Introducción a

Python

Temas

Intro

Intro

Python es un lenguaje de programación:

>_ De propósito general

>_ Interpretado

>_ Multiparadigma

>_ No tipado

>_ Multiplataforma

Intro

Muy popular porque:

>_ Es muy amigable y fácil de aprender

>_ Oferece una amplia variedad de librerías

>_ Versátil para múltiples aplicaciones

Intro

Entornos de Desarrollo:

>_ IDEs: --pycharm --vscode --sublime

>_ Notebooks: --jupyter

Intro

Instalación y Setup:

>_ Anaconda/miniconda --download

>_ Python.org --download

Entornos de Ejecución Online:

>_ GoogleColab --open

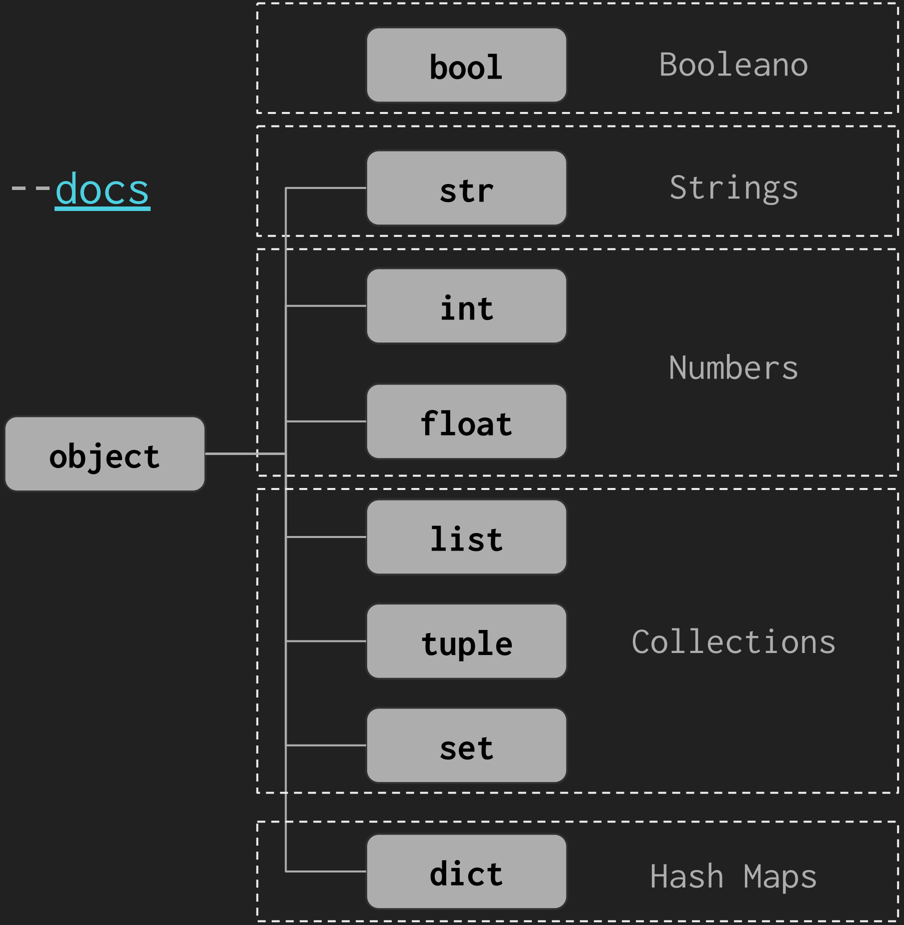
>_ Kaggle --open

>_ DeepNote --open

Sintaxis

Sintaxis

```
>_ tipos_de_datos --docs
```



Sintaxis

>_ estructuras_de_datos --docs

```
>>> str_var = "hola"
>>> str_var = 'hola'
>>> int_var = 1
>>> float_var = 1.0
>>> list_var = [1,2,3]
>>> list_var = list() # empty list
>>> bool_var = True
```

```
>>> bool_var = False
>>> tuple_var = (1,2,3)
>>> tuple_var = tuple() # empty tuple
>>> set_var = {1,2,3}
>>> set_var = set() # empty set
>>> dict_var = {"key":"value"}
>>> dict_var = dict(key = "value")
```

Sintaxis

>_ estructuras_de_datos --docs

```
>>> list_var = [1,2,3,4]
>>> list_var[0]
1
>>> list_var[-1]
4
>>> list_var[1:3]
[2,3]
```

```
>>> list_var.append(5)
>>> list_var
[1,2,3,4,5]
>>> list_var.remove(1)
>>> list_var
[2,3,4,5]
```

Sintaxis

>_ estructuras_de_datos --docs

```
>>> dict_var = {
      "k1":"value",
      "k2":"value",
      }
>>> dict_var["k1"]
'value'
>>> dict_var.get("k1")
'value'
```

```
>>> dict_var.keys()
dict_keys(["k1","k2"])
>>> dict_var.values()
dict_values(['value','value'])
>>> dict_var.items()
dict_values([("k1",'value'),("k2",'value')])
```

Sintaxis

```
>_ estructuras_de_control --docs
```

```python
n = 10
if n > 0:
    print("n is positive")
elif n < 0:
    print("n is negative")
else:
    print("n is zero")
```

Sintaxis

>_ estructuras_de_control --docs

```python
for i in range(10):
    print(i)
```

```python
words = ["hello","world"]
for word in words:
    print(word)
```

Sintaxis

```
>_ estructuras_de_control --docs
```

```
n = 10
while(n>0):
    print(n)
    n-=1
```

Sintaxis

>_ estructuras_de_control --[docs]

```python
for i in range(10):
    if i == 0: # just passing by
        pass
    if i % 2 == 0:
        print("not printing this number")
        continue
    print(i)
    if i == 9:
        print("breaking the loop")
        break
```

Sintaxis

>_ funciones_y_clases

```
def suma(a, b):
    return a + b

>>> suma(1,2)
3
```

```
suma = lambda a,b: a+b
>>> suma(1,2)
3
```

Sintaxis

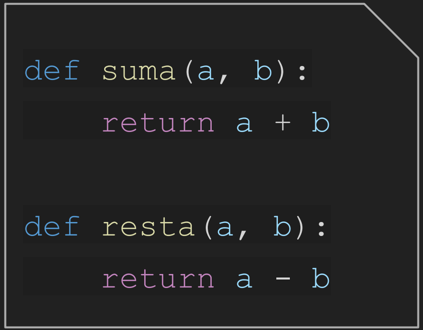
>_ funciones_y_clases

```python
class Person:
    def __init__(self, name, age): # constructor
        self.name = name
        self.age = age
    def present(self):
        print(f"Hi, my name is {self.name} and I'm {self.age} years old.")
```

```
>>> p = Person("Vito", 28)
>>> p.present()
Hi, my name is Vito and I'm 28 years old.
```

Sintaxis

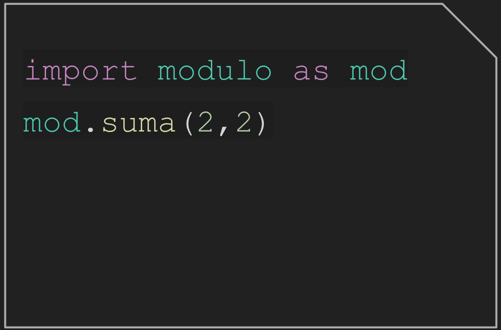
>_ modulos_y_paquetes --docs



```python
def suma(a, b):
    return a + b

def resta(a, b):
    return a - b
```

modulo.py



```python
import modulo as mod
mod.suma(2,2)
```

main.py

Sintaxis

>_ modulos_y_paquetes --[docs]

```
|- calculuspkg/
     |- __init__.py
     |- modulo.py
|- main.py
```

```python
from calculuspkg import modulo
modulo.suma(2,2)
```

main.py

Sintaxis

>_ naming_conventions

Function	function, my_function
Variable	x, var, my_variable
Class	Model, MyClass
Method	class_method, method
Constant	CONSTANT, MY_CONSTANT, MY_LONG_CONSTANT
Module	module.py, my_module.py
Package	package, mypackage

Standard Library

Standard Library

>_ os --operating-system

>_ re --regex

>_ datetime

>_ argparse --console-applications

Oficial Docs

Standard Library

>_ os --docs

```python
import os
os.getcwd() # returns the current working directory
os.listdir(".") # list of files
os.path.join("../relative", "file.py") # "../relative/file.py"
os.path.exists("some_file_or_path.py") # true/false
os.path.basename("gets/the/base/of/a/path/file.py") # returns file.py
```

Standard Library

>_ re --docs --auto-regex

```
import re
pattern = "[0-9]?[0-9]:[0-9][0-9]" # match hour like pattern hh:mm
string = "a string with a pattern 13:20 inside"
re.search(pattern=pattern,string=string).group() # "13:20"
re.sub(pattern,"",string) # "a string with a pattern  inside"
re.findall(pattern=pattern,string=string) # ["13:20"]
re.split(pattern=pattern,string=string) # ["a string with a pattern", "inside"]
```

Standard Library

>_ datetime --docs

```
import datetime
datetime.datetime.now() # datetime.datetime(2023, 4, 24, 10, 54, 39, 969737)
datetime.datetime.fromisoformat("2022-03-02")
datetime.datetime.now() - datetime.datetime.fromisoformat("2022-03-02") # timedelta
```

Standard Library

>_ argparse

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("echo", help="echo the string you use here")
args = parser.parse_args()
print(args.echo)
```

Data Analysis

Data Analysis

```
>_ numpy --docs

>_ pandas --docs

>_ matplotlib --docs

>_ seaborn --docs

>_ pyspark --docs          Big Data
```

Data Analysis

```
>_ numpy --docs
```

‘NumPy is the fundamental package for scientific computing in Python’

Data Analysis

```
>_ pandas --docs
```

‘Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool’

Data Analysis

```
>_ matplotlib --docs
```

‘Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python’

Data Analysis

```
>_ seaborn --docs
```

‘Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.’

Data Analysis

```
>_ pyspark --docs
```

‘PySpark is the Python API for Apache Spark. It enables you to perform real-time, large-scale data processing in a distributed environment using Python.’

Machine Learning

Machine Learning

```
>_ scikit-learn --docs

>_ statsmodels --docs

>_ pytorch --docs

>_ tensorflow --docs

>_ huggingface --docs

>_ pyspark --docs            Big Data
```

Machine Learning

```
>_ scikit-learn --docs
```

‘Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning’

Machine Learning

>_ statsmodels --docs

‘Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration’

Machine Learning

```
>_ pytorch --docs
```

‘End-to-end Machine Learning Framework. PyTorch enables fast, flexible experimentation and efficient production through a user-friendly front-end, distributed training, and ecosystem of tools and libraries.’

Machine Learning

```
>_ tensorflow --docs
```

‘TensorFlow is an end-to-end platform that makes it easy for you to build and deploy ML models.’

Machine Learning

```
>_ huggingface --docs
```

‘Build, train and deploy state of the art models powered by the reference open source in machine learning.’

Machine Learning

```
>_ pyspark --docs
```

‘Pyspark MLlib is a scalable machine learning library that provides a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.’

Web

Web

```
>_ django --docs

>_ fastapi --docs

>_ requests --docs

>_ beautiful-soup --docs
```

Web

>_ django --docs

‘Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.’

Web

```
>_ fastapi --docs
```

‘FastAPI is a modern, fast (high-performance), web framework for building APIs with Python.’

Web

>_ requests --docs

‘Requests is an elegant and simple HTTP library for Python, built for human beings.’

Web

```
>_ beautiful-soup --docs
```

‘Beautiful Soup is a Python library for pulling data out of HTML and XML files.’