Objects for Type Hinting

Various built-in types for type hinting are provided. Currently, two types exist -- `GenericAlias <types-genericalias>` and `Union <types-union>`. Only `GenericAlias` is exposed to C.

> Create a `GenericAlias <types-genericalias>` object. Equivalent to calling the Python class `types.GenericAlias`. The *origin* and *args* arguments set the `GenericAlias`'s `__origin__` and `__args__` attributes respectively. *origin* should be a `PyTypeObject*`, and *args* can be a `PyTupleObject*` or any `PyObject*`. If *args* passed is not a tuple, a 1-tuple is automatically constructed and `__args__` is set to `(args,)`. Minimal checking is done for the arguments, so the function will succeed even if *origin* is not a type. The `GenericAlias`'s `__parameters__` attribute is constructed lazily from `__args__`. On failure, an exception is raised and `NULL` is returned.
>
> Here's an example of how to make an extension type generic:
>
> ```
> ...
> static PyMethodDef my_obj_methods[] = {
>     // Other methods.
>     ...
>     {"__class_getitem__", Py_GenericAlias, METH_O|METH_CLASS, "my_obj is
> generic over its contained type"}
>     ...
> }
> ```
>
> The data model method `~object.__class_getitem__`.
>
> 3.9

> The C type of the object returned by `Py_GenericAlias`. Equivalent to `types.GenericAlias` in Python.
>
> 3.9

□□□ #2
□ □□□ □□□ 2026-10-31 16:55:00 UTC
□ □□□ □□□ 2026-10-31 16:55:01 UTC