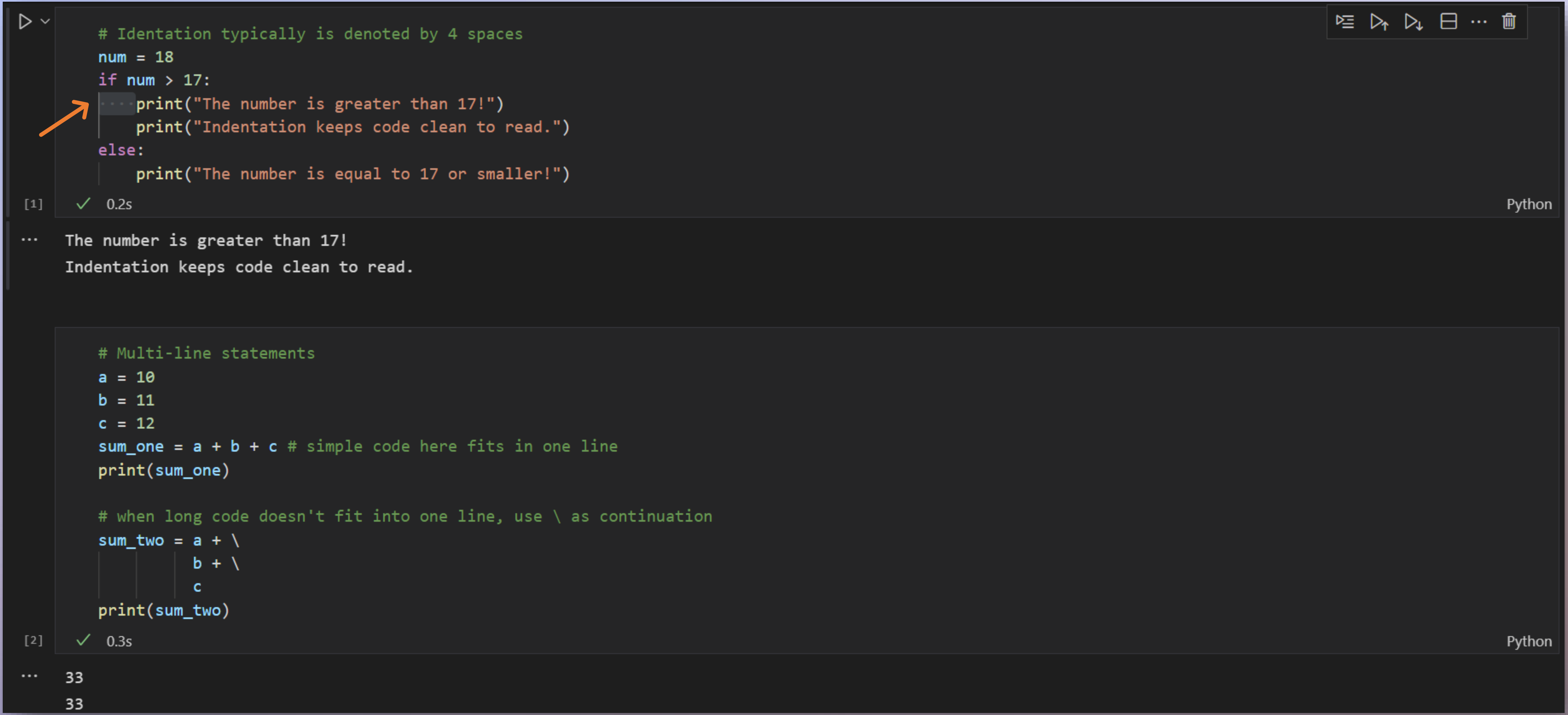
Basic Syntax

- **Indentation** is used to indicate block of code.
- **Multi-line statements** to denote continuation of code and uses \ (backward slash).
- **Identifiers** are names used to assign variables, functions, classes, or other objects.
- **Reserved words** are Python specific and **cannot** be used as identifiers and are mostly lowercase.
- See **PEP 8** – Style Guide for Python Code.

https://peps.python.org/pep-0008/

Syntax – Indentation & Multi-line



```python
# Identation typically is denoted by 4 spaces
num = 18
if num > 17:
    print("The number is greater than 17!")
    print("Indentation keeps code clean to read.")
else:
    print("The number is equal to 17 or smaller!")
```

[1] ✓ 0.2s Python

```
The number is greater than 17!
Indentation keeps code clean to read.
```

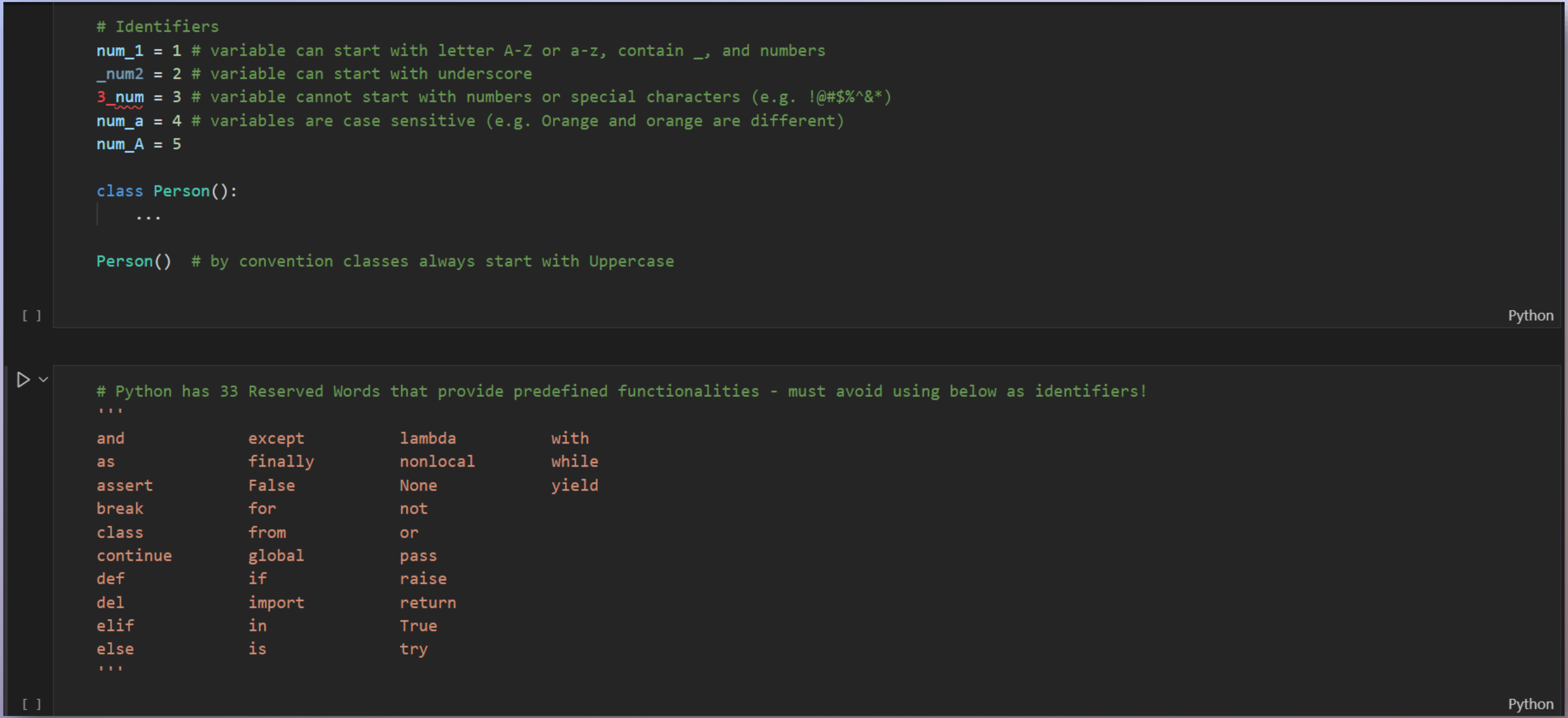
```python
# Multi-line statements
a = 10
b = 11
c = 12
sum_one = a + b + c # simple code here fits in one line
print(sum_one)

# when long code doesn't fit into one line, use \ as continuation
sum_two = a + \
          b + \
          c
print(sum_two)
```

[2] ✓ 0.3s Python

```
33
33
```

Syntax – Identifiers & Reserved Words



```python
# Identifiers
num_1 = 1 # variable can start with letter A-Z or a-z, contain _, and numbers
_num2 = 2 # variable can start with underscore
3_num = 3 # variable cannot start with numbers or special characters (e.g. !@#$%^&*)
num_a = 4 # variables are case sensitive (e.g. Orange and orange are different)
num_A = 5

class Person():
    ...

Person()  # by convention classes always start with Uppercase
```

[ ] Python

```python
# Python has 33 Reserved Words that provide predefined functionalities - must avoid using below as identifiers!
'''
and             except          lambda          with
as              finally         nonlocal        while
assert          False           None            yield
break           for             not
class           from            or
continue        global          pass
def             if              raise
del             import          return
elif            in              True
else            is              try
'''
```

[ ] Python