Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4.
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9.
10. f1(5)
11. print('bye')
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4. 
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9. 
10. f1(5)
11. print('bye')

• We start with line 10
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4.
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9.
10. f1(5)
11. print('bye')

• We start with line 10
• Line 10 contains a call to f1, so we move execution to line 5
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4. 
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9. 
10. f1(5)
11. print('bye')

- We start with line 10
- Line 10 contains a call to f1, so we move execution to line 5
- We copy 5 to local variable n (line 5) in f1
Let’s trace some code!

1. `def f2(n):`
2. `n = n*10`
3. `print(n)`
4. 
5. `def f1(n):`
6. `k = n+1`
7. `f2(k)`
8. `print(n,k)`
9. 
10. `f1(5)`
11. `print('bye')`

- We start with line 10
- Line 10 contains a call to `f1`, so we move execution to line 5
- We copy 5 to local variable `n` (line 5) in `f1`
- Create variable `k` and assign it a value of 6
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4. 
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9. 
10. f1(5)
11. print('bye')

- We start with line 10
- Line 10 contains a call to f1, so we move execution to line 5
- We copy 5 to local variable n (line 5) in f1
- Create variable k and assign it a value of 6
- Line 7 contains a call to f2, so we move execution to line 1
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4.     
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9.     
10. f1(5)
11. print('bye')

• We start with line 10
• Line 10 contains a call to f1, so we move execution to line 5
• We copy 5 to local variable n (line 5) in f1
• Create variable k and assign it a value of 6
• Line 7 contains a call to f2, so we move execution to line 1
• We copy the value of 6 to local variable n (line 1) in f2
Let’s trace some code!

1. ```python
def f2(n):
2.     n = n*10
3.     print(n)
```

4. 
5. ```python
def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
```

9. 
10. ```python
f1(5)
```
11. ```python
print('bye')
```

• We start with line 10
• Line 10 contains a call to f1, so we move execution to line 5
• We copy 5 to local variable n (line 5) in f1
• Create variable k and assign it a value of 6
• Line 7 contains a call to f2, so we move execution to line 1
• We copy the value of 6 to local variable n (line 1) in f2
• We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4. 
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9. 
10. f1(5)
11. print('bye')

• We start with line 10
• Line 10 contains a call to f1, so we move execution to line 5
• We copy 5 to local variable n (line 5) in f1
• Create variable k and assign it a value of 6
• Line 7 contains a call to f2, so we move execution to line 1
• We copy the value of 6 to local variable n (line 1) in f2
• We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
• We print 60
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4.
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9.
10. f1(5)
11. print('bye')

- We start with line 10
- Line 10 contains a call to f1, so we move execution to line 5
- We copy 5 to local variable n (line 5) in f1
- Create variable k and assign it a value of 6
- Line 7 contains a call to f2, so we move execution to line 1
- We copy the value of 6 to local variable n (line 1) in f2
- We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
- We print 60
- f2 is done, what do we do next?
Let’s trace some code!

1. `def f2(n):`
2. `n = n*10`
3. `print(n)`
4. 
5. `def f1(n):`
6. `k = n+1`
7. `f2(k)`
8. `print(n,k)`
9. 
10. `f1(5)`
11. `print('bye')`

- We start with line 10
- Line 10 contains a call to f1, so we move execution to line 5
- We copy 5 to local variable n (line 5) in f1
- Create variable k and assign it a value of 6
- Line 7 contains a call to f2, so we move execution to line 1
- We copy the value of 6 to local variable n (line 1) in f2
- We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
- We print 60
- f2 is done, **what do we do next?**
Let’s trace some code!

1. def f2(n):
2.    n = n*10
3.    print(n)
4. 
5. def f1(n):
6.    k = n+1
7.    f2(k)
8.    print(n,k)
9. 
10. f1(5)
11. print('bye')

• We start with line 10
  • Line 10 contains a call to f1, so we move execution to line 5
  • We copy 5 to local variable n (line 5) in f1
• Create variable k and assign it a value of 6
  • Line 7 contains a call to f2, so we move execution to line 1
  • We copy the value of 6 to local variable n (line 1) in f2
  • We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
  • We print 60
  • f2 is done, what do we do next?
  • Go to line 8 in the code
Let’s trace some code!

1. def f2(n):
2.     n = n*10
3.     print(n)
4. 
5. def f1(n):
6.     k = n+1
7.     f2(k)
8.     print(n,k)
9. 
10. f1(5)
11. print('bye')

• We start with line 10
• Line 10 contains a call to f1, so we move execution to line 5
• We copy 5 to local variable n (line 5) in f1
• Create variable k and assign it a value of 6
• Line 7 contains a call to f2, so we move execution to line 1
• We copy the value of 6 to local variable n (line 1) in f2
• We change the value of local variable n to 60 (line 2). Does anything happen to n in f1?
• We print 60
• f2 is done, what do we do next?
• Go to line 8 in the code, but how does your computer know that?
Activation Records

Function calls are implemented using a stack of activation records

An activation record contains all the information needed to continue execution after completing a method call:
- Values of arguments and local variables
- Instruction pointer – address of next instruction to execute

Algorithm:
- Push main’s activation record to the stack
- While stack is not empty:
  - Let A be the activation record at the top of the stack
  - if A makes a function call, create new activation record with that call and push it to the stack
  - if A finishes execution, destroy it