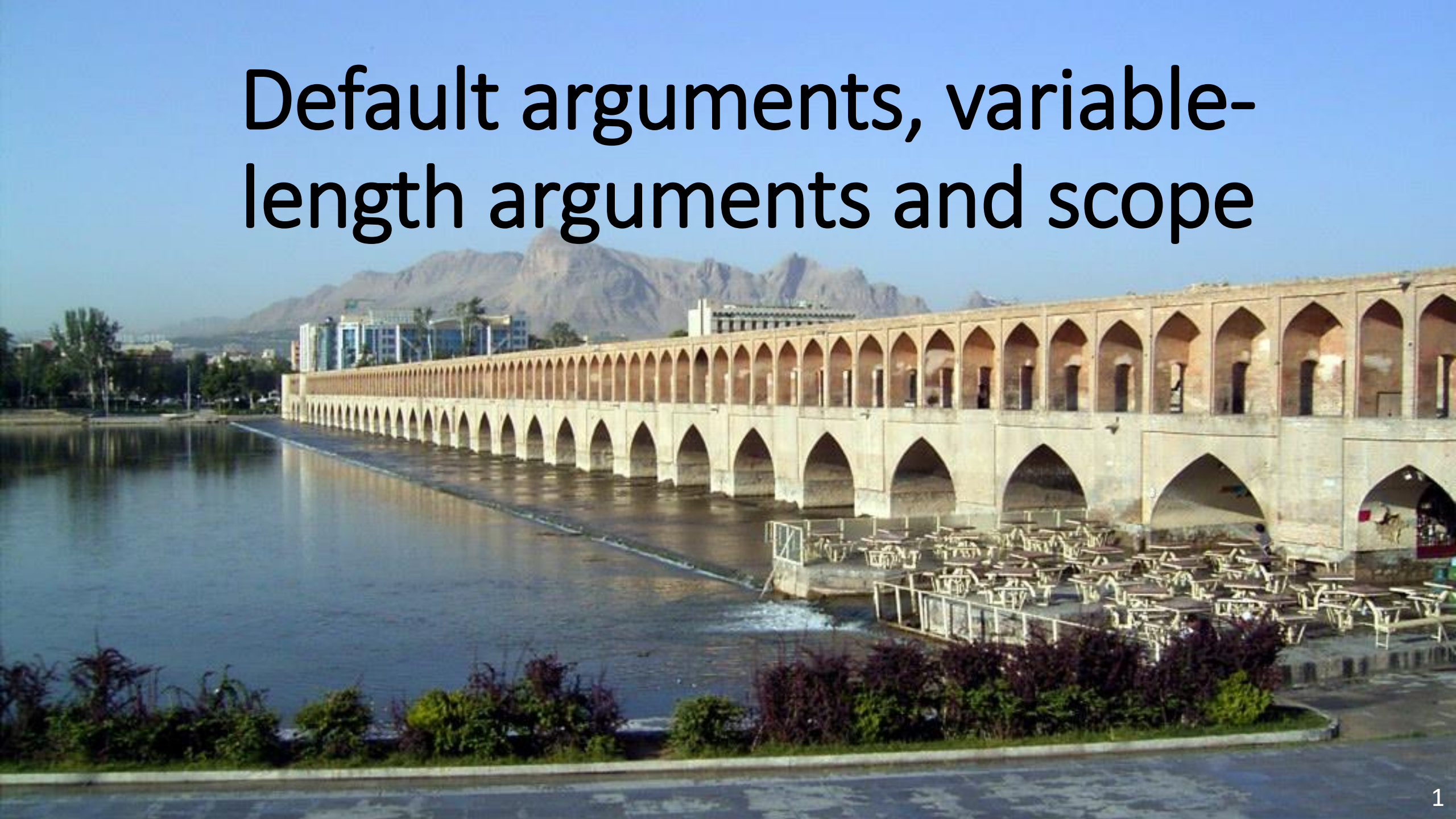
Default arguments, variable-length arguments and scope



Crash course on scope in functions

- Not all objects are accessible everywhere in a script
- Scope - part of the program where an object or name may be accessible
  - ✓ Global scope - defined in the main body of a script
  - ✓ Local scope - defined inside a function
  - ✓ Built-in scope - names in the pre-defined built-ins module

Global vs. local scope (1)

```
def square(value):
    """Returns the square of a number."""
    new_val = value ** 2
    return new_val
square(3)
```

```
9
```

```
new_val
```

```
<hr />---------------------------------------------------------------
NameError                          Traceback (most recent call last)
<ipython-input-3-3cc6c6de5c5c> in <module>()
<hr />-> 1 new_value
NameError: name 'new_val' is not defined
```

Global vs. local scope (2)

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    new_val = value ** 2
    return new_val
square(3)
```

```
9
```

```
new_val
```

```
10
```

Global vs. local scope (3)

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    new_value2 = new_val ** 2
    return new_value2
square(3)
```

```
100
```

```
new_val = 20

square(3)
```

```
400
```

Global vs. local scope (4)

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    global new_val
    new_val = new_val ** 2
    return new_val
square(3)
```

```
100
```

```
new_val
```

```
100
```

Nested functions

Nested functions (1)

```
def outer( ... ):
    """ ... """
    x = ...

    def inner( ... ):
        """ ... """
        y = x ** 2
    return ...
```

Nested functions (2)

```
def mod2plus5(x1, x2, x3):
    """Returns the remainder plus 5 of three values."""

    new_x1 = x1 % 2 + 5
    new_x2 = x2 % 2 + 5
    new_x3 = x3 % 2 + 5

    return (new_x1, new_x2, new_x3)
```

Nested functions (3)

```
def mod2plus5(x1, x2, x3):
    """Returns the remainder plus 5 of three values."""

    def inner(x):
        """Returns the remainder plus 5 of a value."""
        return x % 2 + 5

    return (inner(x1), inner(x2), inner(x3))
```

```
print(mod2plus5(1, 2, 3))
```

```
(6, 5, 6)
```

Returning functions

```
def raise_val(n):
    """Return the inner function."""

    def inner(x):
    """Raise x to the power of n."""
        raised = x ** n
        return raised

    return inner
```

```
square = raise_val(2)
cube = raise_val(3)
print(square(2), cube(4))
```

```
4 64
```

Default and flexible arguments

You'll learn:

- Writing functions with default arguments
- Using flexible arguments
  - ✓ Pass any number of arguments to a functions

Add a default argument

```
def power(number, pow=1):
    """Raise number to the power of pow."""
    new_value = number ** pow
    return new_value

power(9, 2)
```

```
81
```

```
power(9, 1)
```

```
9
```

```
power(9)
```

```
9
```

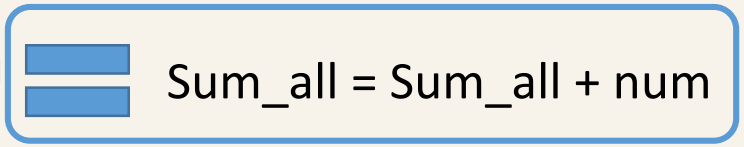
Flexible arguments: *args (1)

```
def add_all(*args):
    """Sum all values in *args together."""

    # Initialize sum
    sum_all = 0

    # Accumulate the sum
    for num in args:
        sum_all += num

    return sum_all
```



Flexible arguments: *args (2)

```
add_all(1)
```

```
1
```

```
add_all(1, 2)
```

```
3
```

```
add_all(5, 10, 15, 20)
```

```
50
```

Let's practice!