

Memory Model

- The thing to look at in the first set is the growth and shrinking of the stack, and how variable names refer to locations in memory.
- Local namespaces are coloured, while the global namespace is not.

Memory Model

- In this example, also note how arguments refer to variables right after we've called a function, but before we've executed any lines of code.

Memory Model

Memory

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    -----> Where we are.  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Stack

p	
r	
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    ----->  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

Stack

p	0x1
r	
q	

Memory Model

```
def f(x):  
    ----->  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

Stack

z	
y	
x	0x1
p	0x1
r	
q	

Memory Model

```
def f(x):  
    y=5  
    ----->  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

int	5
0x2	

Stack

z	
y	0x2
x	0x1
p	0x1
r	
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    ----->  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

int	5
0x2	

int	65
0x3	

Stack

z	0x3
y	0x2
x	0x1
p	0x1
r	
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    ----->  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

int	65
0x3	

Stack

p	0x1
r	0x3
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    ----->  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

int	65
0x3	

Stack

v	
w	0x1
p	0x1
r	0x3
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    ----->  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    print p, q, r
```

Memory

int	13
0x1	

int	6
0x4	

int	65
0x3	

Stack

v	0x4
w	0x1
p	0x1
r	0x3
q	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(w):  
    v=6  
    return w+4
```

```
if __name__ == "__main__":  
    p = 13  
    r = f(p)  
    q = g(p)  
    ----->  
    print p, q, r
```

Memory

int	13
0x1	

int	17
0x5	

int	65
0x3	

Stack

p	0x1
r	0x3
q	0x5

Memory Model

- Now we switch from distinct variable names in each function, to reusing variable names in each function.
- Note that the only difference in the diagram is the names of the variables. They all still point to the same locations in memory.

Memory Model

Memory

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    -----> Where we are.  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Stack

x	
y	
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    ----->  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

Stack

x	0x1
y	
z	

Memory Model

```
def f(x):  
    ----->  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

Stack

z	
y	
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=5  
    ----->  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	5
0x2	

Stack

z	
y	0x2
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    ----->  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	5
0x2	

int	65
0x3	

Stack

z	0x3
y	0x2
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    ----->  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	65
0x3	

Stack

x	0x1
y	0x3
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    ----->  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	65
0x3	

Stack

x	0x1
y	
x	0x1
y	0x3
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    ----->  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	6
0x4	

int	65
0x3	

Stack

x	0x1
y	0x4
x	0x1
y	0x3
z	

Memory Model

```
def f(x):  
    y=5  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    ----->  
    print x, y, z
```

Memory

int	13
0x1	

int	17
0x5	

int	65
0x3	

Stack

x	0x1
y	0x3
z	0x5

Memory Model

- Now we have a function calling a function.
- Note how the stack grows and decreases.

Memory Model

Memory

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
-----> Where we are.  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Stack

x	
y	
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    ----->  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

Stack

x	0x1
y	
z	

Memory Model

```
def f(x):  
    ----->  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

Stack

z	
y	
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    ----->  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	5
0x2	

x	0x2
y	
z	
y	
x	0x1
x	0x1
y	
z	

Stack

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    ----->  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	5
0x2	

int	6
0x3	

x	0x2
y	0x3
z	
y	
x	0x1
x	0x1
y	
z	

Stack

Memory Model

```
def f(x):  
    y=g(5)  
    ----->  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	9
0x4	

Stack

z	
y	0x4
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    ----->  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	9
0x4	

int	117
0x5	

Stack

z	
y	0x4
x	0x1
x	0x1
y	
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    ----->  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	117
0x5	

Stack

x	0x1
y	0x5
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    ----->  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	117
0x5	

Stack

x	0x1
y	
x	0x1
y	0x5
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    ----->  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    print x, y, z
```

Memory

int	13
0x1	

int	6
0x6	

int	117
0x5	

Stack

x	0x1
y	0x6
x	0x1
y	0x5
z	

Memory Model

```
def f(x):  
    y=g(5)  
    z= x*y  
    return z
```

```
def g(x):  
    y=6  
    return x+4
```

```
if __name__ == "__main__":  
    x = 13  
    y = f(x)  
    z = g(x)  
    ----->  
    print x, y, z
```

Memory

int	13
0x1	

int	17
0x7	

int	117
0x5	

Stack

x	0x1
y	0x5
z	0x5