Página 1

Página 2

Página 3

Módulo15Tarefa1

Página 3

Aqui nós vamos *desenvolver* nosso **primeiro** ambiente no ***Streamlit***.

😄 Código 15:

Diferentes tipos de avisos:

🚨 Caixa de Texto de Error.

⚠️ Caixa de Texto de Aviso.

ℹ️ Caixa de Texto de Informação.

✅ Caixa de Texto de Sucesso.

😄 Código 16:

Utilização do código, para obter ajuda e mais informações sobre:

```
st.help(np.sum)
```

np.sum _ArrayFunctionDispatcher numpy.sum(a, axis=None, dtype=None, out=None, keep

```
Sum of array elements over a given axis.

Parameters
----------
a : array_like
    Elements to sum.
axis : None or int or tuple of ints, optional
    Axis or axes along which a sum is performed.  The default,
    axis=None, will sum all of the elements of the input array.  If
    axis is negative it counts from the last to the first axis.

    .. versionadded:: 1.7.0

    If axis is a tuple of ints, a sum is performed on all of the axes
    specified in the tuple instead of a single axis or all the axes as
    before.
dtype : dtype, optional
    The type of the returned array and of the accumulator in which the
    elements are summed.  The dtype of `a` is used by default unless `a`
    has an integer dtype of less precision than the default platform
    integer.  In that case, if `a` is signed then the platform integer
    is used while if `a` is unsigned then an unsigned integer of the
    same precision as the platform integer is used.
out : ndarray, optional
    Alternative output array in which to place the result. It must have
    the same shape as the expected output, but the type of the output
    values will be cast if necessary.
keepdims : bool, optional
    If this is set to True, the axes which are reduced are left
    in the result as dimensions with size one. With this option,
    the result will broadcast correctly against the input array.

    If the default value is passed, then `keepdims` will not be
    passed through to the `sum` method of sub-classes of
    `ndarray`, however any non-default value will be.  If the
    sub-class' method does not implement `keepdims` any
    exceptions will be raised.
initial : scalar, optional
    Starting value for the sum. See `~numpy.ufunc.reduce` for details.

    .. versionadded:: 1.15.0
```

```
where : array_like of bool, optional
    Elements to include in the sum. See `~numpy.ufunc.reduce` for details.

    .. versionadded:: 1.17.0

Returns
-------
sum_along_axis : ndarray
    An array with the same shape as `a`, with the specified
    axis removed.   If `a` is a 0-d array, or if `axis` is None, a scalar
    is returned.  If an output array is specified, a reference to
    `out` is returned.

See Also
--------
ndarray.sum : Equivalent method.

add.reduce : Equivalent functionality of `add`.

cumsum : Cumulative sum of array elements.

trapz : Integration of array values using the composite trapezoidal rule.

mean, average

Notes
-----
Arithmetic is modular when using integer types, and no error is
raised on overflow.

The sum of an empty array is the neutral element 0:

>>> np.sum([])
0.0

For floating point numbers the numerical precision of sum (and
``np.add.reduce``) is in general limited by directly adding each number
individually to the result causing rounding errors in every step.
However, often numpy will use a  numerically better approach (partial
pairwise summation) leading to improved precision in many use-cases.
This improved precision is always provided when no ``axis`` is given.
When ``axis`` is given, it will depend on which axis is summed.
Technically, to provide the best speed possible, the improved precision
is only used when the summation is along the fast axis in memory.
Note that the exact precision may vary depending on other parameters.
In contrast to NumPy, Python's ``math.fsum`` function uses a slower but
```

```
more precise approach to summation.
Especially when summing a large number of lower precision floating point
numbers, such as ``float32``, numerical errors can become significant.
In such cases it can be advisable to use `dtype="float64"` to use a higher
precision for the output.

Examples
--------
>>> np.sum([0.5, 1.5])
2.0
>>> np.sum([0.5, 0.7, 0.2, 1.5], dtype=np.int32)
1
>>> np.sum([[0, 1], [0, 5]])
6
>>> np.sum([[0, 1], [0, 5]], axis=0)
array([0, 6])
>>> np.sum([[0, 1], [0, 5]], axis=1)
array([1, 5])
>>> np.sum([[0, 1], [np.nan, 5]], where=[False, True], axis=1)
array([1., 5.])

If the accumulator is too small, overflow occurs:

>>> np.ones(128, dtype=np.int8).sum(dtype=np.int8)
-128

You can also start the sum with a value other than zero:

>>> np.sum([10], initial=5)
15
```

😄 Código 17:

Retorna paramentos em QUERY:

```
▼ {
   ▼ "show_map" : [
      0 : "True"
   ]
   ▼ "selected" : [
      0 : "asia"
```

```
    1 : "america"
  ]
}
```

😄 Código 18:

Exemplo de botão:

Botão

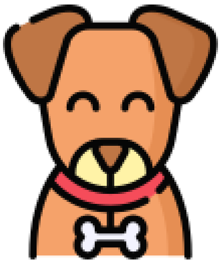
😄 Código 19:

Imagem com funçãod de botão (Clique):

```
st.title("DOG")

st.markdown("[![Click me](https://cdn-icons-png.flaticon.com/128/2454/2454302.png)
```

DOG



😄 Código 20:

Selecionar cores:

🟩

A cor escolhida foi #00f900

😎 FIM 😎