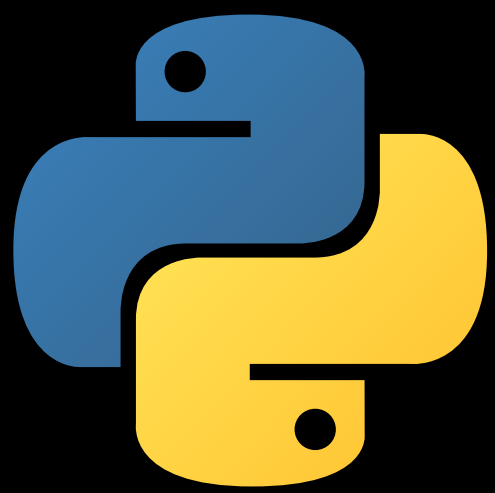
#PYTHONTIPS



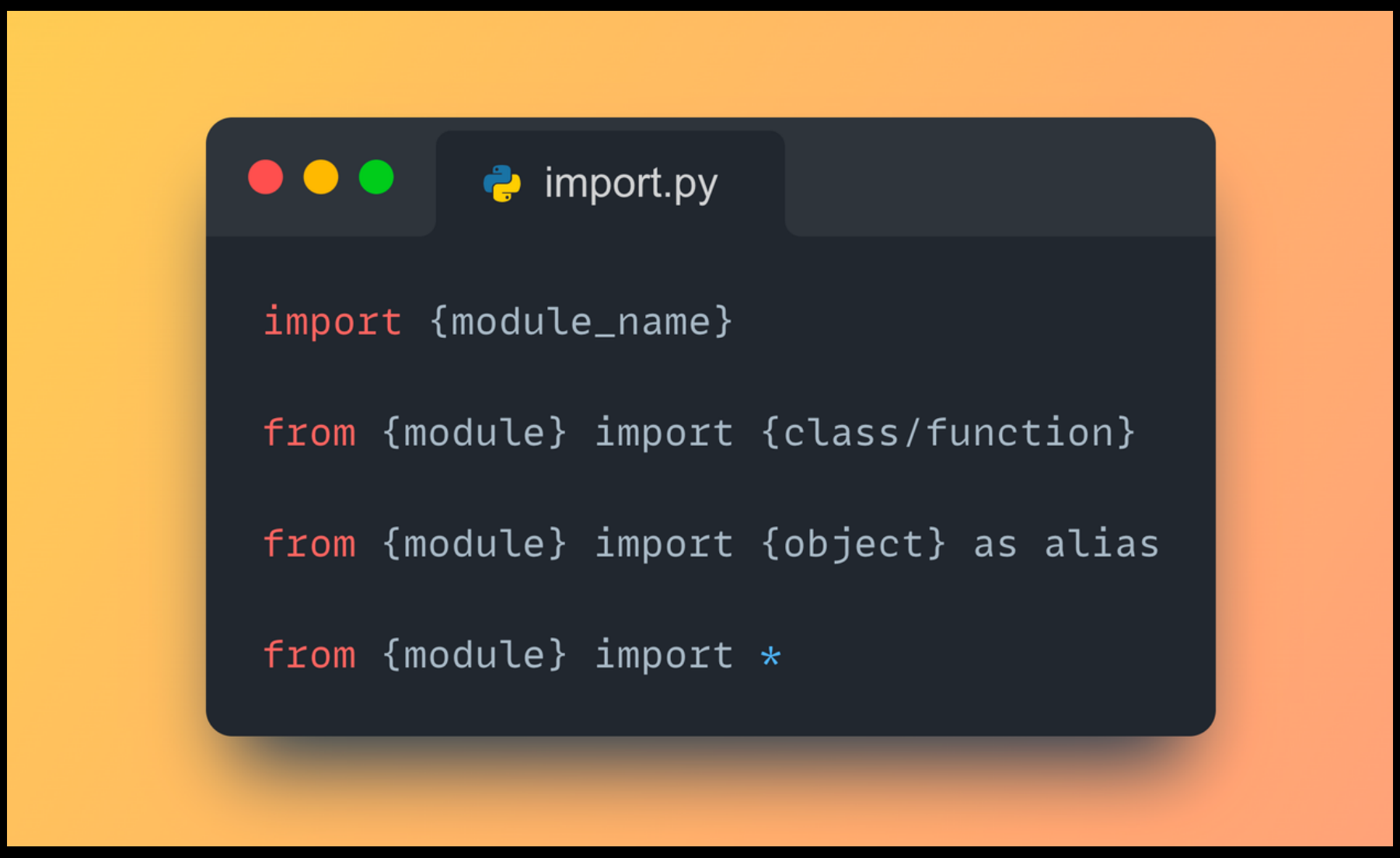
IMPORT UNDER THE HOOD

IN PYTHON

@VEDANTSOLANKI

WHAT IS IMPORT ?

- Python's import statement enables the user to use particular modules functionality in any other program or module.
- Following are the ways to import any module or objects to other module or python script



#PYTHONTIPS

SO,
WHAT HAPPENS
WHEN WE
'IMPORT'
A
MODULE ?

@VEDANTSOLANKI

IMPORT {MODULE_NAME}

- When we import a module, Python does two things:
  1. It checks if the module has been loaded and cached in the sys.modules. If not, it will execute the module and create a reference to the module object.
  2. After that it will add the module name to the global namespace referencing the same module object.
- let's understand this with the help of an example

#PYTHONTIPS

import.py

```
'''
  The following program imports the datetime module and
  prints out the datetime object in the sys.modules and
  global namespace:
'''

>>> import sys
>>> import datetime

>>> print('sys.modules:', hex(id(sys.modules['datetime'])))
# output : ('sys.modules:', '0x110140478')

>>> if 'math' in globals():
...     print('globals: ', hex(id(globals()['math'])))
# output :

>>> if 'datetime' in globals():
...     print('globals: ', hex(id(globals()['datetime'])))
# output : ('globals: ', '0x110140478')
```

@VEDANTSOLANKI

- As you can see we did not get any output for math but for datetime.
- Here datetime variable references the same object module
- Now if you again try to import the same datetime module for the second time, python won't execute the datetime module again but it will get it from sys.modules cache

FROM {MODULE} IMPORT {CLASS/FUNCTION}

- When we import any function , class or any other object from a module python does the exact same two things
- Just the difference is it will create a reference to the module object and not the entire module
- And it will add a variable which refers to a imported object to the global namespace
- let's understand this with the help of an example

#PYTHONTIPS

import.py

```
>>> import sys
>>> from pprint import pprint
>>> from datetime import date
>>> print('sys.modules:', hex(id(sys.modules['datetime'])))
# output : ('sys.modules:', '0x1101405c8')

>>> pprint(globals())
'''
output :
{'__builtins__': <module '__builtin__' (built-in)>,
 '__doc__': None,
 '__name__': '__main__',
 '__package__': None,
 'date': <type 'datetime.date'>,
 'pprint': <function pprint at 0x11013e230>,
 'sys': <module 'sys' (built-in)>
}
'''
```

- As you can see in above example, python loaded the datetime module into the sys.modules.
- However , it only created a reference to the date function and not the datetime module object in the global namespace

@VEDANTSOLANKI

#PYTHONTIPS

FROM {MODULE} IMPORT {OBJECT} AS ALIAS

- When you import any function , class or any other object from a module and uses an alias for that python does the exact same two things
- Just the difference is it will create an alias that reference to the module object and not the entire module
- And it will add an alias which refers to a imported object to the global namespace
- let's understand this with the help of an example

#PYTHONTIPS

import.py

```
>>> import sys
>>> from datetime import date as d
>>> from pprint import pprint
>>> print('sys.modules:', hex(id(sys.modules['datetime'])))
# output : ('sys.modules:', '0x110180478')

>>> print('globals:', hex(id(globals()['date'])))
'''
output :
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'date'
'''

>>> print('globals:', hex(id(globals()['d'])))
# output : ('globals:', '0x10a80f420')

>>> pprint(globals())
'''
output :
{'__builtins__': <module '__builtin__' (built-in)>,
 '__doc__': None,
 '__name__': '__main__',
 '__package__': None,
 'ceiling': <built-in function ceil>,
 'd': <type 'datetime.date'>,
 'pprint': <function pprint at 0x11017e320>,
 'sys': <module 'sys' (built-in)>
}
'''
```

Notice this

@VEDANTSOLANKI

FROM {MODULE} IMPORT *

- When we import a module, Python does two things:

1. It checks if the module has been loaded and cached in the sys.modules. If not, it will execute the module and create a reference to the module object.
2. Add all objects from the module to the global namespace

- As you can see in below example, Python adds all the functions from the datetime module to global namespaces & if any reference already exist then python will replace their references

- Though this format often leads to bugs that are difficult to track. Therefore, it's adviced to not use this format

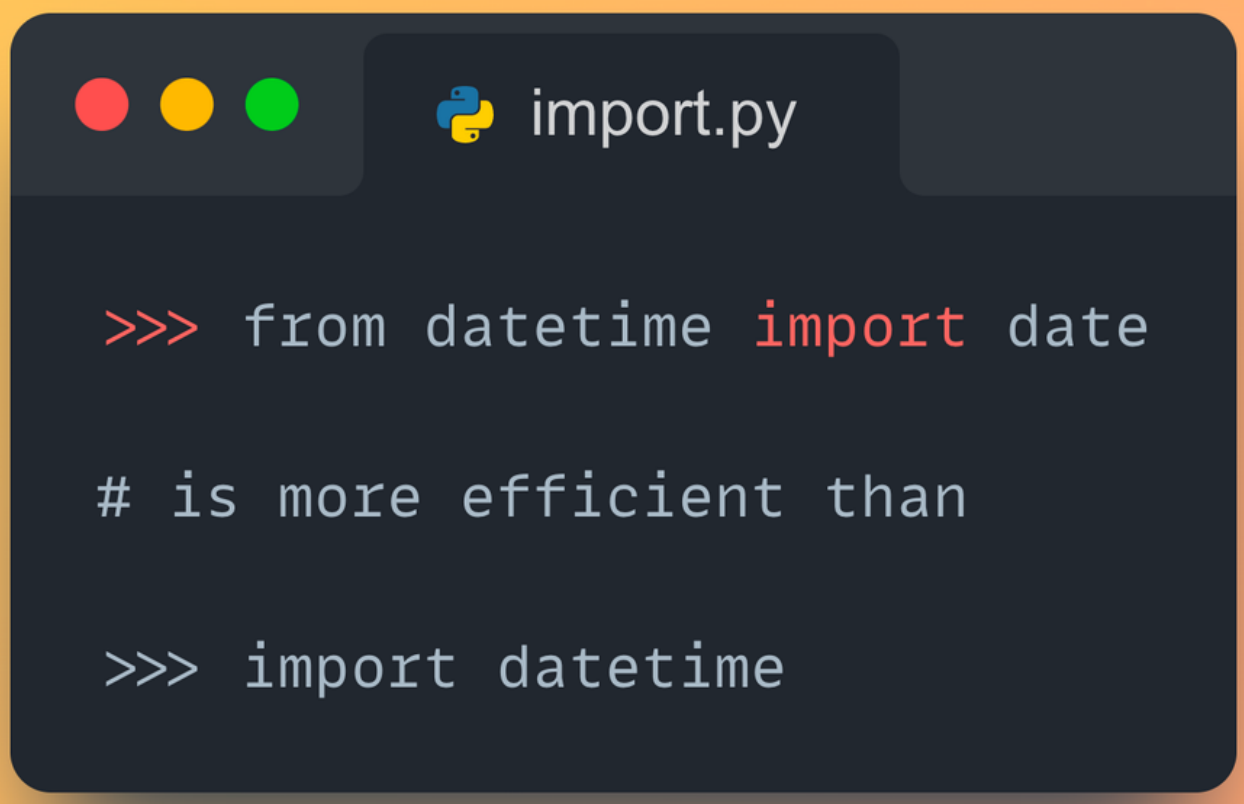
import.py

```
>>> import sys
>>> from pprint import pprint
>>> from datetime import *
>>> print('sys.modules:', hex(id(sys.modules['datetime'])))
# output : ('sys.modules:', '0x1101805c8')

>>> pprint(globals())
'''
output :
{'MAXYEAR': 9999,
 'MINYEAR': 1,
 '__builtins__': <module '__builtin__' (built-in)>,
 '__doc__': None,
 '__name__': '__main__',
 '__package__': None,
 'date': <type 'datetime.date'>,
 'datetime': <type 'datetime.datetime'>,
 'datetime_CAPI': <capsule object "datetime.datetime_CAPI" at 0x110183b40>,
 'pprint': <function pprint at 0x11017e230>,
 'sys': <module 'sys' (built-in)>,
 'time': <type 'datetime.time'>,
 'timedelta': <type 'datetime.timedelta'>,
 'tzinfo': <type 'datetime.tzinfo'>}
'''
```

MISCONCEPTION

- One of the most common misconceptions of the import statement is that many consider



- in above example, first statement imports only date function while the second statement imports the whole datetime module

- However , python loads the whole module in both the cases.
- The only difference is that the , The first statement creates a var that references the date function from the datetime module
- while the second statement creates a var that references the datetime module object.

#PYTHONTIPS

THINGS TO KEEP IN MIND !

- The Python import statement loads a module only once and caches it in the sys.modules.
- Avoid using the
  from module import *
  as it might introduce some bug due to refrences
- Do not use
  from module import object
  to optimize the program speed because it won't !

@VEDANTSOLANKI

@TAG

SOMEONE WHO WILL FIND THIS HELPFUL

FOLLOW FOR MORE !

Vedant Solanki

@vedantsolanki