# Day 14: Python Functions

Method overloading is not supported in Python.

```
def greeting(name):
    print(f"Hello, {name}")

def greeting(name, age):
    print(f"Hello, {name}. You are {age} years old.")

greet("Alice")  # ❌ This will cause an error (missing 'age')
```

## Positional-Only Arguments

You can specify that a function can have ONLY positional arguments, or ONLY keyword arguments.

To specify that a function can have only positional arguments, add `, /` after the arguments:

```
def my_function(x, /):
    print(x)

my_function(3)
my_function(x = 3) # This will give error
```

## Keyword-Only Arguments

To specify that a function can have only keyword arguments, add `*,` *before* the arguments:

```
def my_function(*, x):
  print(x)
```

```
my_function(x = 3)
my_function(3) # This will give error
```

## Combine Positional-Only and Keyword-Only

Any argument *before* the `/,` are positional-only, and any argument *after* the `*,` are keyword-only.

```
def my_function(a, b, /, *, c, d):
  print(a + b + c + d)

my_function(5, 6, c = 7, d = 8)
```

## Recursion

Python also accepts function recursion, which means a defined function can call itself.

It is similar to nested-loop.

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
  else:
    result = 0
  return result

print("Recursion Example Results:")
tri_recursion(6)
```