VARIABLES IN PYTHON

TABLE OF CONTENTS

1. Variable Assignment
2. Object References
3. Object Identity
4. Variable Names
5. Reserved Keywords

TABLE OF CONTENTS

▶ 1. **Variable Assignment**

2. Object References

3. Object Identity

4. Variable Names

5. Reserved Keywords

VARIABLE ASSIGNMENT

VARIABLE ASSIGNMENT

- *Preparation: Using the iPython Shell*
- Standard Variable Assignment
- Chained Assignment
- Multiple Assignment
- Variable Types

VARIABLE ASSIGNMENT

- *Preparation: Using the iPython Shell*
- **Standard Variable Assignment**
- Chained Assignment
- Multiple Assignment
- Variable Types

```
n = 300
```

VARIABLE ASSIGNMENT

- *Preparation: Using the iPython Shell*
- Standard Variable Assignment
- **Chained Assignment**
- Multiple Assignment
- Variable Types

```
n = m = 300
```

VARIABLE ASSIGNMENT

- *Preparation: Using the iPython Shell*
- Standard Variable Assignment
- Chained Assignment
- **Multiple Assignment**
- Variable Types

```
n, m = 300, 400
```

VARIABLE ASSIGNMENT

- *Preparation: Using the iPython Shell*
- Standard Variable Assignment
- Chained Assignment
- Multiple Assignment
- **Variable Types**

```
>>> n = 300
>>> type(n)
int
>>> type(300)
int
```

TABLE OF CONTENTS

1. Variable Assignment
2. ▶ **Object References**
3. Object Identity
4. Variable Names
5. Reserved Keywords

OBJECT REFERENCES

OBJECT REFERENCES

- Everything is an Object
- Variables are References
- Orphaned Objects

Everything is an Object

```
"hello, world"
```

Everything is an Object

```
"hello, world"
```

```
42
```

Everything is an Object

```
"hello, world"
```

```
42
```

Everything is an Object

```
MyObject
```

`"hello, world"`

`42`

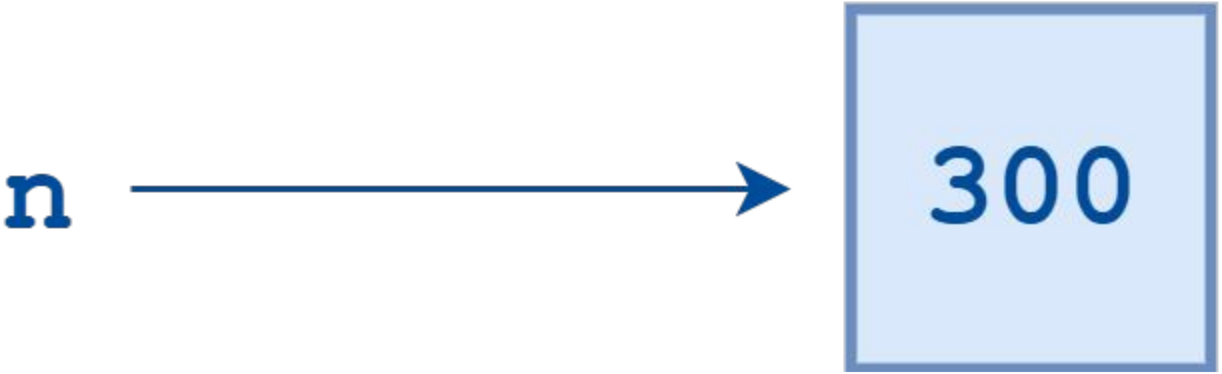
Everything is an Object

`MyObject`

`range`

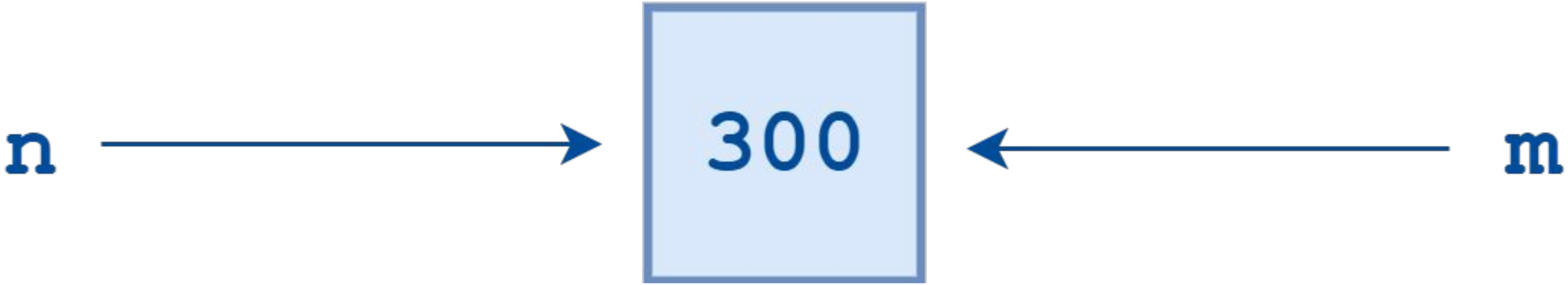
```python
n = 300
```

```python
n = 300
```



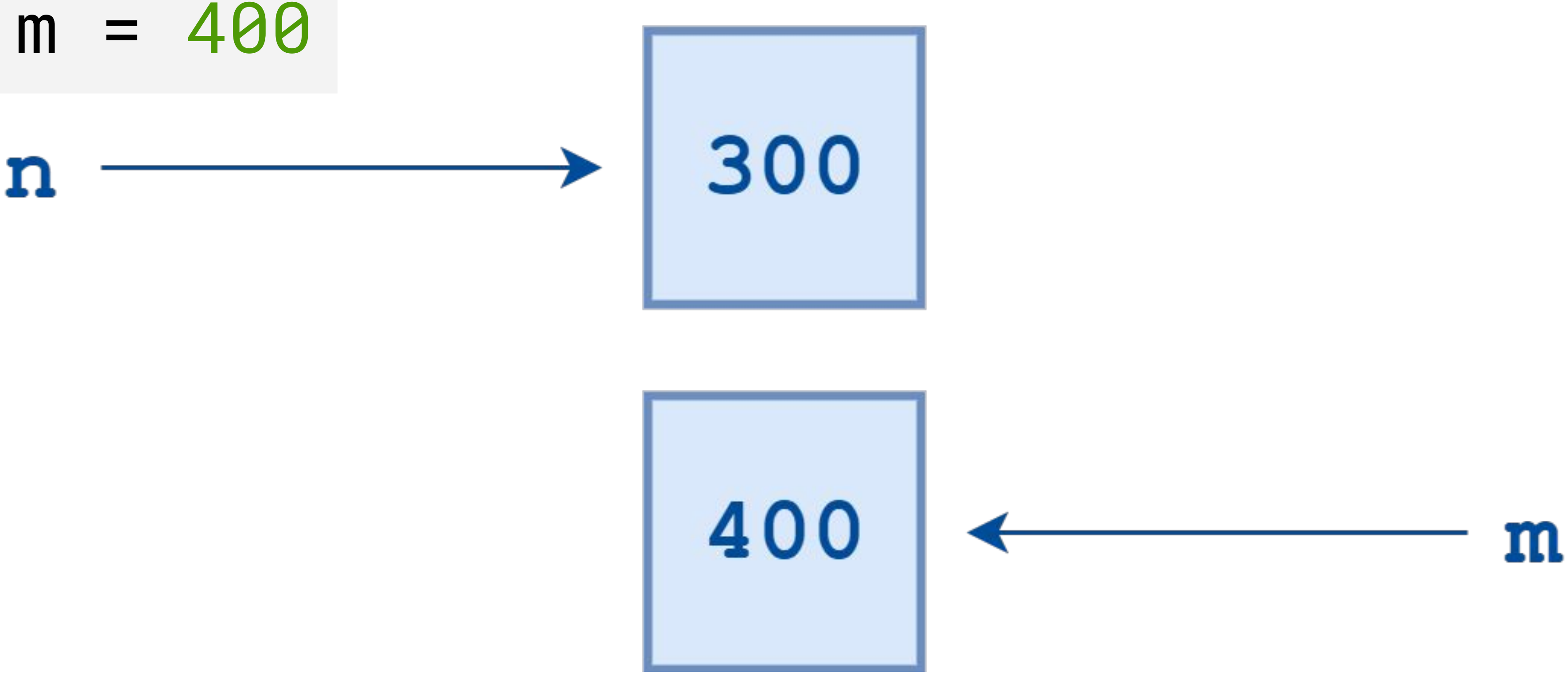
```
m = n
```

```
m = n
```

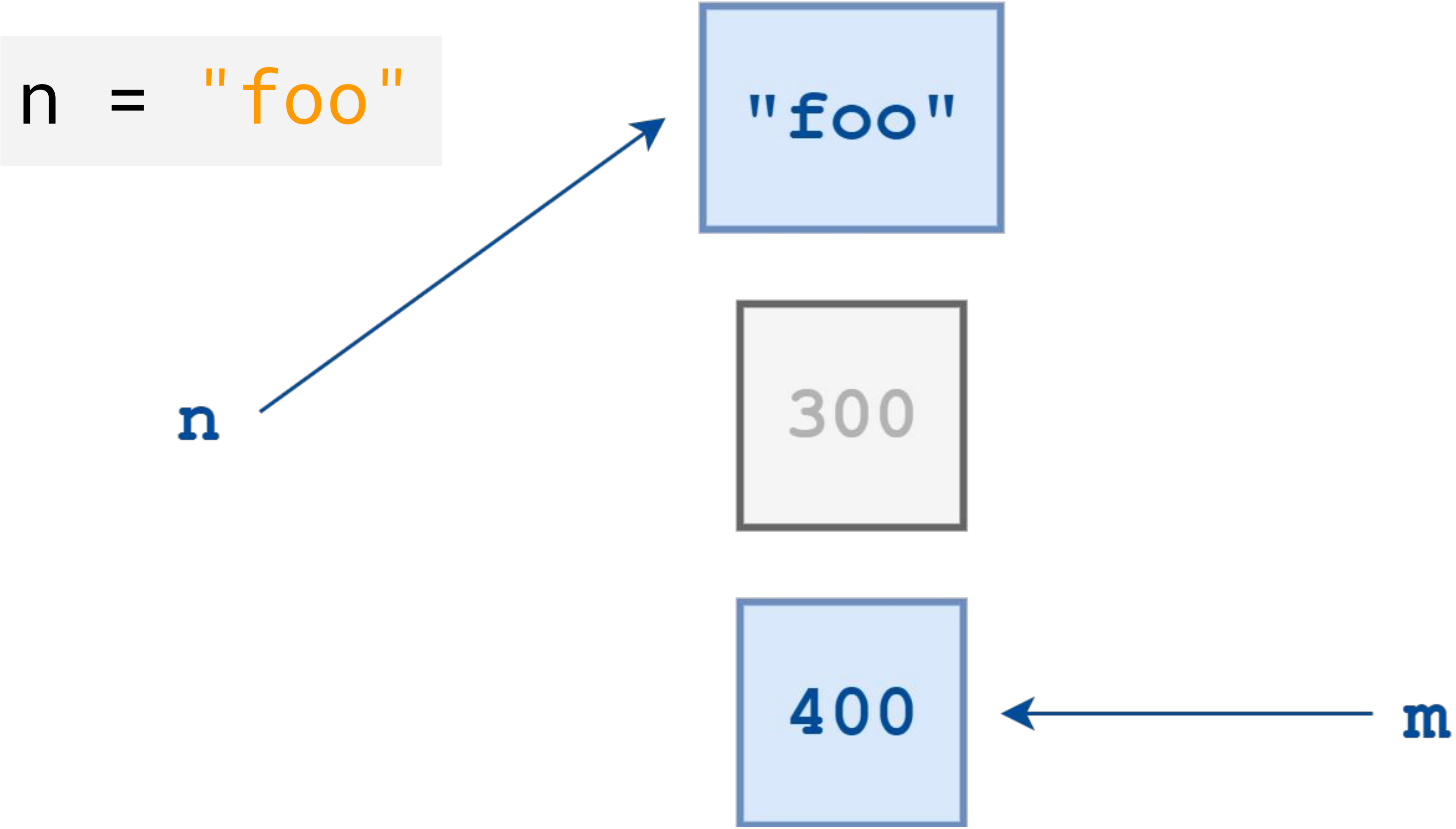


```
m = 400
```

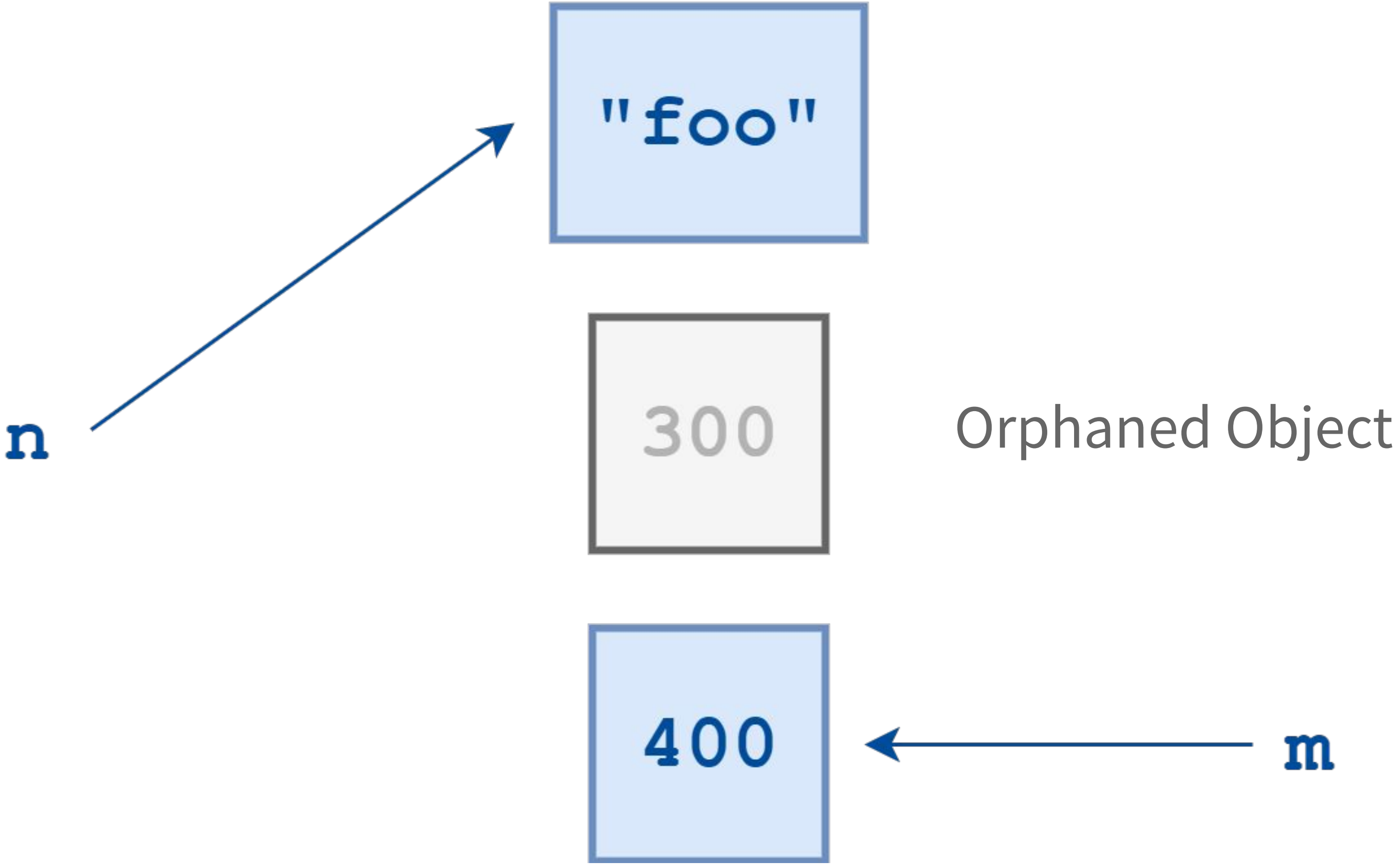
```
m = 400
```



```
n = "foo"
```



```python
n = "foo"
```



Real Python

Garbage Collection

OBJECT REFERENCES

- Everything is an Object
- Variables are References
- Orphaned Objects

OBJECT REFERENCES

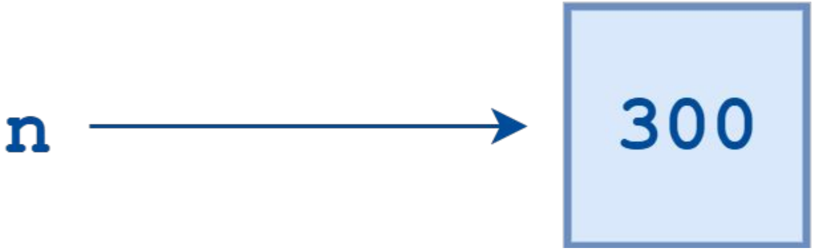
- **Everything is an Object**
- Variables are References
- Orphaned Objects

`"hello, world"` `42` `MyObject` `range`

OBJECT REFERENCES

- Everything is an Object
- **Variables are References**
- Orphaned Objects

```
n = 300
```

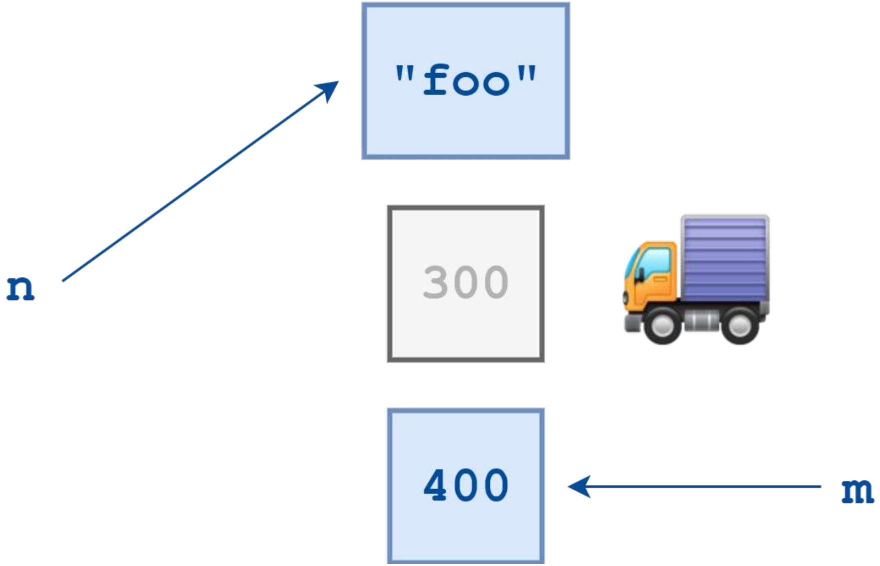


n

300

OBJECT REFERENCES

- Everything is an Object
- Variables are References
- **Orphaned Objects**



OBJECT IDENTITY

TABLE OF CONTENTS

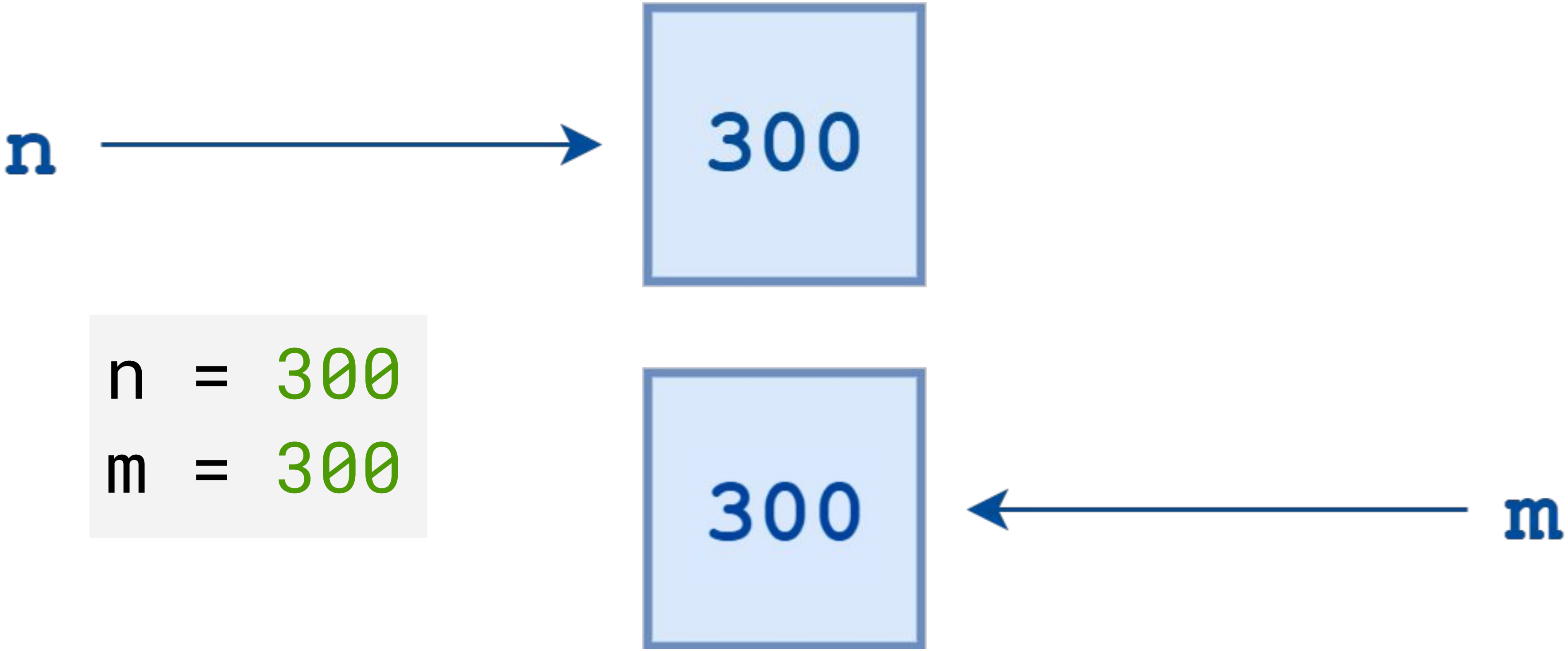
1. Variable Assignment
2. Object References
3. **Object Identity**
4. Variable Names
5. Reserved Keywords

OBJECT IDENTITY

- Object Value vs. Object Identity
- Small Integer Caching
- *Challenge: Python Pub Quiz*

Python Caches Small Integers!

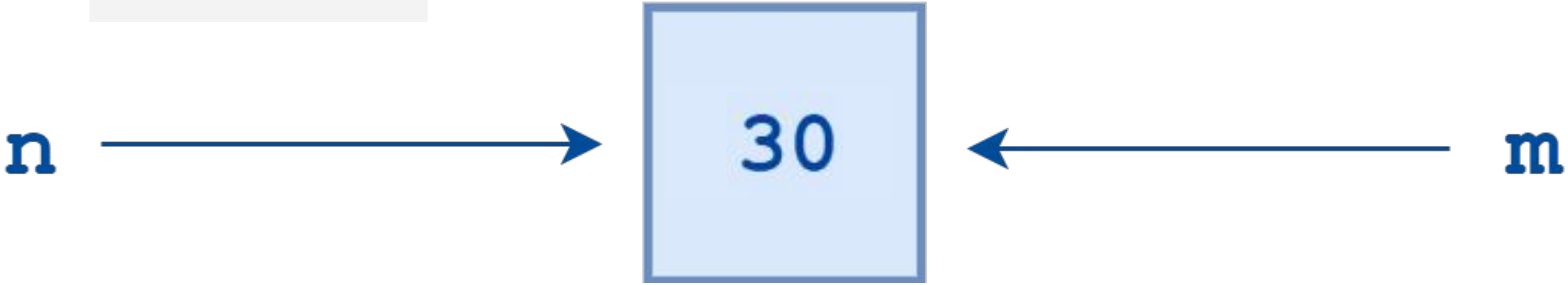
```
n = 300
m = 300
```



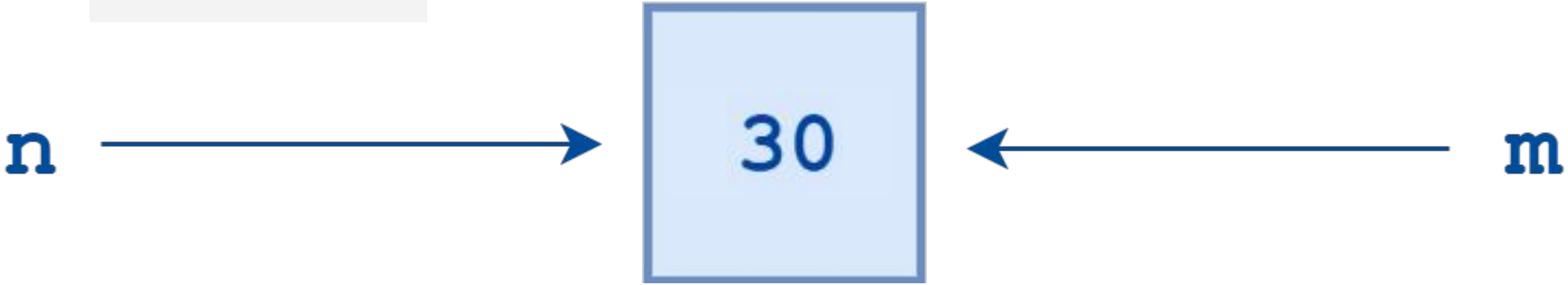
```
n = 300
m = 300
```

```
n = 30
m = 30
```

```
n = 30
m = 30
```



```python
n = 30
m = 30
```



n

30

m

```python
m = n
```

Small Integer Caching

```
[-5] [-4] [...] [255] [256]
```

Python Pub Quiz Question

```python
a, b = 250, 250

for _ in range(250, 260):
    if a is not b:
        break
    a += 1
    b += 1

print(a)
```

OBJECT IDENTITY

- Object Value vs. Object Identity
- Small Integer Caching
- *Challenge: Python Pub Quiz*

OBJECT IDENTITY

- **Object Value vs. Object Identity**
- Small Integer Caching
- *Challenge: Python Pub Quiz*

OBJECT IDENTITY

- Object Value vs. Object Identity
- **Small Integer Caching**
- *Challenge: Python Pub Quiz*

```
[-5] [-4] [...] [255] [256]
```

OBJECT IDENTITY

- Object Value vs. Object Identity
- Small Integer Caching
- ***Challenge: Python Pub Quiz***

Python Pub Quiz Question

```python
a, b = 250, 250

for _ in range(250, 260):
    if a is not b:
        break
    a += 1
    b += 1

print(a)
```

VARIABLE NAMES

TABLE OF CONTENTS

1. Variable Assignment
2. Object References
3. Object Identity
4. ▶ **Variable Names**
5. Reserved Keywords

VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8

VARIABLE NAMING CONVENTIONS

- Any length

VARIABLE NAMING CONVENTIONS

- Any length
- Upper- and lowercase characters

VARIABLE NAMING CONVENTIONS

- Any length
- Upper- and lowercase characters
- Underscores

VARIABLE NAMING CONVENTIONS

- Any length
- Upper- and lowercase characters
- Underscores
- Digits (not at the beginning)

VARIABLE NAMING CONVENTIONS

- Any length
- Upper- and lowercase characters
- Underscores
- Digits (not at the beginning)
- Unicode

VARIABLE NAMES

- Naming Conventions
- **Hello PEP-8**

RECAP: VARIABLE NAMES

- Naming Conventions
- Hello PEP-8

VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8

```
this_is_a_super_long_variable_name_but_hey_its_all_good = 300
n = 300
```

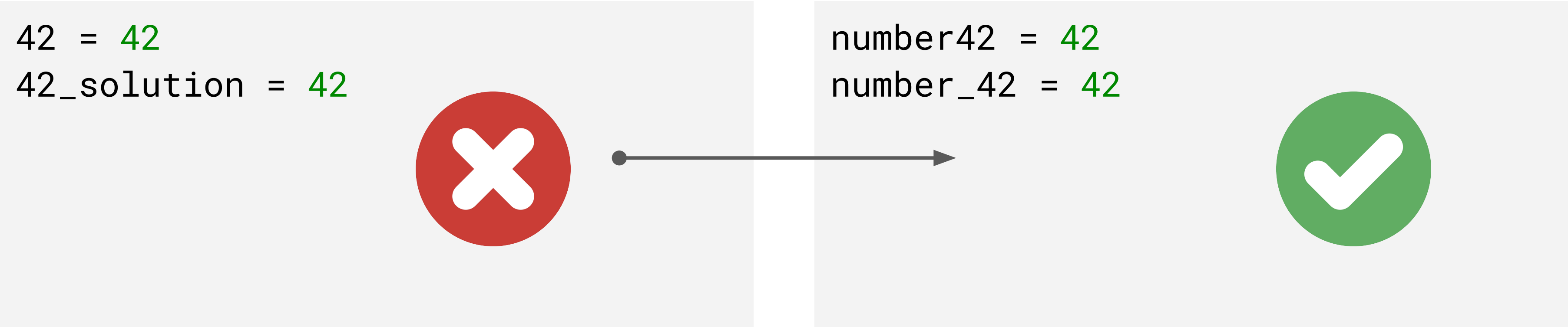
VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8

```
voilà = "this works in Python 3"
```

VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8



```
42 = 42
42_solution = 42
```

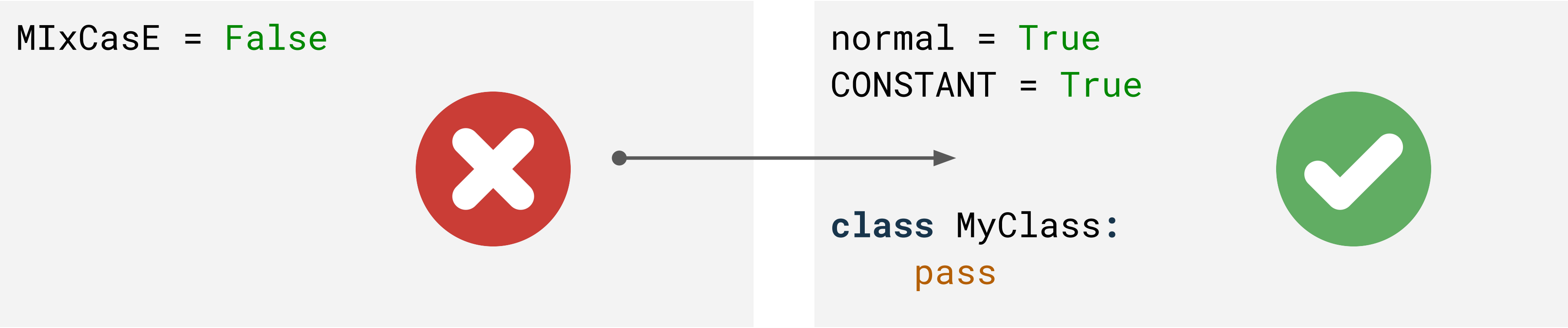
✗

```
number42 = 42
number_42 = 42
```

✓

VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8



```python
MIxCasE = False
```

✗

```python
normal = True
CONSTANT = True


class MyClass:
    pass
```

✓

VARIABLE NAMES

- **Naming Conventions**
- Hello PEP-8

```
a_good_variable = "looks like this!"
```

snake_case

VARIABLE NAMES

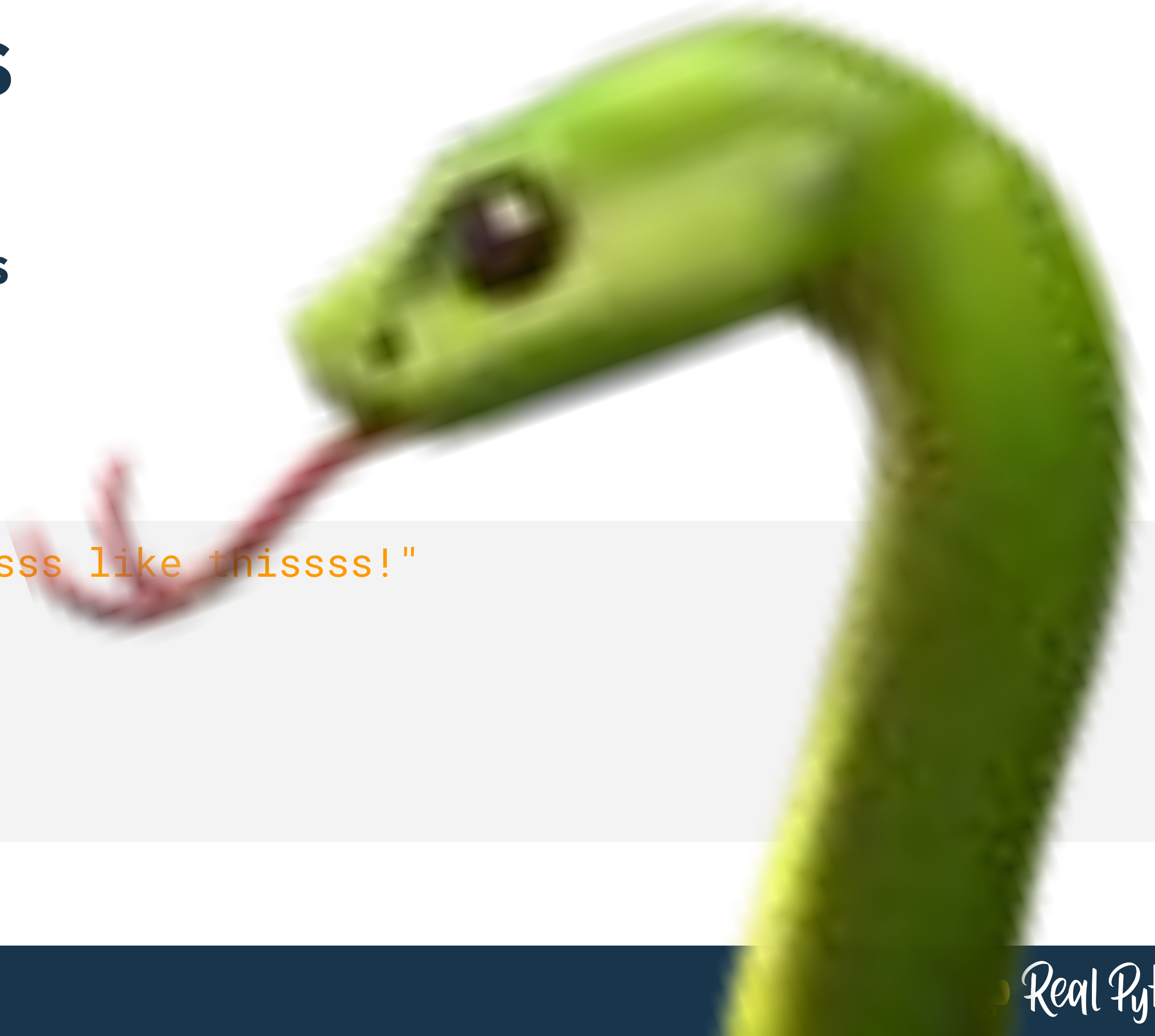
- **Naming Conventions**
- Hello PEP-8

```
a_good_variable = "lookss like thiss!"
```

VARIABLE NAMES

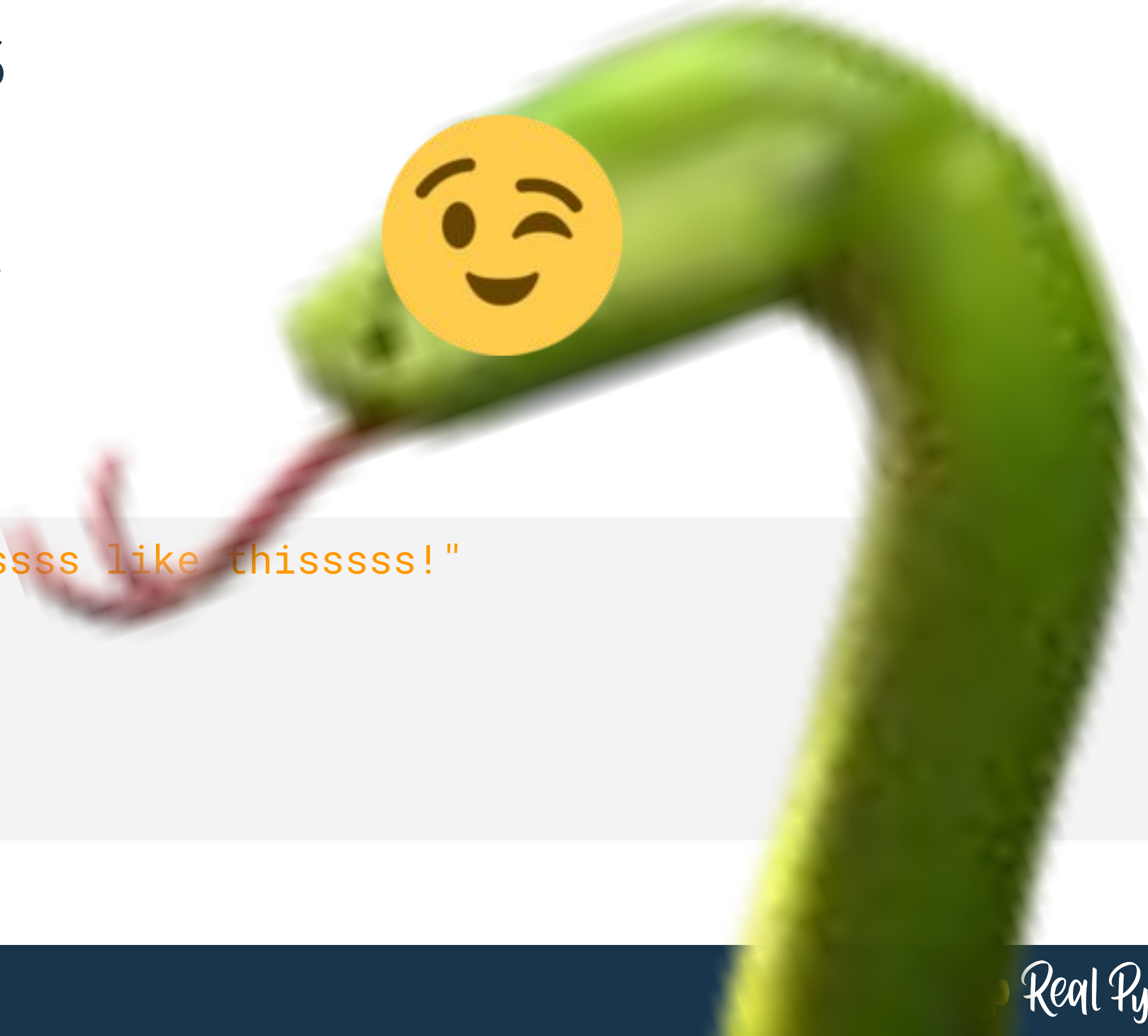
- **Naming Conventions**
- Hello PEP-8

```python
a_good_variable = "looksssss like thissssss!"
```



VARIABLE NAMES

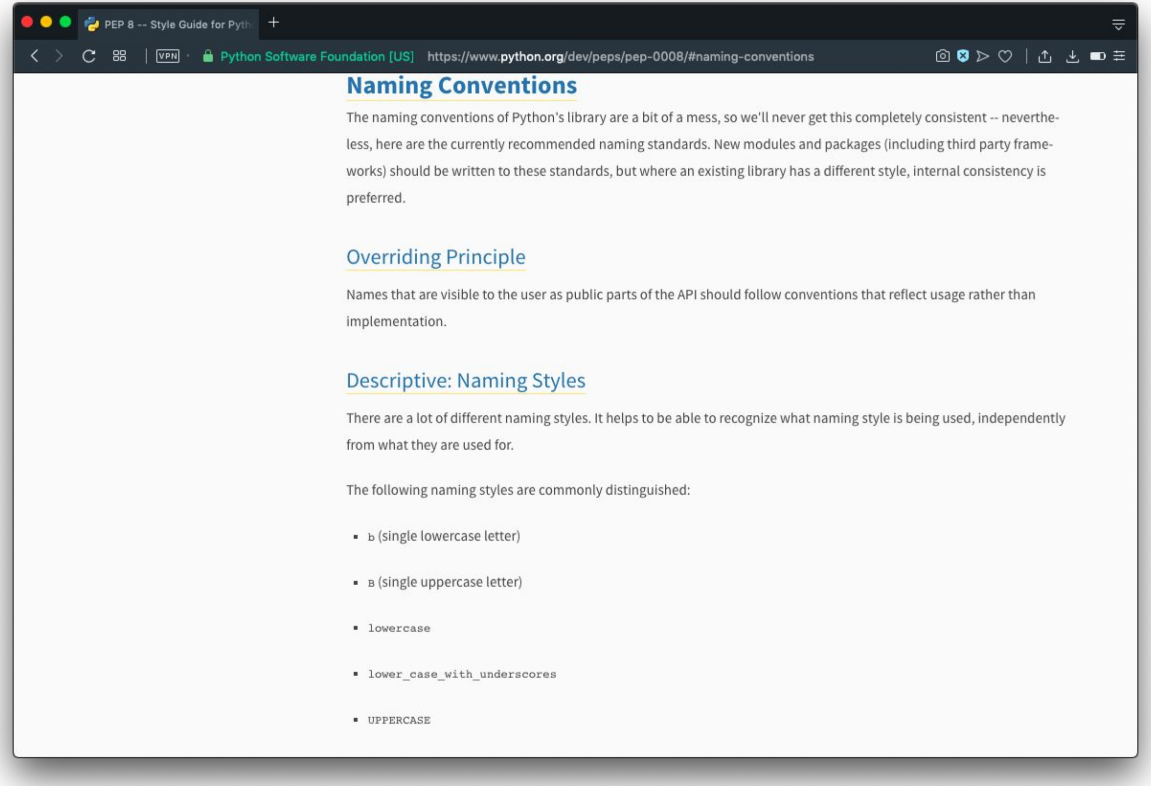
- **Naming Conventions**
- Hello PEP-8



```
a_good_variable = "looksssss like thisssss!"
```

VARIABLE NAMES

- Naming Conventions
- **Hello PEP-8**



PEP 8 -- Style Guide for Pyth
VPN
Python Software Foundation [US]
https://www.python.org/dev/peps/pep-0008/#naming-conventions

Naming Conventions

The naming conventions of Python's library are a bit of a mess, so we'll never get this completely consistent -- nevertheless, here are the currently recommended naming standards. New modules and packages (including third party frameworks) should be written to these standards, but where an existing library has a different style, internal consistency is preferred.

Overriding Principle

Names that are visible to the user as public parts of the API should follow conventions that reflect usage rather than implementation.

Descriptive: Naming Styles

There are a lot of different naming styles. It helps to be able to recognize what naming style is being used, independently from what they are used for.

The following naming styles are commonly distinguished:

- `b` (single lowercase letter)
- `B` (single uppercase letter)
- `lowercase`
- `lower_case_with_underscores`
- `UPPERCASE`

TABLE OF CONTENTS

1. Variable Assignment
2. Object References
3. Object Identity
4. Variable Names
5. ▶ **Reserved Keywords**

RESERVED KEYWORDS

RESERVED KEYWORDS

- Special Words
- Syntax Error
- Getting help()

What Keywords Exist?

```
help("keywords")
```

RESERVED KEYWORDS

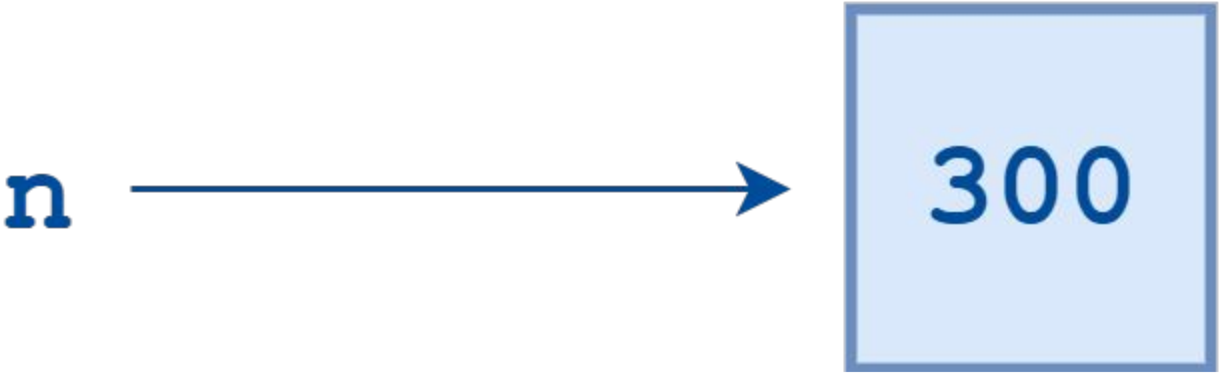
- Special Words
- Syntax Error
- Getting help()

TABLE OF CONTENTS

1. Variable Assignment
2. Object References
3. Object Identity
4. Variable Names
5. Reserved Keywords

CONCLUSION

```python
n = 300
```



`"hello, world"`

`42`

Everything is an Object

`MyObject`

`range`

Python Caches Small Integers!

Python Caches Small Integers!

+ Challenge! 🚀

Variable Naming Conventions

- Any length
- Upper- and lowercase characters
- Underscores
- Digits (not at the beginning)
- Unicode

Reserved Keywords

```python
help("keywords")
```