NONETYPE

```
type(None)
# NoneType

# Python <=2.7
from types import NoneType

# Python 3
from types import NoneType
ImportError: cannot import name 'NoneType' from 'types'

```

```python
# Python <2.2
from types import IntType
if type(item) is IntType: ...

# Python >=2.2
if isinstance(item, int): ...
```

SIMPLENAMESPACE

```
 1 from types import SimpleNamespace
 2 data = SimpleNamespace(a=1, b=2)
 3 data
 4 # namespace(a=1, b=2)
 5
 6 data.a
 7 # 1
 8
 9 data.c = 3
10 data
11 # namespace(a=1, b=2, c=3)
12
13 data['a']
14 TypeError: 'types.SimpleNamespace' object is not subscriptable
```

```
1 SimpleNamespace = type(sys.implementation)
```

```
from types import MappingProxyType
data = {'a': 1, 'b': 2}
read_only = MappingProxyType(data)
read_only
# mappingproxy({'a': 1, 'b': 2})

del read_only['a']
TypeError: 'mappingproxy' object does not support item deletion

read_only['a'] = 3
TypeError: 'mappingproxy' object does not support item assignment

data['a'] = 6
# mappingproxy({'a': 6, 'b': 2})

```

CHAINMAP

```python
from collections import ChainMap

a = {1: 2, 3: 4}
b = {5: 6, 7: 8}
chain = ChainMap(a, b)
chain
# ChainMap({1: 2, 3: 4}, {5: 6, 7: 8})

chain[5]
# 6
chain[1]
# 2

a[9] = 10
chain
# ChainMap({1: 2, 3: 4, 9: 10}, {5: 6, 7: 8})
```

SIZED

```python
from typing import Sized

isinstance(range(10), Sized)
# True

isinstance((i for i in range(10)), Sized)
# False

```

NAMEDTUPLE

```python
from typing import NamedTuple

class User(NamedTuple):
    name: str
    age: int
    hometown: str = 'SpB'

user = User('Gram', 23)
user
# User(name='Gram', age=23, hometown='SpB')

```

```python
from dataclasses import dataclass

@dataclass(order=True, frozen=True)
class User:
    name: str
    age: int
    hometown: str = 'SpB'

user = User('Gram', 23)
user
# User(name='Gram', age=23, hometown='SpB')

```

```python
tuple(user)
# ('Gram', 23, 'SpB')

user.name
# 'Gram'

user[1]
# 23

user._asdict()
# OrderedDict([('name', 'Gram'), ('age', 23), ('hometown', 'SpB')])
```

```python
from decimal import Decimal

Amount = Decimal
UserName = str


def get_debt(user: UserName) -> Amount:
    ...

```