Python para computación científica 🚀

NumPy, aspectos del cálculo del arreglo n-dimensional

C++ Review DUNE

Es una *organización* en GitHub con diversos recursos para aprender a programar en C++ y Python con la caja de herramientas DUNE Numerics.

- Resumen del estudio de DUNE Numerics en el año 2021 📚💻
- Pad `https://hackmd.io/@cpp-review-dune/S1p2a43No`

C++ Review DUNE

Es una *organización* en GitHub con diversos recursos para aprender a programar en C++ y Python con la caja de herramientas DUNE Numerics.

- Resumen del estudio de DUNE Numerics en el año 2021 📚💻
- Pad `https://hackmd.io/@cpp-review-dune/S1p2a43No`

Grupo en Telegram Review Python PeC[3]

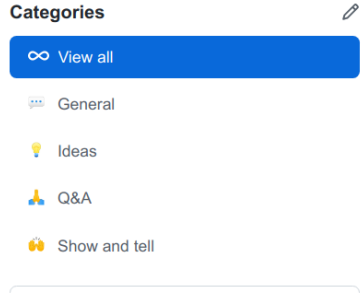
- Compartir pantalla.
- Preferir los enlaces que los archivos `$(du -sh file.ext) < 5MiB`.
- Audio grupal.

Más información en `https://telegram.org/blog/group-video-calls`.

GitHub Discussions 💬

- Adjuntar imágenes.
- Soporta resaltado de sintáxis de C++/Python.
- Debate de ideas.
- Responder preguntas y votar respuestas.
- `https://github.com/cpp-review-dune/python/discussions`.

Más información en `https://docs.github.com/es/discussions`.



Vistazo general de Python

En la actualidad, Python es un *lenguaje de scripting*

- favorito para **iniciarse en la programación**.
- preferido por los especialistas en la **seguridad informática**.
- **más utilizado** en la inteligencia artificial.
- de propósito general, multiparadigma, dinámicamente tipado e **interpretado**.
- usado como *bindings* (`cython`, `pybind11`, `nanobind`) para otros lenguajes como C/C++/Fortran.
- desde la versión `3.10` está **enfocado en el rendimiento**, con en el proyecto `github.com/faster-cpython`.
- prefiere el uso de **nombres en inglés** que símbolos, por ejemplo: `and`, `or`, `not`, `as`, `continue`, `finally`.

Vistazo general de Python

En la actualidad, Python es un *lenguaje de scripting*

- favorito para **iniciarse en la programación**.
- preferido por los especialistas en la **seguridad informática**.
- **más utilizado** en la inteligencia artificial.
- de propósito general, multiparadigma, dinámicamente tipado e **interpretado**.
- usado como *bindings* (`cython`, `pybind11`, `nanobind`) para otros lenguajes como C/C++/Fortran.
- desde la versión `3.10` está **enfocado en el rendimiento**, con en el proyecto `github.com/faster-cpython`.
- prefiere el uso de **nombres en inglés** que símbolos, por ejemplo: `and`, `or`, `not`, `as`, `continue`, `finally`.

Nuetras *reglas de programación* serán:

1. Piense antes de programar.
2. Un programa es un ensayo legible por humanos sobre la resolución de problemas que también se ejecuta en una computadora.
3. La mejor forma de mejorar sus habilidades de programación y resolución de problemas es practicando.
4. Pruebe su código a menudo y a fondo.
5. Si fue difícil de escribir, posiblemente sea difícil de entender. Agregue un comentario.
6. Toda entrada es mala hasta que se pruebe lo contrario.
7. Una *función* debe realizar una sola tarea.
8. Asegúrese de que su nueva *clase* pase las pruebas.

Elementos especiales en Python

Las palabras clave, símbolos y caracteres que pueden ser usado en un programa de Python son conocidos como *tokens*.

Palabras clave

Son palabras especiales que **no pueden usarse para nombrar** cosas. Algunas de ellas son `lambda`, `class`, `raise`, `try`, `return`, `import`, `yield`, `None`, `except`, `global`, `del`, `from`, `pass`, `with`, `break`, `is`, `def`.

Elementos especiales en Python

Las palabras clave, símbolos y caracteres que pueden ser usado en un programa de Python son conocidos como *tokens*.

Palabras clave

Son palabras especiales que **no pueden usarse para nombrar** cosas. Algunas de ellas son `lambda`, `class`, `raise`, `try`, `return`, `import`, `yield`, `None`, `except`, `global`, `del`, `from`, `pass`, `with`, `break`, `is`, `def`.

Operadores

Son secuencias de caracteres que tiene un significado para el intérprete de Python. Su uso implica alguna operación como adición, sustracción o algo similar. Algunas de ellas son `+`, `**`, `//`, `<>` `<<`, `>>`, `≠`, `%=`, `&=`, `~`.

Elementos especiales en Python

Las palabras clave, símbolos y caracteres que pueden ser usado en un programa de Python son conocidos como *tokens*.

Palabras clave

Son palabras especiales que **no pueden usarse para nombrar** cosas. Algunas de ellas son `lambda`, `class`, `raise`, `try`, `return`, `import`, `yield`, `None`, `except`, `global`, `del`, `from`, `pass`, `with`, `break`, `is`, `def`.

Operadores

Son secuencias de caracteres que tiene un significado para el intérprete de Python. Su uso implica alguna operación como adición, sustracción o algo similar. Algunas de ellas son `+`, `**`, `//`, `<>` `<<`, `>>`, `≠`, `%=`, `&=`, `~`.

Puntuadores y delimitadores

Separan diferentes elementos en declaraciones y expresiones de Python. Algunos lo reconocerás de las matemáticas y otras del idioma inglés. Algunos de ellos son `(`, `)`, `[`, `]`, `,`, `:`, `.`, `=`, `'`, `"`, `\`, `@`.

Elementos especiales en Python

Las palabras clave, símbolos y caracteres que pueden ser usado en un programa de Python son conocidos como *tokens*.

Palabras clave

Son palabras especiales que **no pueden usarse para nombrar** cosas. Algunas de ellas son `lambda`, `class`, `raise`, `try`, `return`, `import`, `yield`, `None`, `except`, `global`, `del`, `from`, `pass`, `with`, `break`, `is`, `def`.

Operadores

Son secuencias de caracteres que tiene un significado para el intérprete de Python. Su uso implica alguna operación como adición, sustracción o algo similar. Algunas de ellas son `+`, `**`, `//`, `<>` `<<`, `>>`, `≠`, `%=`, `&=`, `~`.

Puntuadores y delimitadores

Separan diferentes elementos en declaraciones y expresiones de Python. Algunos lo reconocerás de las matemáticas y otras del idioma inglés. Algunos de ellos son `(`, `)`, `[`, `]`, `,`, `:`, `.`, `=`, `'`, `"`, `\`, `@`.

Literales

Es una notación para **representar un valor fijo**, el cual es un valor que no puede cambiar en un programa. En contraste a los literales, las variables y símbolos que puede asignarse un valor que puede ser modificado durante la ejecución de un código. Por ejemplo, `123` es un literal, este es un valor fijo y no puede ser modificado.

Elementos especiales en Python

Las palabras clave, símbolos y caracteres que pueden ser usado en un programa de Python son conocidos como *tokens*.

Palabras clave

Son palabras especiales que **no pueden usarse para nombrar** cosas. Algunas de ellas son `lambda`, `class`, `raise`, `try`, `return`, `import`, `yield`, `None`, `except`, `global`, `del`, `from`, `pass`, `with`, `break`, `is`, `def`.

Operadores

Son secuencias de caracteres que tiene un significado para el intérprete de Python. Su uso implica alguna operación como adición, sustracción o algo similar. Algunas de ellas son `+`, `**`, `//`, `<>` `<<`, `>>`, `≠`, `%=`, `&=`, `~`.

Puntuadores y delimitadores

Separan diferentes elementos en declaraciones y expresiones de Python. Algunos lo reconocerás de las matemáticas y otras del idioma inglés. Algunos de ellos son `(`, `)`, `[`, `]`, `,`, `:`, `.`, `=`, `'`, `"`, `\`, `@`.

Literales

Es una notación para **representar un valor fijo**, el cual es un valor que no puede cambiar en un programa. En contraste a los literales, las variables y símbolos que puede asignarse un valor que puede ser modificado durante la ejecución de un código. Por ejemplo, `123` es un literal, este es un valor fijo y no puede ser modificado.

Se recomienda seguir la guía de estilo para el código Python PEP 8 y el uso de estos ayudantes como dev-dependencias:

- `autopep8`, formats Python code to conform to the PEP 8 style guide.
- `black`, *The uncompromising code formatter*.
- `Ruff`, *An extremely fast Python linter*.
- `mypy`, *An optional static type checker for Python*.

En programación, se habla de *ciudadanos de primera clase* para referirse a un elemento del lenguaje que posee la mayor cantidad de privilegios dentro del mismo lenguaje. Algunas de sus características son

- Aparecer en una expresión.
- Estar asignado a una variable.
- Ser usado como argumento.
- Ser devuelto por una llamada de función.

En programación, se habla de *ciudadanos de primera clase* para referirse a un elemento del lenguaje que posee la mayor cantidad de privilegios dentro del mismo lenguaje. Algunas de sus características son

- Aparecer en una expresión.
- Estar asignado a una variable.
- Ser usado como argumento.
- Ser devuelto por una llamada de función.

En Python, toda cosa en el sistema es considerado como un objeto. Un objeto en Python tiene

- una identidad,
- algunos atributos,
- cero o más nombres.

Cada vez que Python crea un objeto, recibe un número de identificación.

Funciones Python

Las *funciones* en los lenguajes de programación comparten muchas características de las *funciones matemáticas*, pero agrega algunas características que las hagan más útiles en la programación. En particular, una función Python:

- representa **una sola operación** ha ser desarrollada.
- toma cero o más **argumentos** como entrada.
- **retorna un solo valor** (potencialmente un objeto compuesto) como salida.

Una función es importante porque representa una *encapsulación*. Por encapsulación, queremos decir que los detalles de una operación se pueden ocultar, proporcionando operaciones más gruesas que nosotros, como programadores, podemos usar sin tener que entender los detalles internos de la función.
En más detalle, las funciones proveen las siguientes características que ayudan en la programación:

- Resolución de problema divide y vencerás: dividen los programas en partes más pequeñas.
- Abstracción: proporcionan una interfaz de operación de nivel superior que la función.

🌟 ¿Cuándo usar una función?

Según [1], estas son algunas pautas que pueden resultar útil.

- **Hacer una sola cosa y bien**: la función debe ser la encapsulación de una única operación, funciones que intenten hacer también muchas cosas son candidatas a ser divididas en múltiples funciones (refactorizadas).
- **Legible**.
- **No demasiado largo**: Si la función es demasiado larga, es posible que debe dividirse en múltiples funciones.
- **Reutilizable**: De ser posible, la función debe ser autónoma y no depende de algún matiz de una llamada del programa. Si tiene dependencias, estas deben ser explícitas para que otros puedan usarlo fácilmente.
- **Completo**: Asegúrse que funcione en todas las situaciones posibles y tenga en cuenta todos los casos que podría usarse.
- **Refactorización**: Es el proceso de tomar el código existente y modificarlo de manera que su estructura mejore de alguna manera conservando la misma funcionalidad.

Algunas veces escribimos funciones sin la declaración return. Las funciones que no retornan un valor son llamadas procedimientos. En ese caso retorna None que representa la nada.

Archivos y excepciones

Un archivo es una colección de bytes que usualmente reside en el disco. Los archivos soportan tanto la codificación ASCII como Unicode.

Referencias

[1] W. F. Punch y R. Enbody, *The Practice of Computing Using Python*, 3.ª ed. Pearson Education, 2017.