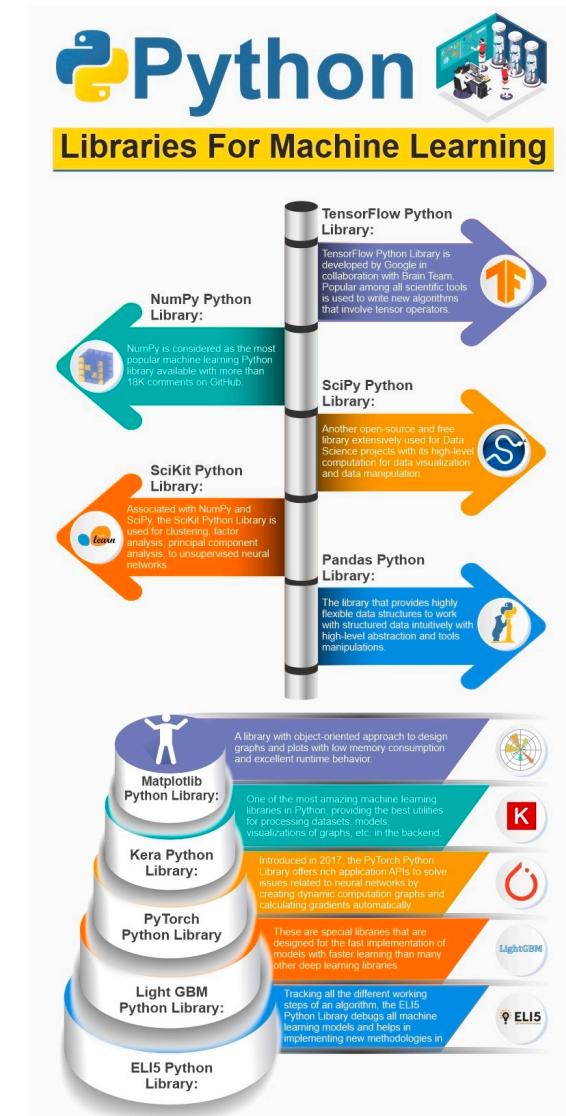
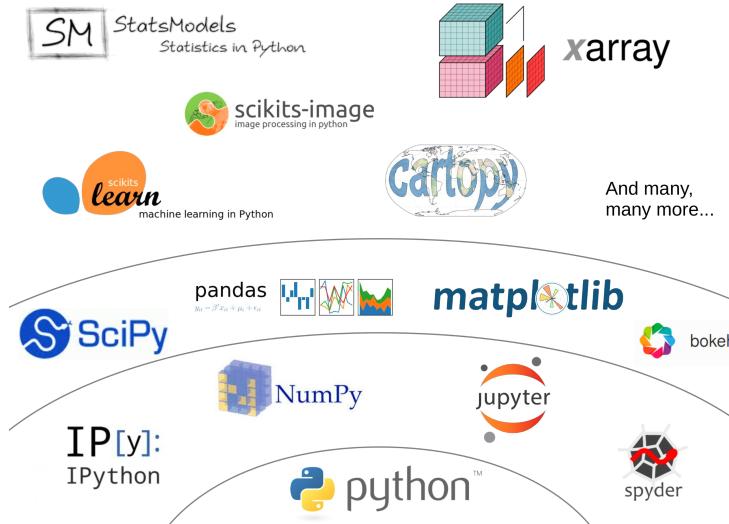


```
my_var = 12  
print(type(my_var))
```

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

Características de Python

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

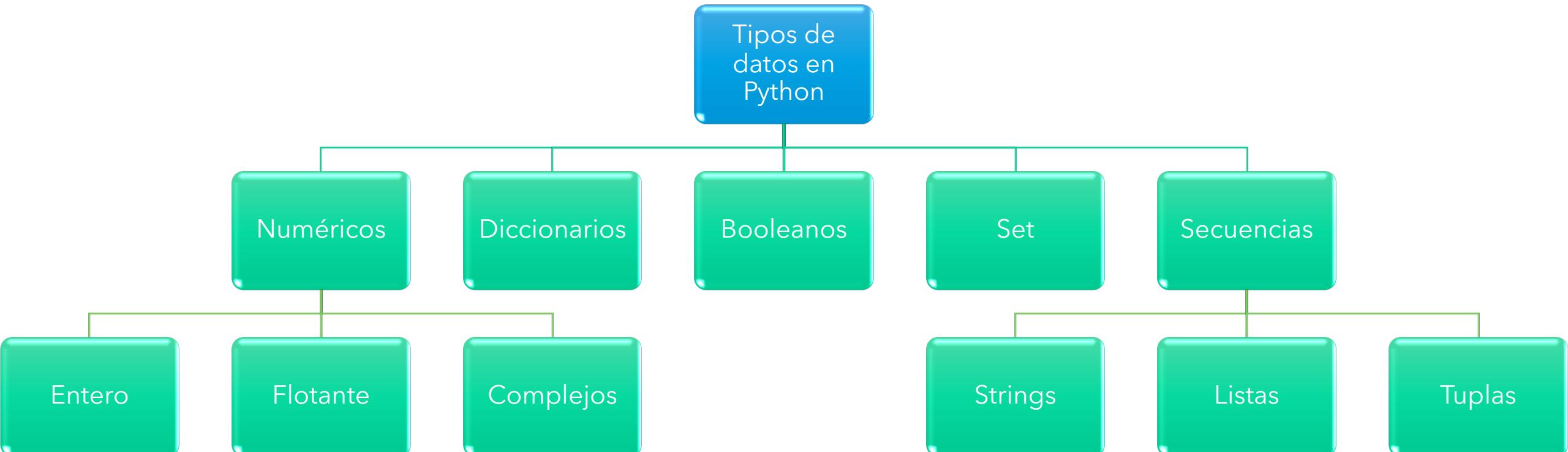


Google Colaboratory

<https://colab.research.google.com/>



Variables, Tipos y Operadores



Numéricos

Enteros

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

Flotantes

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

Complejos

- $c1 = (1, 2j)$

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



Operadores Aritméticos

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

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.



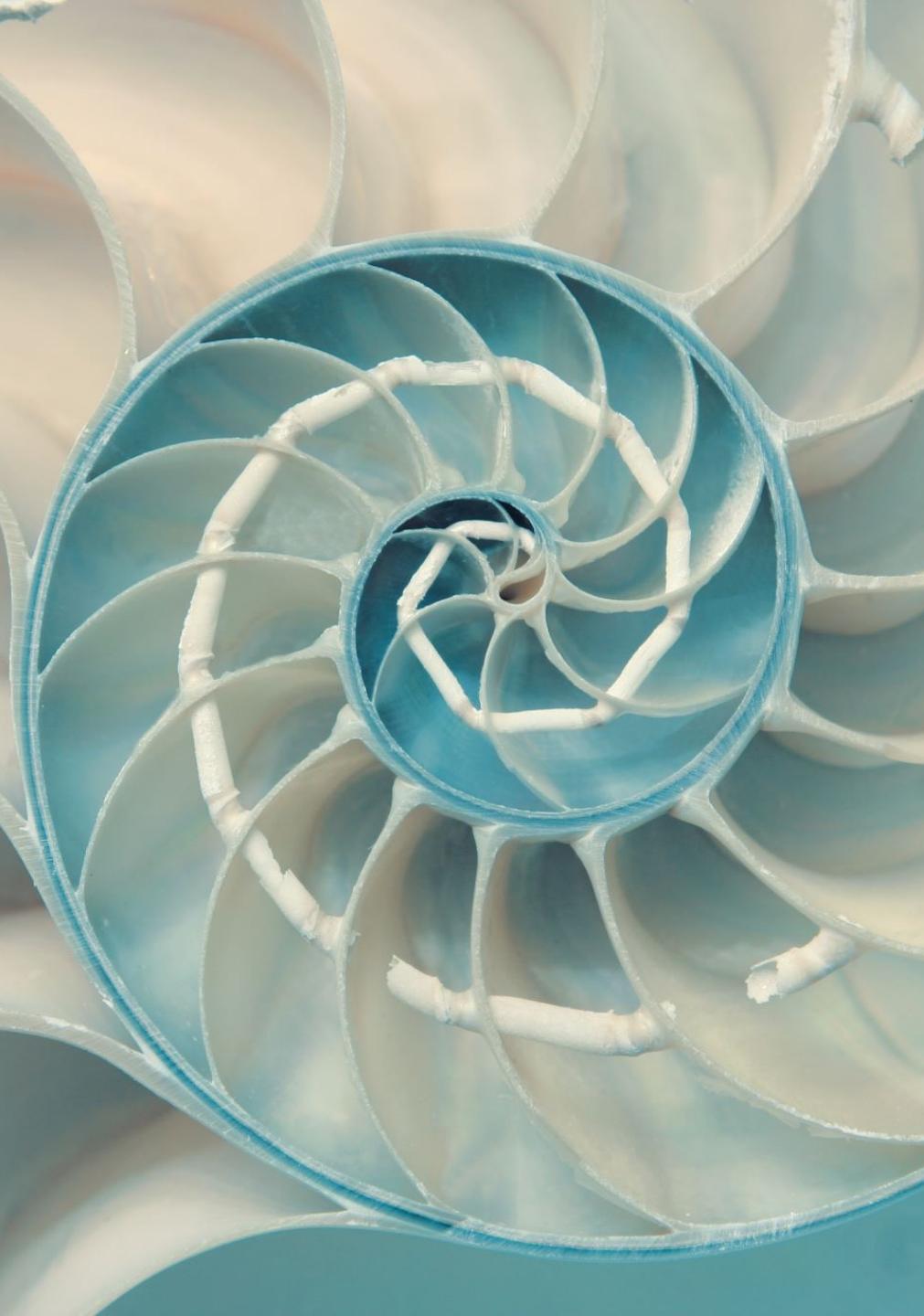
Flujos de Entrada

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



Ejercicio 1

- Pedir **dos** números **enteros** al usuario.
 $+,-,\ast,/,\%$
- Efectuar una operación sobre ambos números.
- Almacenar el resultado en una tercer variable e imprimir el resultado.



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 :
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

- 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()`

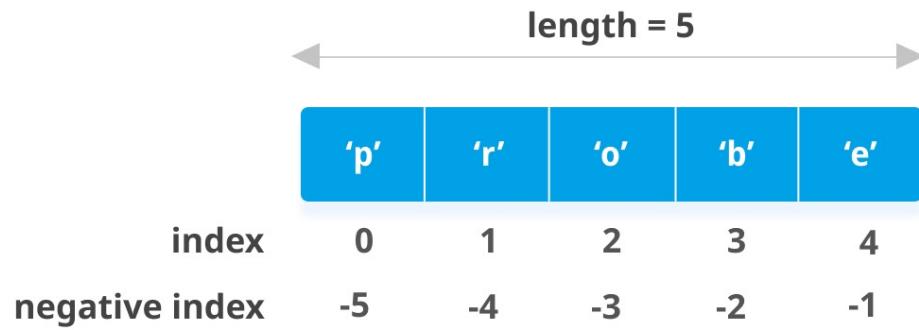
```
>>> vocales = list('aeiou')  
>>> vocales  
['a', 'e', 'i', 'o', 'u']
```

- 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

- 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).

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'>
```



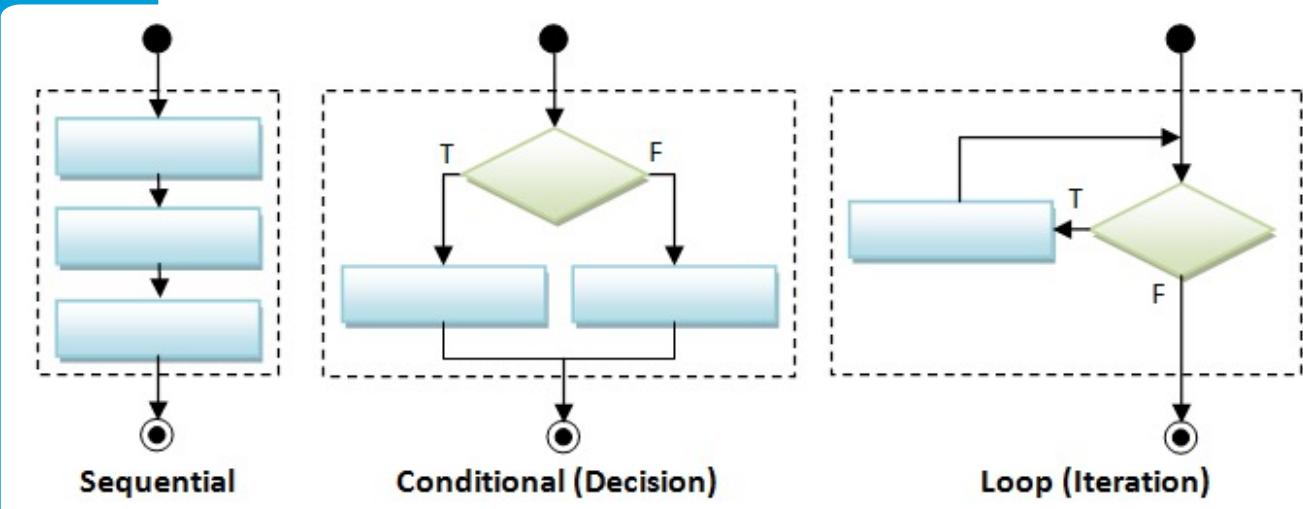
Booleanos en Python

Instrucciones de Control en Python



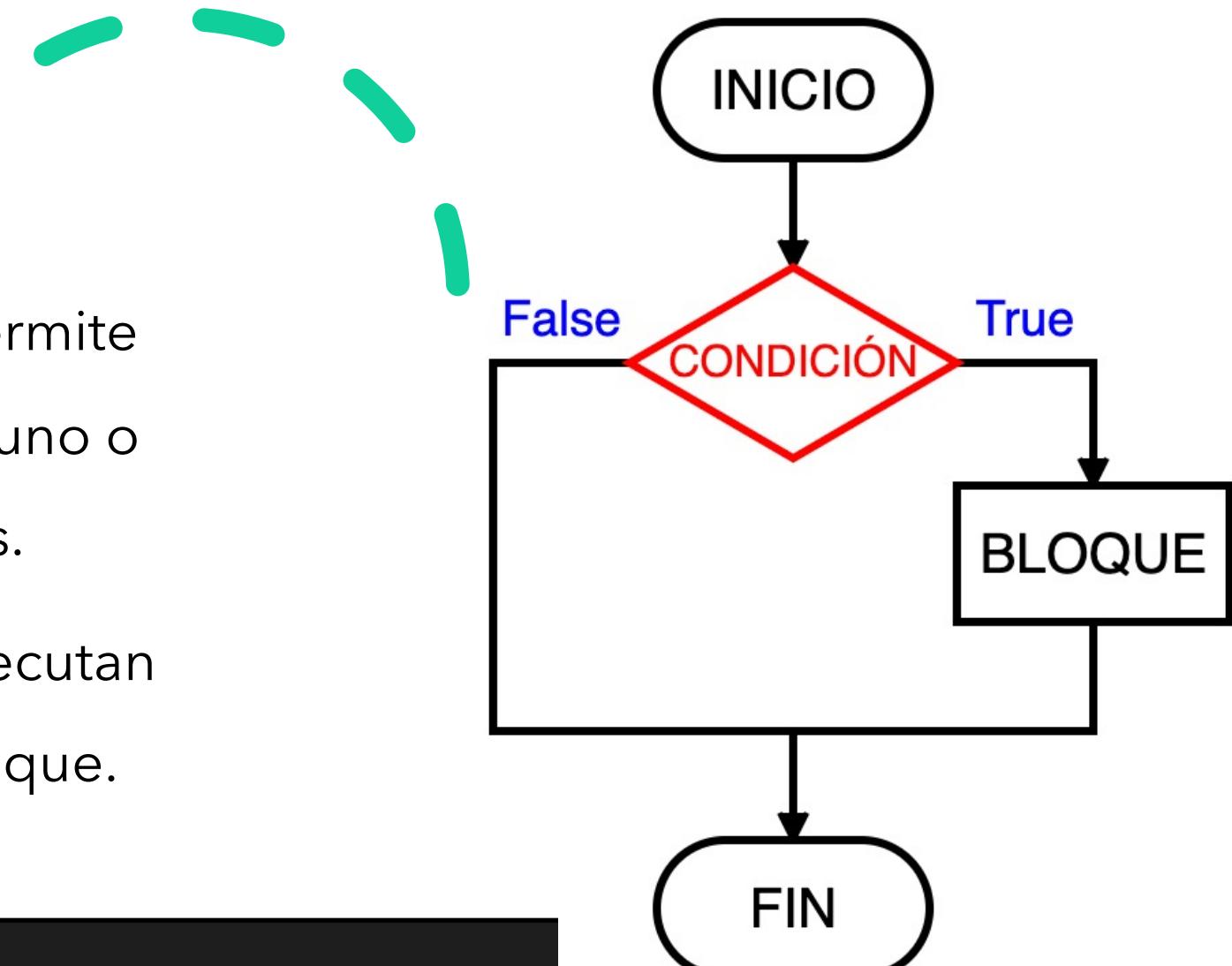
```
34
35     self.debug = debug
36     self.logger = logger
37     if path:
38         self._path = path
39         self._logger.info("Path set to %s", path)
40
41     @classmethod
42     def from_settings(cls, settings):
43         debug = settings.getboolean("debug")
44         return cls(job_dir=settings["job_dir"],
45                   debug=debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         return True
```

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.

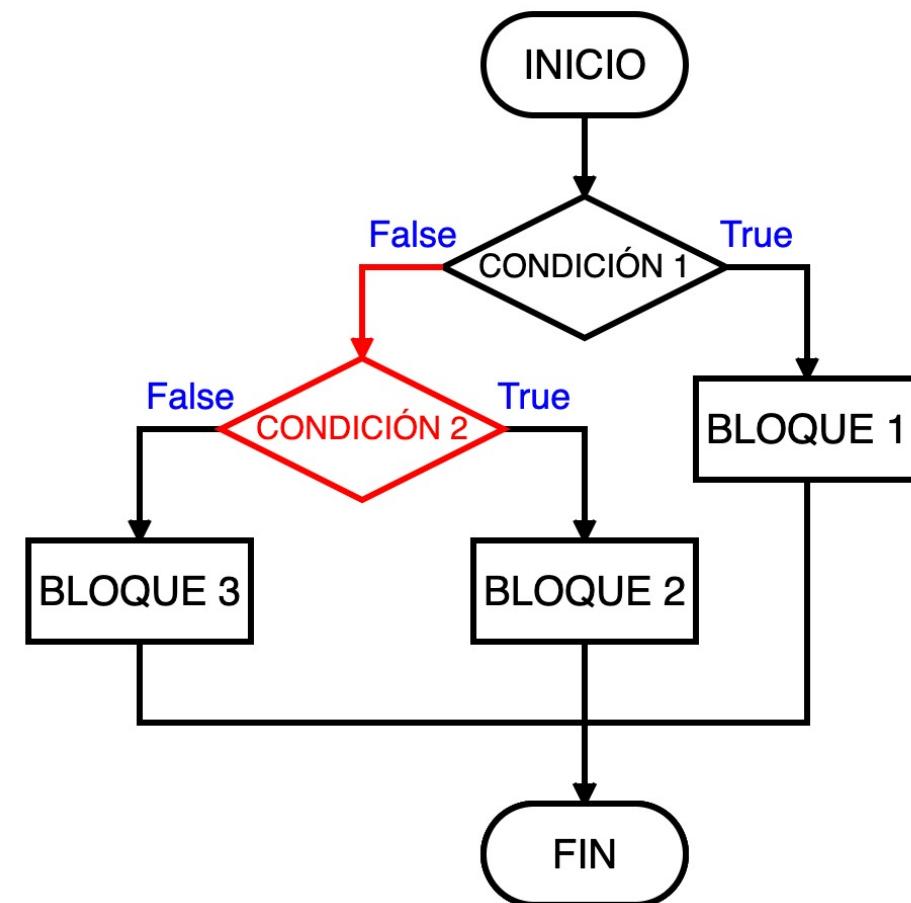


```
if condición:  
    aquí van las órdenes que se ejecutan si la condición es cierta  
    y que pueden ocupar varias líneas
```

```
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)





Ejercicio 6

1. Se deberá determinar si un número ingresado por el usuario es **par** o **impar**.
2. Escribir una función para determinar el **mayor** de dos números.

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. >>>
```

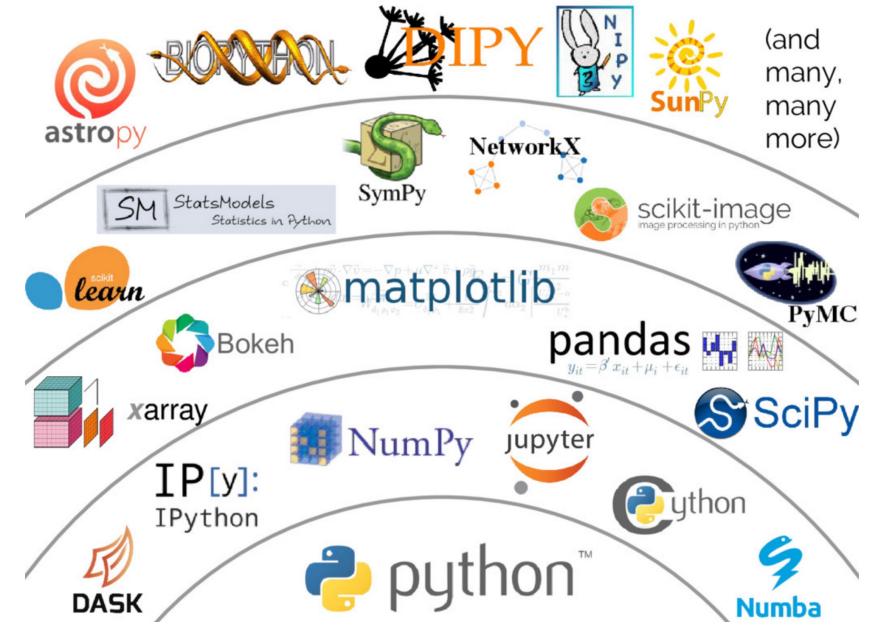


Ejercicio 8 – Ciclos For

- 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 o más grande.

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.

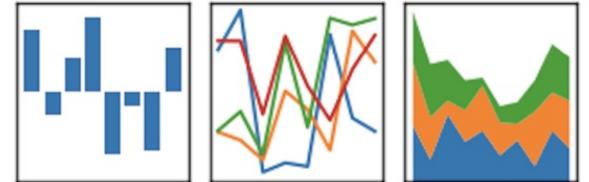


Pandas



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas



Data frames

Es una colección de datos habitualmente **tabulada**.

Contenida en una única tabla cada columna representa una variable y cada fila representa un registro.

	Mountain	Height (m)	Range	Coordinates	Parent mountain	First ascent	Ascents bef. 2004	Failed attempts bef. 2004
0	Mount Everest / Sagarmatha / Chomolungma	8848	Mahalangur Himalaya	27°59'17"N 86°55'31"E	NaN	1953	>>145	121.0
1	K2 / Qogir / Godwin Austen	8611	Baltoro Karakoram	35°52'53"N 76°30'48"E	Mount Everest	1954	45	44.0
2	Kangchenjunga	8586	Kangchenjunga Himalaya	27°42'12"N 88°08'51"E	Mount Everest	1955	38	24.0
3	Lhotse	8516	Mahalangur Himalaya	27°57'42"N 86°55'59"E	Mount Everest	1956	26	26.0
4	Makalu	8485	Mahalangur Himalaya	27°53'23"N 87°05'20"E	Mount Everest	1955	45	52.0
5	Cho Oyu	8188	Mahalangur Himalaya	28°05'39"N 86°39'39"E	Mount Everest	1954	79	28.0
6	Dhaulagiri I	8167	Dhaulagiri Himalaya	28°41'48"N 83°29'35"E	K2	1960	51	39.0
7	Manaslu	8163	Manaslu Himalaya	28°33'00"N 84°33'35"E	Cho Oyu	1956	49	45.0
8	Nanga Parbat	8126	Nanga Parbat Himalaya	35°14'14"N 74°35'21"E	Dhaulagiri	1953	52	67.0
9	Annapurna I	8091	Annapurna Himalaya	28°35'44"N 83°49'13"E	Cho Oyu	1950	36	47.0

Lectura de datos con Pandas

```
In [ ]: #Leer archivo CSV  
df = pd.read_csv("https://raw.githubusercontent.com/ulises1229/INTRO-PYTHON-  
ENESJ/master/data/salaries.csv")
```

Nota: El comando anterior tiene muchos argumentos opcionales para ajustar el proceso de importación de datos.

Hay una serie de comandos en pandas para leer **otros formatos de datos**:

```
pd.read_excel('myfile.xlsx', sheet_name='Sheet1', index_col=None, na_values=['NA'])  
pd.read_stata('myfile.dta')  
pd.read_sas('myfile.sas7bdat')  
pd.read_hdf('myfile.h5', 'df')
```

Explorar data frames

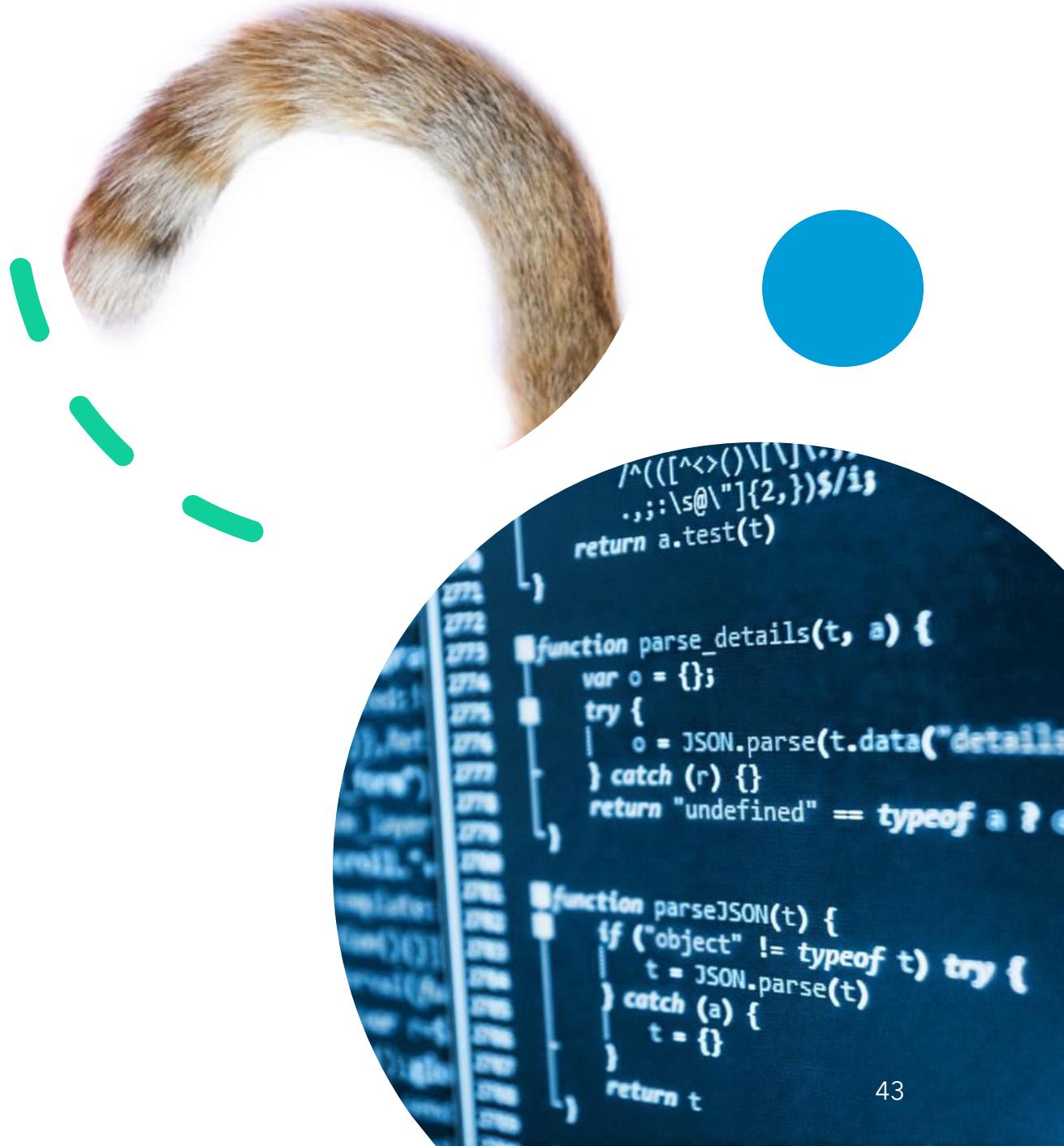
```
In [3]: #Lista de los primeros 5 registros  
df.head()
```

Out[3] :

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Ejercicio 9

- Se deberán leer los primeros 10, 20, 50 registros.
- ¿Podrían adivinar cómo ver los últimos registros?
- Pista:



Data Frames: Tipos de datos

Tipos en pandas	Tipos nativos en Python	Descripción
object	string	El dtype más general. Se asignará a la columna si la columna tiene tipos mixtos (números y cadenas).
int64	int	Caracteres numéricos. 64 se refiere a la memoria asignada para contener este carácter.
float64	float	Caracteres numéricos con decimales. Si una columna contiene números y NaNs (ver abajo), pandas usará por defecto float64, en caso de que el valor que falta tenga un decimal.
datetime64, timedelta[ns]	N/A (Ver el módulo datetime en Python)	Valores destinados a contener datos de tiempo. Busque en estos experimentos de series temporales.

Data Frame: Tipos de datos

```
In [4]: # Comprobar un tipo de columna determinado  
df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #Comprueba los tipos de todas las columnas  
df.dtypes
```

```
Out[4]: rank          object  
discipline      object  
phd            int64  
service         int64  
sex            object  
salary          int64  
dtype: object
```

Atributos de Data Frames

Los objetos de Python tienen atributos y métodos.

Atributos: usualmente se acceden a través de `.`

Métodos: usualmente se acceden a través de `()`

df.attribute	Descripción
dtypes	enumerar los tipos de las columnas
columns	listar los nombres de columna
axes	listar las etiquetas de fila y los nombres de columna
ndim	número de dimensiones
size	número de elementos
shape	devolver una tupla que representa la dimensionalidad
values	representación numpy de los datos

Ejercicio 10

1. Encuentre cuántos registros tiene este DataFrame
2. ¿Cuáles son los nombres de las columnas?
3. ¿Qué tipos de columnas tenemos en este data frame?

Métodos para Data Frames

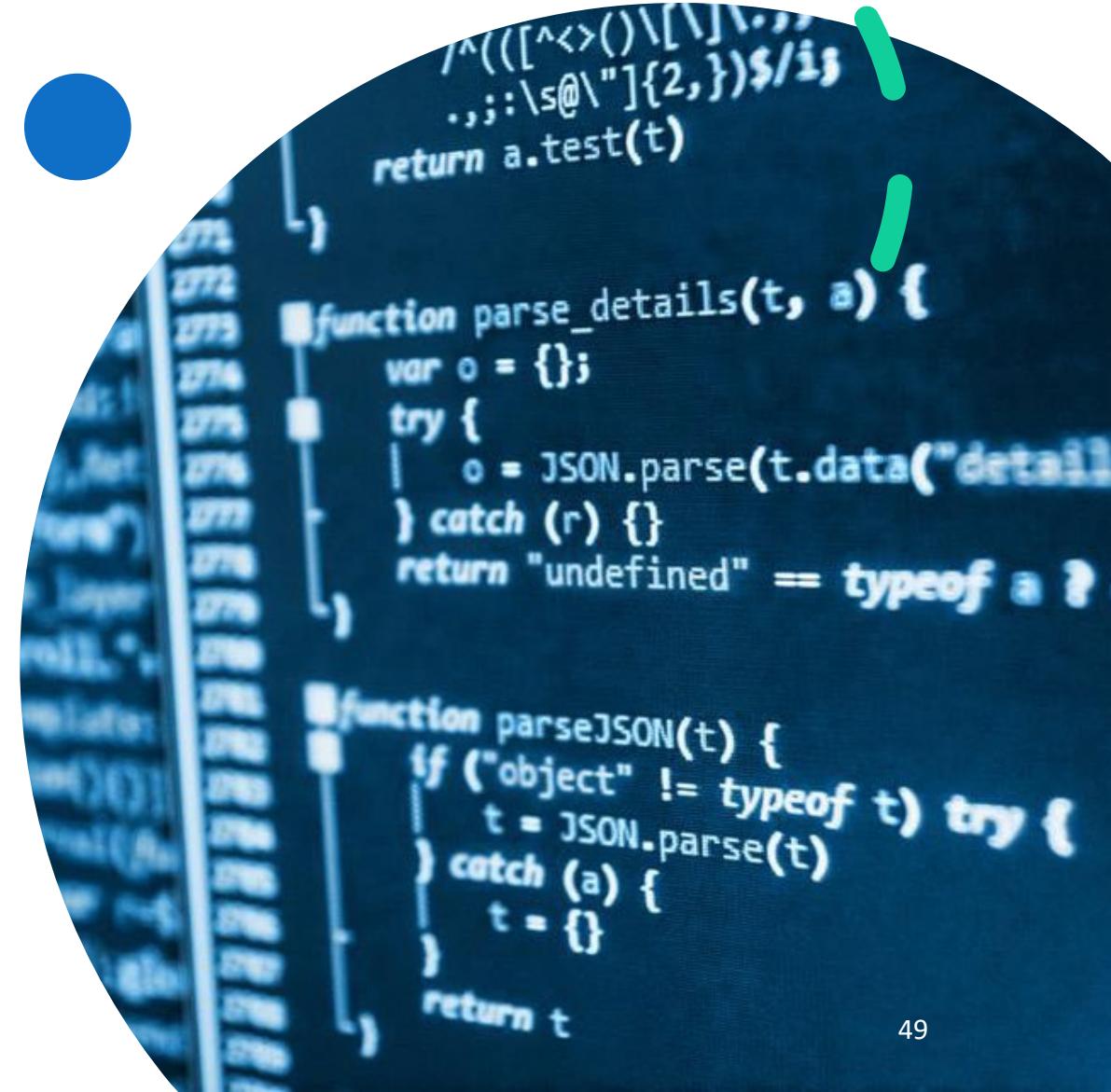
A diferencia de los atributos, los métodos en Python tienen paréntesis.

Todos los atributos y métodos se pueden enumerar con una función dir(): dir(df)

df.method()	Descripción
head([n]), tail([n])	Primera/última n filas
describe()	Generar estadísticas descriptivas (solo para columnas numéricas)
max(), min()	Devolver valores máximos/min para todas las columnas numéricas
mean(), median()	Devolver la media/mediana para todas las columnas numéricas
std()	Desviación estándar
sample([n])	devuelve una muestra aleatoria del marco de datos
dropna()	Eliminar todos los registros con valores NA

Ejercicio 11

1. Proporcione el resumen de las columnas numéricas del conjunto de datos
2. Calcular la desviación estándar para todas las columnas numéricas;
3. ¿Cuáles son las medias de los primeros 50 registros del conjunto de datos?
 - **Sugerencia:** Utilice el método `head()` para crear un subconjunto de los primeros 50 registros y, a continuación, calcule la media



Selección de una columna en el Data Frame

Método 1: Subconjunto del DF utilizando el nombre de columna:

`df['sex']`

Método 2: Utilice el nombre de columna como atributo:

`df.sex`

Nota: Existe un atributo rank para los data frames de pandas, por lo que para seleccionar una columna con un nombre "rank" debemos usar el método 1.

Ejercicio 12

1. Calcular las estadísticas básicas para la columna de salario;
 2. Encuentre cuántos valores hay en la columna de salario (utilice el método count);
 3. Calcular la media para la variable salario;

```
2769     ^(([^<>()\\[\\]\\.,,,"")  
2770     .,:\\s@\\"]{2,})$/is  
2771     return a.test(t)  
2772 }  
2773  
2774 ■■■■■function parse_details(t, a) {  
2775     var o = {};  
2776     try {  
2777         o = JSON.parse(t.data("details"))  
2778     } catch (r) {}  
2779     return "undefined" == typeof a ?  
2780     {} :  
2781  
2782 ■■■■■function parseJSON(t) {  
2783     if ("object" != typeof t) try {  
2784         t = JSON.parse(t)  
2785     } catch (a) {  
2786         t = {}  
2787     }  
2788     return t  
2789 },  
2790 }
```

Método *groupby* para Data Frames

Usando el método "**group by**" podemos:

- Dividir los datos en grupos en función de algunos criterios
- Calcular estadísticas (o aplicar una función) a cada grupo
- Similar a la función dplyr() en R

```
In [ ]: #Agrupando los datos usando rank  
df_rank = df.groupby(['rank'])
```

```
In [ ]: #Calcular la media de cada  
columna numérica por cada grupo  
df_rank.mean()
```

	phd	service	salary
rank			
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Método *groupby* para Data Frames

Una vez creado el objeto *groupby* podemos calcular varias estadísticas para cada grupo:

```
In [ ]: #Calculate mean salary for each professor rank:  
df.groupby('rank')[['salary']].mean()
```

rank	salary
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Nota: Si se utilizan corchetes individuales para especificar la columna (por ejemplo, salario), la salida es el objeto Pandas Series. Cuando se utilizan corchetes dobles, la salida es un data frame

Método *groupby* para Data Frames

- **groupby** notas de rendimiento:
 - No se produce ninguna agrupación/división hasta que sea necesario.
 - La creación del objeto groupby solo comprueba que ha pasado una asignación válida
 - Por defecto las claves de grupo se ordenan durante la operación groupby. Se puede que desee pasar *sort=False* para una posible aceleración:

```
In [ ]: #Calculate mean salary for each professor rank:  
df.groupby(['rank'], sort=False) [['salary']].mean()
```

Data Frame: filtering

Para crear un subconjunto de los datos, podemos aplicar la indexación booleana. Esta indexación se conoce comúnmente como filtro. Por ejemplo, si queremos crear un subconjunto de las filas en las que el valor salarial es mayor que \$120K:

```
In [ ]: #Calcular el salario promedio para cada rango de profesor:  
df_sub = df[ df['salary'] > 120000 ]
```

Cualquier operador booleano se puede utilizar para crear un subconjunto de los datos:

```
In [ ]:  
#Select only those rows that contain female professors:  
df_f = df[ df['sex'] == 'Female' ]
```

Data Frames: Ordenamientos

Podemos ordenar los datos por un valor en la columna. De forma predeterminada, la ordenación se producirá en orden ascendente y se devolverá un nuevo marco de datos.

```
In [ ]: # Cree un nuevo marco de datos a partir del original ordenado por la
columna Salario
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[]:

		rank	discipline	phd	service	sex	salary
55	AsstProf		A	2	0	Female	72500
23	AsstProf		A	2	0	Male	85000
43	AsstProf		B	5	0	Female	77000
17	AsstProf		B	4	0	Male	92000
12	AsstProf		B	1	0	Male	88000

Data Frames: Ordenamientos

Podemos ordenar los datos usando 2 o más columnas:

```
In [ ]: df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False] )  
df_sorted.head(10)
```

Out[]:

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

Ejercicio 13

1. Se deberán recuperar el profesor mejor y peor pagado de todo el dataframe.
2. Se deberá realizar el ejercicio anterior por sexo.
3. Se deberá calcular un promedio del salario por sexo.



Valores faltantes

Los valores que faltan se marcan como NaN

```
In [ ]: # Leer un conjunto de datos con valores faltantes
vuelos = pd.read_csv(" https://raw.githubusercontent.com/ulises1229/INTRO-PYTHON-ENESJ/master/data/flights.csv ")
```

```
In [ ]: # Seleccione las filas que tienen al menos un valor faltante
vuelos[vuelos.isnull().any(axis=1)].head()
```

```
Out[ ]:   year  month  day  dep_time  dep_delay  arr_time  arr_delay  carrier  tailnum  flight  origin  dest  air_time  distance  hour  minute
  330  2013      1     1    1807.0       29.0    2251.0        NaN      UA  N31412    1228    EWR    SAN        NaN    2425    18.0      7.0
  403  2013      1     1        NaN       NaN        NaN      AA  N3EHAAC    791     LGA    DFW        NaN    1389    NaN      NaN
  404  2013      1     1        NaN       NaN        NaN      AA  N3EVAA    1925     LGA    MIA        NaN    1096    NaN      NaN
  855  2013      1     2    2145.0       16.0       NaN      UA  N12221    1299    EWR    RSW        NaN    1068    21.0     45.0
  858  2013      1     2        NaN       NaN        NaN      AA      NaN    133     JFK    LAX        NaN    2475    NaN      NaN
```

Valores faltantes

Hay una serie de métodos para tratar con los valores que faltan en el marco de datos:

df.method()	Descripción
dropna()	Retirar observaciones faltantes
dropna(how='all')	Elimina todas las celdas con NA
dropna(axis=1, how='all')	Elimina la columna si faltan todos los valores
dropna(thresh = 5)	Elimina las filas que contienen menos de 5 valores que no faltan
fillna(0)	Reemplaza los valores faltantes por ceros
isnull()	Devuelve True si falta el valor
notnull()	Devuelve True para los valores que no faltan

Ejercicio Alternativo de Entrega

Collections

Collections - high quality data and datasets organized by topic.

The screenshot shows a grid of 15 collection cards, each with an icon, a title, a brief description, and a 'View Collection' button. The categories include:

- Bibliographic data: Existing databases or services providing substantial bibliographic data.
- Climate Change: A collection of the most important "general" datasets on climate change.
- Demographics (population): Population data and data analytics.
- Economic Data and Indicators: A collection of economic indicators available on DataHub.
- Education: US education data.
- Football: A collection of awesome football datasets including national teams, clubs, match schedules etc.
- GeoJSON: GeoJSON datasets available on DataHub.
- Health Care Data: Ready-to-use datasets on DataHub about Health Care.
- Inflation: Datasets re Inflation.
- Linked Open Data: An overview of the Linked Open Data datasets.
- Logistics: Ready-to-use logistics datasets - explore, download and use in your tool!
- Machine Learning / Statistical: Examples of machine learning datasets.
- Movies and TV: Various resources for Movies and TV data.
- Open Corporates: Open Database of corporate entities.
- Property Prices: Property Prices Datasets available on DataHub.

- De forma individual, deberán seleccionar una **categoría** y **set de datos** en la plataforma datahub: <https://datahub.io/>.
1. Deberán obtener al menos 3 métricas de estadística descriptiva de todo el conjunto de datos
 - **Media**
 - **STD**
 - **MEDIANA**
 - **MODA**
 - **VAR**
 2. Repetir el ejercicio anterior solo para un subconjunto de datos.