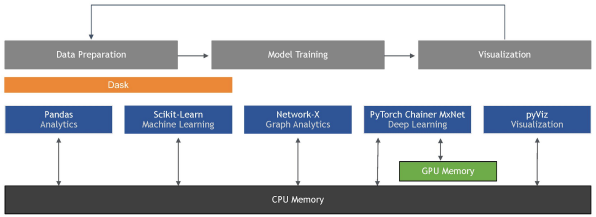
Funciones y Métodos en Python

RAFAEL MARTÍN R.

Funciones y Métodos en Python



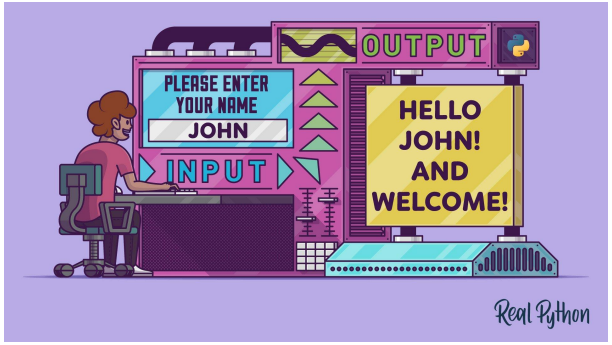
Herramientas esenciales para la ciencia de datos

Las funciones y métodos son componentes fundamentales en Python que permiten escribir código modular y reutilizable. En esta presentación, exploraremos cómo estos elementos son esenciales para la ciencia de datos, desde la manipulación de datos hasta el análisis estadístico. Aprenderemos a crear funciones personalizadas, utilizar métodos integrados y aplicar estos conceptos en proyectos reales de análisis de datos.

RAFAEL MARTÍN R.

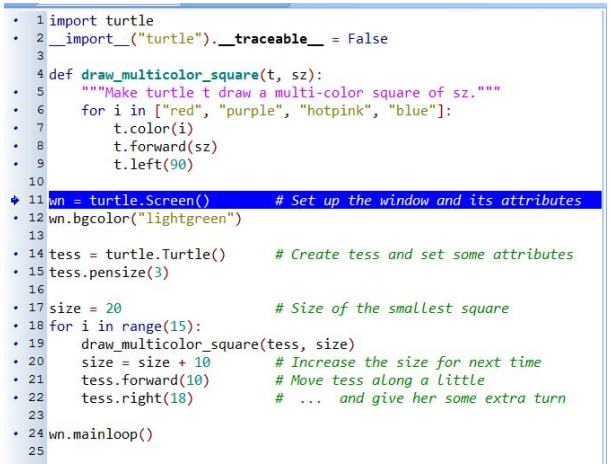
¿Qué es una Función en Python?

Una función en Python es un bloque de código reutilizable que encapsula una tarea específica. Como una caja negra, acepta entradas (parámetros), realiza operaciones definidas y produce salidas (resultados). Las funciones nos permiten escribir código más organizado, evitar repeticiones y facilitar el mantenimiento de nuestros programas. Son fundamentales para la programación modular y eficiente.



RAFAEL MARTÍN R.

Sintaxis de Funciones

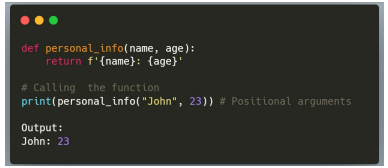


```python
import turtle
__import__("turtle").__traceable__ = False

def draw_multicolor_square(t, sz):
    """Make turtle t draw a multi-color square of sz."""
    for i in ["red", "purple", "hotpink", "blue"]:
        t.color(i)
        t.forward(sz)
        t.left(90)

wn = turtle.Screen()        # Set up the window and its attributes
wn.bgcolor("lightgreen")

tess = turtle.Turtle()      # Create tess and set some attributes
tess.pensize(3)

size = 20                   # Size of the smallest square
for i in range(15):
    draw_multicolor_square(tess, size)
    size = size + 10        # Increase the size for next time
    tess.forward(10)        # Move tess along a little
    tess.right(18)          #  ...  and give her some extra turn

wn.mainloop()

```

Estructura básica de una función en Python

- def: palabra clave para definir una función
- nombre_funcion: identificador descriptivo y significativo
- parametros: valores que la función recibe
- cuerpo: bloque de código indentado
- return: devuelve el resultado de la función
- def calcular_promedio(numeros): return sum(numeros)/len(numeros)

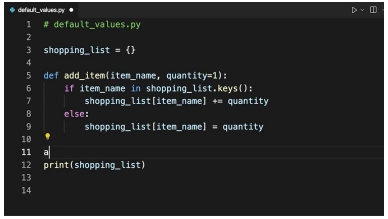
Parámetros y Argumentos



```
def personal_info(name, age):
    return f'{name}: {age}'

# Calling  the function
print(personal_info("John", 23)) # Positional arguments

Output:
John: 23
```

Parámetros Posicionales

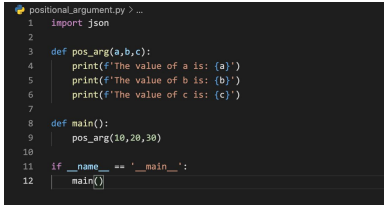
- Orden específico al llamar la función
- Obligatorios si no tienen valor por defecto
- Ejemplo: def suma(a, b)



```
default_values.py
# default_values.py

shopping_list = {}

def add_item(item_name, quantity=1):
    if item_name in shopping_list.keys():
        shopping_list[item_name] += quantity
    else:
        shopping_list[item_name] = quantity

a
print(shopping_list)
```

Parámetros con Valores por Defecto

- Valores predefinidos si no se especifican
- Opcionales al llamar la función
- Ejemplo: def descuento(precio, porcentaje=10)



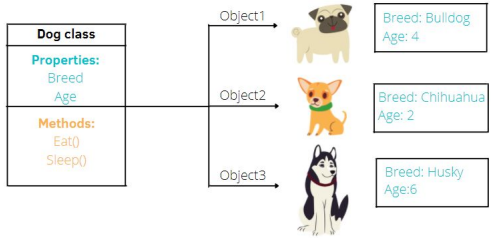
```
positional_argument.py > ...
import json

def pos_arg(a,b,c):
    print(f'The value of a is: {a}')
    print(f'The value of b is: {b}')
    print(f'The value of c is: {c}')

def main():
    pos_arg(10,20,30)

if __name__ == '__main__':
    main()
```

Args y Kwargs

- *args para número variable de argumentos
- **kwargs para argumentos nombre=valor variables
- Permite flexibilidad en llamadas a funciones

¿Qué es un Método?

Un método es una función especial que pertenece a un objeto o clase en Python. A diferencia de las funciones regulares, los métodos siempre están asociados a una instancia específica y pueden acceder y modificar sus atributos mediante el parámetro 'self'. Los métodos son fundamentales en la programación orientada a objetos, permitiendo que los objetos tengan comportamientos específicos y mantengan su estado interno.



Métodos vs Funciones

+ Funciones

- Bloques de código independientes y reutilizables
- Se definen con 'def' y pueden existir solas
- Pueden ser llamadas directamente
- No requieren un objeto o clase

× Métodos

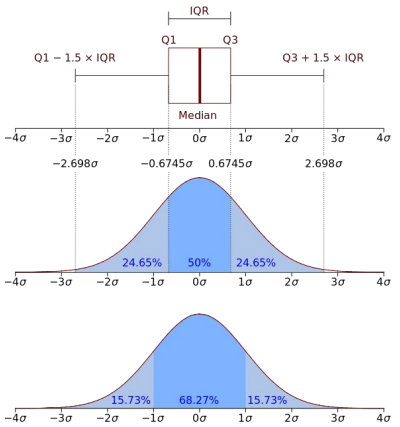
- Siempre pertenecen a una clase u objeto
- Requieren 'self' como primer parámetro
- Se acceden mediante notación punto (objeto.método())
- Pueden acceder a atributos del objeto

Funciones en Ciencia de Datos

```python
def load_data():

    #set the path to csv
    filePath = '/content/employee_data.csv'
    # read file
    employeeDF = pd.read_csv(filePath)

    # Display Info about the data
    print(employeeDF.info())

    #Rename column
    employeeDF =
employeeDF.rename(columns={"number":"EmpId","first_name":"Firstname","
last_name":"Lastname","birth_date":"Birthdate","gender":"Gender"})

    #setting datatype
    employeeDF['Birthdate'] =
employeeDF['Birthdate'].astype('datetime64')
    #gender is a category
    employeeDF['Gender'] = employeeDF['Gender'].astype('category')

    #See the changes
    print("Updated Data")
    print(employeeDF.info())

    #Display top 5 rows of data
    print(employeeDF.head())

    #Get statistics of the data
    print(employeeDF.describe(include='all'))

    return employeeDF
```

Funciones para Manipulación de Datos

- Limpieza: eliminar duplicados y valores nulos
- Transformación: normalización y estandarización de datos
- Cálculos: medias, desviaciones y correlaciones



Ejemplo: Normalización de Datos

- Función que escala valores entre 0 y 1
- Aplicación a columnas numéricas del DataFrame
- Visualización de resultados normalizados

Métodos en Pandas

df.head() =>

	Anime	Episodes	Year
0	One Piece	1009	1999
1	Naruto	720	2002
2	Bleach	366	2004
3	Hunter X Hunter	148	2011
4	Attack On Titan	74	2013

df.head(8) =>

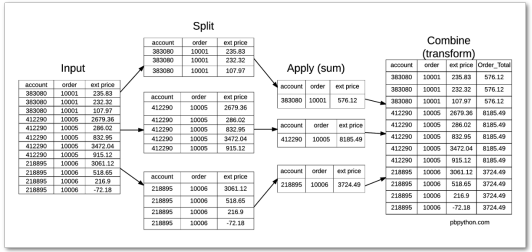
	Anime	Episodes	Year
0	One Piece	1009	1999
1	Naruto	720	2002
2	Bleach	366	2004
3	Hunter X Hunter	148	2011
4	Attack On Titan	74	2013
5	Gintama	366	2006
6	Code Geass	50	2007
7	Death Note	37	2008

NBA Dataset:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

Summary Table Generated by .describe() Method:

	Number	Age	Weight	Salary
count	457.000000	457.000000	457.000000	4.460000e+02
mean	17.678337	26.938731	221.522976	4.842684e+06
std	15.966090	4.404016	26.368343	5.229238e+06
min	0.000000	19.000000	161.000000	3.088800e+04
25%	5.000000	24.000000	200.000000	1.044792e+06
50%	13.000000	26.000000	220.000000	2.839073e+06
75%	25.000000	30.000000	240.000000	6.500000e+06
max	99.000000	40.000000	307.000000	2.500000e+07



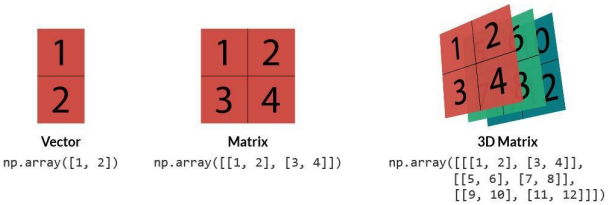
Métodos de Series y DataFrames

Análisis Descriptivo

Agregación y Transformación

RAFAEL MARTÍN R.

Métodos en NumPy



- mean(), std(), min(), max() para análisis estadístico básico
- reshape() y transpose() para reorganizar arrays
- concatenate() y stack() para combinar arrays
- sort() y argsort() para ordenamiento de datos
- unique() y where() para filtrado y búsqueda
- dot() y matmul() para operaciones matriciales

Creando Funciones para Análisis

```
In [2]: 1 sns.get_dataset_names()
```

```
Out[2]: ['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'taxis',
 'tips',
 'titanic']
```

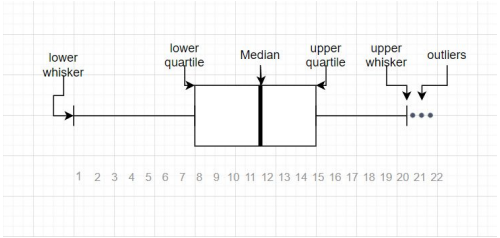
```
In [3]: 1 df = sns.load_dataset('titanic')
        2 df_copy =df.copy()  # keep a copy just in case
        3 df.head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class
0	0	3	male	22.0	1	0	7.2500	S	Third

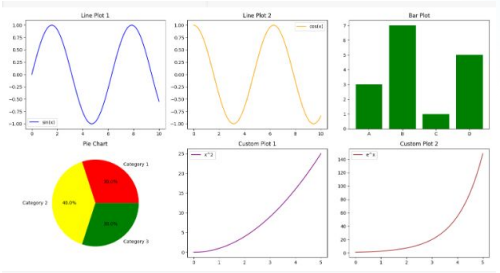
Análisis Exploratorio de Datos

Función personalizada que automatiza la generación de estadísticas descriptivas y distribuciones de variables numéricas en datasets.



Detección de Outliers

Implementación de una función que identifica valores atípicos utilizando métodos estadísticos como Z-score o IQR.



Visualización Automatizada

Función que genera automáticamente gráficos relevantes basados en el tipo de datos y las relaciones entre variables.

RAFAEL MARTÍN R.

Ejercicio Práctico 1

Limpieza y Preparación

Crear función clean_data() para manejar valores nulos y estandarizar formatos de datos.

Análisis Estadístico

Implementar calculate_stats() para obtener media, mediana y desviación estándar del dataset.

Visualización

Desarrollar plot_data() para crear histogramas y gráficos de dispersión automáticamente.

Ejercicio Práctico 2: Métodos de Pandas

Análisis del Dataset

Cargar un dataset de ventas y explorar sus características usando head(), info() y describe()

Métodos de Agregación

Aplicar groupby() y agg() para calcular estadísticas por categoría de productos

Transformación de Datos

Usar apply() y transform() para normalizar columnas numéricas y crear nuevas características

Solución Guiada

Revisión paso a paso del código y explicación de los resultados obtenidos

Mejores Prácticas

Guía para escribir código eficiente y mantenible

- Usar nombres descriptivos que reflejen la funcionalidad del código
- Documentar funciones con docstrings claros y ejemplos
- Implementar manejo de errores con try-except
- Aplicar el principio DRY evitando código repetitivo
- Mantener funciones cortas y con propósito único
- Realizar pruebas unitarias para validar el funcionamiento

Recursos Adicionales

Documentación Oficial

- Python Docs: Funciones y métodos incorporados
- Pandas API Reference para DataFrames
- NumPy Documentation para arrays y cálculos

Recursos de Práctica

- Ejercicios prácticos en GitHub
- Notebooks de ejemplos en Kaggle
- Desafíos de programación en CodeWars

Material Complementario

- Cursos gratuitos en DataCamp
- Tutoriales en Real Python
- Foros de discusión en Stack Overflow

Resumen y Conclusiones

Conceptos Clave y Próximos Pasos

Las funciones y métodos son fundamentales en Python para la ciencia de datos. Las funciones son bloques de código reutilizables independientes, mientras que los métodos están vinculados a objetos específicos. Su aplicación en análisis de datos permite automatizar tareas, mantener código limpio y realizar operaciones complejas de manera eficiente. Para continuar aprendiendo, practica creando tus propias funciones y explora la documentación de las bibliotecas principales.