Analyzing function calls

Stack diagrams - allow us to visualize the memory and state of the program

Steps for writing stack diagrams:

1. Pause when you see the function call
2. Create a stack frame for the called function
   a. parameters allocated
   b. local variables in scope stored here
3. Copy argument values to parameters
4. Execute the function, line by line, until the return
5. Send back the return value
6. Pop the function off the stack
7. Resume the calling function

Tip: Use Python Tutor to check your answers!
Exercise - add.py

```python
A program that adds two numbers using a function

def add(x, y):
    print(x, y)
    return x + y

def main():
    n = 2
    out = add(n, 4)
    print(out)
```

main()
Exercise - add.py

Function Stack

Heap

```
def main():
    n = 2
    out = add(n, 4)
    print(out)

main()
```
Exercise - add.py

```
A program that adds two numbers using a function

```edef add(x, y):
def main():	print(x, y)	return x + y
def main():
    n = 2
    out = add(n, y)
    print(out)
main()
```
Exercise - add.py

```
A program that adds two numbers using a function

```

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    n = 2
    out = add(n, 4)
    print(out)

main()
```
Exercise - add.py

```
A program that adds two numbers using a function

```
Exercise - add.py

Function Stack

<table>
<thead>
<tr>
<th>add</th>
<th>x</th>
<th>y</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Heap

- 6 (return value)
- 4
- 2

A program that adds two numbers using a function:

```python
def add(x, y):
    print(x, y)
    return x + y
```

```python
def main():
    n = 2
    out = add(n, 4)
    print(out)

main()
```
Exercise - add.py

A program that adds two numbers using a function

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    n = 2
    out = add(n, 4)
    print(out)

main()
```
Exercise - add.py

A program that adds two numbers using a function.

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    n = 2
    out = add(n, 4)
    print(out)

main()
```

```py
n
x
y
add
?
? return value
main

out
```
Exercise - add2.py

Function Stack | Heap

main

def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    x = add(x, x)
    print(x)

main()
Exercise - add2.py

```python
A program that adds two numbers using a function

```def add(x, y):
    print(x, y)
    return x + y```def main():
x = 3
x = add(x, x)
print(x)```main()```
Exercise - add2.py

Function Stack

Heap

main

x

2

A program that adds two numbers using a function

def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    y = add(x, x)
    print(y)

main()
Exercise - add2.py

A program that adds two numbers using a function.

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    x = add(x, x)
    print(x)

main()
```
Exercise - add2.py

A program that adds two numbers using a function

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    x = add(x, x)
    print(x)

main()
```
Exercise - add2.py

```
add2.py

A program that adds two numbers using a function

```
Exercise - add2.py

A program that adds two numbers using a function

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    x = add(x, x)
    print(x)
```

main()
Exercise - add2.py

A program that adds two numbers using a function

```python
def add(x, y):
    print(x, y)
    return x + y

def main():
    x = 2
    x = add(x, x)
    print(x)

main()
```
Exercise - add3.py

```python
***
A program that adds two numbers using a function
***

def add(x, y):
    x = 10
    print(x, y)
    return x + y

def main():
    x = ?
    out = add(x, x)
    print(x, out)
main()
```
Exercise - add3.py

A program that adds two numbers using a function

```python
def add(x, y):
    x = 10
    print(x, y)
    return x + y

def main():
    x = 2
    out = add(x, y)
    print(x, out)
main()
```
Exercise - add3.py

Function Stack

- add
- x
- y

Heap

- main
- x
- 2

```
# A program that adds two numbers using a function

def add(x, y):
    x = 10
    print(x, y)
    return x + y

def main():
    x = 2
    out = add(x, x)
    print(x, out)
main()
```
Exercise - add3.py

A program that adds two numbers using a function:

```python
# A program that adds two numbers using a function

def add(x, y):
    x = 10
    print(x, y)
    return x + y

def main():
    x = 2
    out = add(x, x)
    print(x, out)

main()
```
Exercise - add3.py

Function Stack

```
add
x
y
main
x
```

Heap

```
10
2
```

```python
add(x, y):
    return x + y

main():
    x = 2
    out = add(x, x)
    print(x, out)
```
Exercise - add3.py

Function Stack

- add
- x
- y
- return value

Heap

- 10
- 12
- 2

- x

```python
def add(x, y):
    x = 10
    print(x, y)
    return x + y

def main():
    x = 2
    out = add(x, x)
    print(x, out)

main()
```
Exercise - add3.py

Note that x has the same value in main() after add() returns!
Exercise - add3.py

Note that x has the same value in main() after add() returns!
Exercise - tip.py

Function Stack

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()}
Exercise - tip.py

Function Stack

Heap

main

cost 10

Assume cost = 10

```python
def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()
```
Exercise - tip.py

```python
A program that computes the total cost with tip using a function.

```compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total```

```python
def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)
```
Exercise - tip.py

```python
def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()
```
Exercise - tip.py

```python
def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is ", total_cost)

main()
```
Exercise - tip.py

```python
A program that computes the total cost with tip using a function.

```
Exercise - tip.py

Function Stack
- add
- amount
- tip_percent
- tip
- total
- return value

Heap
- 1.0
- 11.0
- return value

main
- cost
- percent
- total_cost

A program that computes the total cost with tip using a function.

```python
def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: ")
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()
```
A program that computes the total cost with tip using a function.

```python
# A program that computes the total cost with tip using a function.

def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()
```
Exercise - tip.py

```
# A program that computes the total cost with tip using a function.

def compute_total(amount, tip_percent):
    tip = amount * tip_percent
    total = amount + tip
    return total

def main():
    cost = float(input("Enter the cost: "))
    percent = 0.1
    total_cost = compute_total(cost, percent)
    print("The total cost is", total_cost)

main()
```