Iteration

Announcements

Return

Return Statements

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

  f(x) for user-defined function f: switch to a new environment; execute f's body

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

`f(x)` for user-defined function `f`: switch to a new environment; execute `f`'s body

`return` statement within `f`: switch back to the previous environment; `f(x)` now has a value

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

`f(x)` for user-defined function `f`: switch to a new environment; execute `f`'s body

`return` statement within `f`: switch back to the previous environment; `f(x)` now has a value

Only one return statement is ever executed while executing the body of a function

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

f(x) for user-defined function f: switch to a new environment; execute f's body

return statement within f: switch back to the previous environment; f(x) now has a value

Only one return statement is ever executed while executing the body of a function

```
def end(n, d):
    """Print the final digits of N in reverse order until D is found.

    >>> end(34567, 5)
    7
    6
    5
    """
```

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

f(x) for user-defined function f: switch to a new environment; execute f's body

return statement within f: switch back to the previous environment; f(x) now has a value

Only one return statement is ever executed while executing the body of a function

```python
def end(n, d):
    """Print the final digits of N in reverse order until D is found.

    >>> end(34567, 5)
    7
    6
    5
    """
    while n > 0:
        last, n = n % 10, n // 10
        print(last)
```

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

f(x) for user-defined function f: switch to a new environment; execute f's body

return statement within f: switch back to the previous environment; f(x) now has a value

Only one return statement is ever executed while executing the body of a function

```
def end(n, d):
    """Print the final digits of N in reverse order until D is found.

    >>> end(34567, 5)
    7
    6
    5
    """
    while n > 0:
        last, n = n % 10, n // 10
        print(last)
        if d == last:
            return None
```

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

  f(x) for user-defined function f: switch to a new environment; execute f's body

  return statement within f: switch back to the previous environment; f(x) now has a value

Only one return statement is ever executed while executing the body of a function

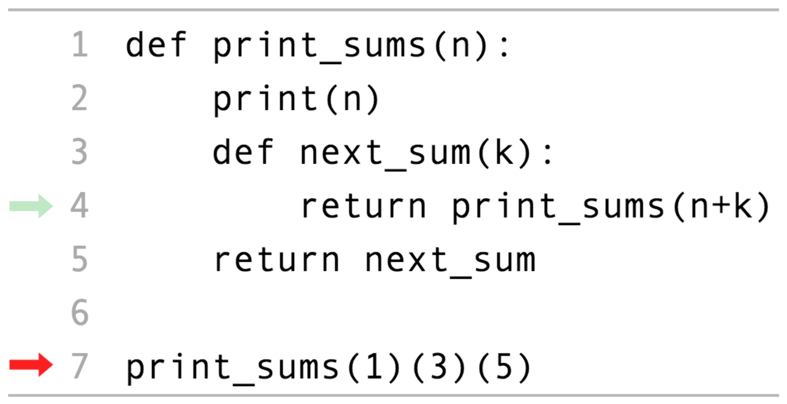
```
def end(n, d):
    """Print the final digits of N in reverse order until D is found.

    >>> end(34567, 5)
    7
    6
    5
    """
    while n > 0:
        last, n = n % 10, n // 10
        print(last)
        if d == last:
            return None
```

(Demo)

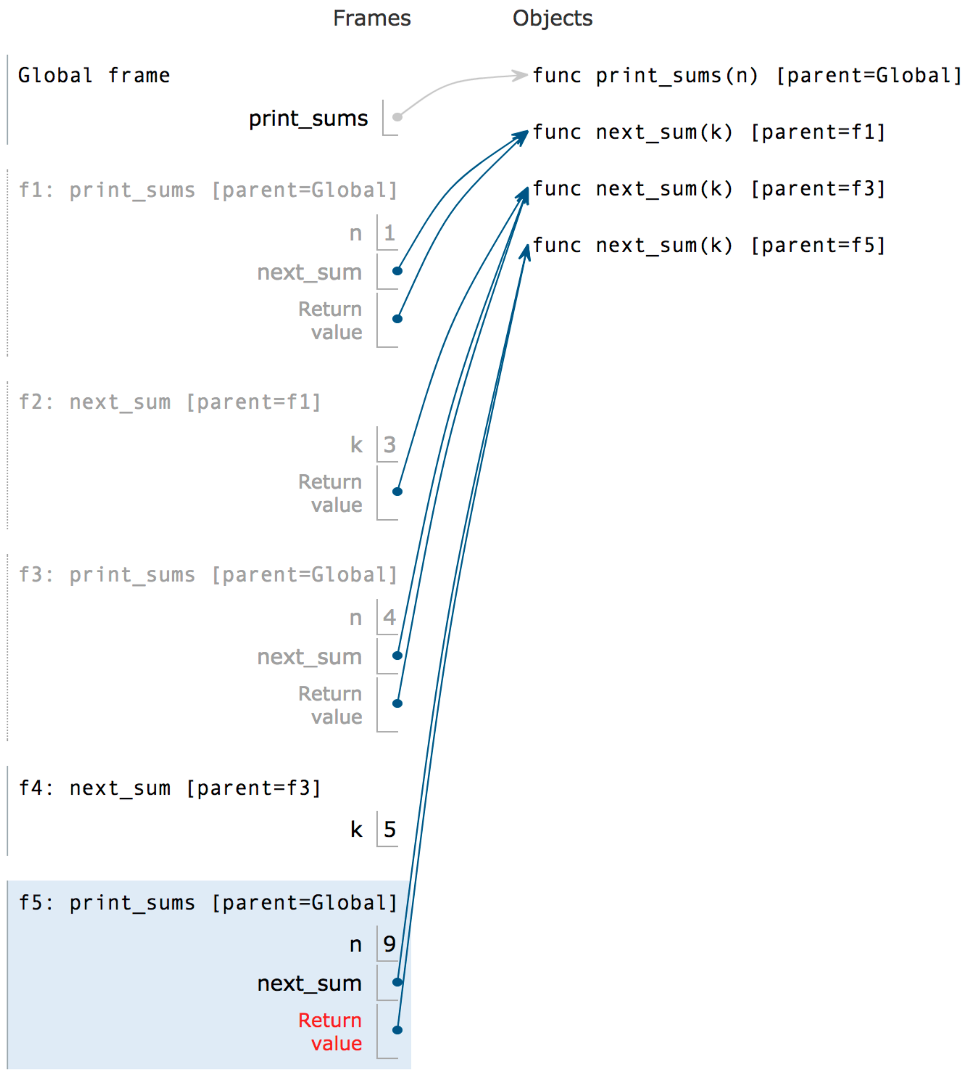
Self-Reference

(Demo)

Returning a Function Using Its Own Name



```
def print_sums(n):
    print(n)
    def next_sum(k):
        return print_sums(n+k)
    return next_sum

print_sums(1)(3)(5)
```



Frames
Objects
Global frame
print_sums
func print_sums(n) [parent=Global]
func next_sum(k) [parent=f1]
func next_sum(k) [parent=f3]
func next_sum(k) [parent=f5]
f1: print_sums [parent=Global]
n 1
next_sum
Return value
f2: next_sum [parent=f1]
k 3
Return value
f3: print_sums [parent=Global]
n 4
next_sum
Return value
f4: next_sum [parent=f3]
k 5
f5: print_sums [parent=Global]
n 9
next_sum
Return value

http://pythontutor.com/composingprograms.html#code=def%20print_all%28k%29%3A%0A%20%20%20%20print%28k%29%0A%20%20%20%20return%20print_all%0A%20%20%20%20%0Aprint_all%281%29%283%29%285%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

http://pythontutor.com/composingprograms.html#code=def%20print_sums%28n%29%3A%0A%20%20%20%20print%28n%29%0A%20%20%20%20def%20next_sum%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20print_sums%28n%2Bk%29%0A%20%20%20%20return%20next_sum%0A%0Aprint_sums%281%29%283%29%285%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Control

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
if __________:

    _________

else:

    _________
```

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
if __________:
    _________
else:
    _________
```

Execution Rule for Conditional Statements:

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
            if __________:
                _________
            else:
                _________
```

Execution Rule for Conditional Statements:

Each clause is considered in order.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
if __________:
    _________
else:
    _________
```

Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression (if present).

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
if __________:
    _________
else:
    _________
```

Execution Rule for Conditional Statements:

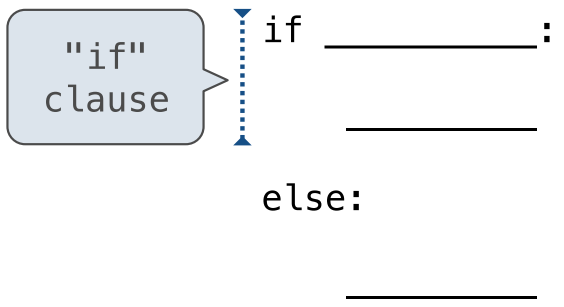
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header),
   execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

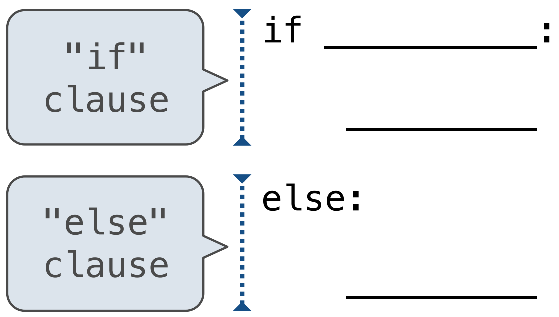
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header),
   execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



```
if __________:
    _________
else:
    _________
```

Execution Rule for Conditional Statements:

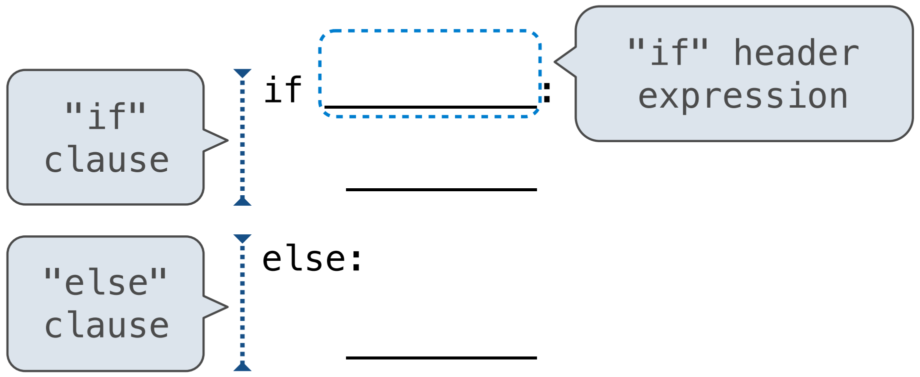
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header),
   execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

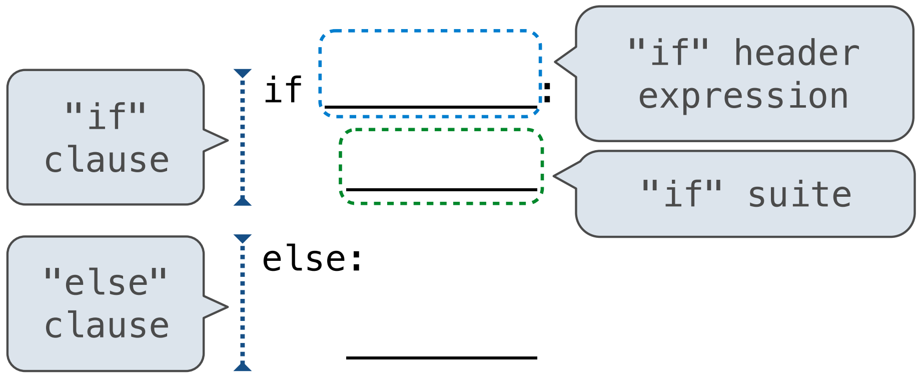
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header),
   execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

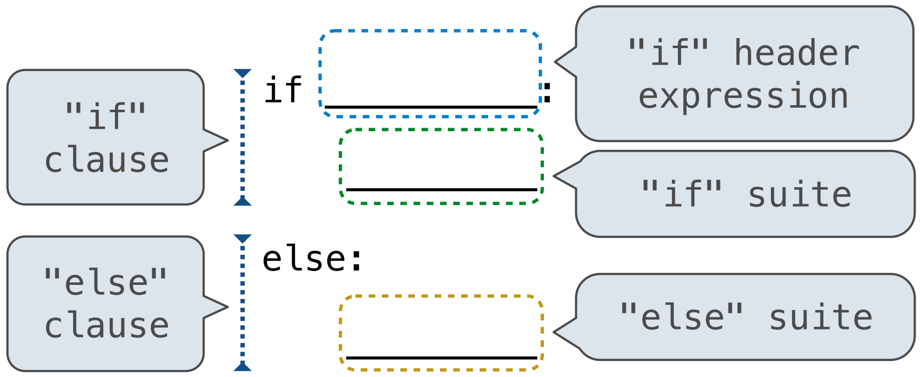
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

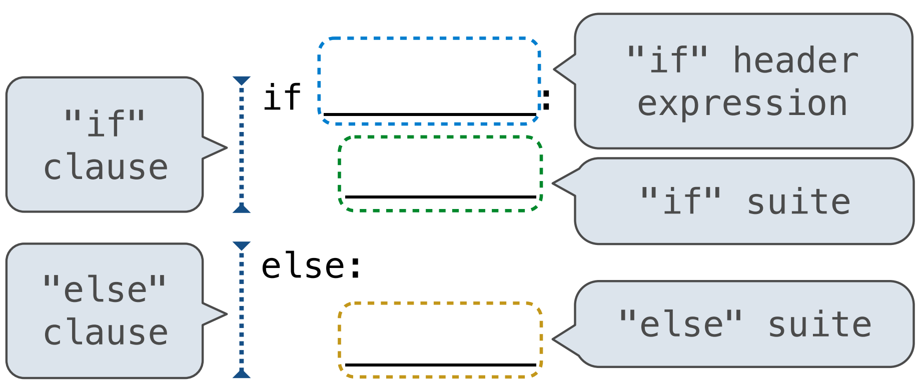
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

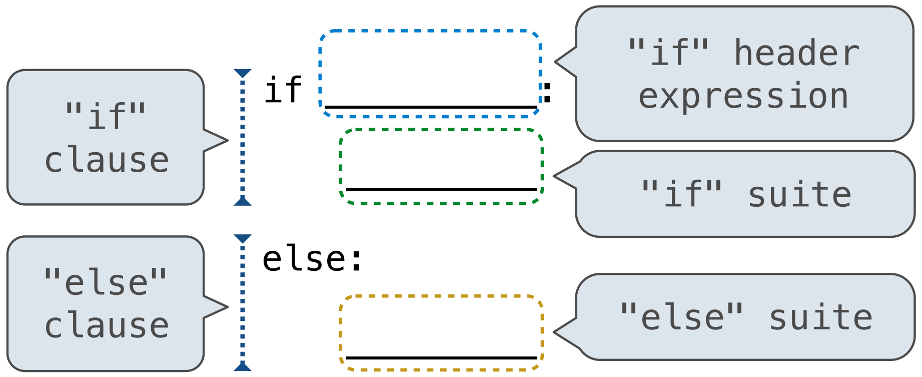
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.



Execution Rule for Conditional Statements:

Each clause is considered in order.

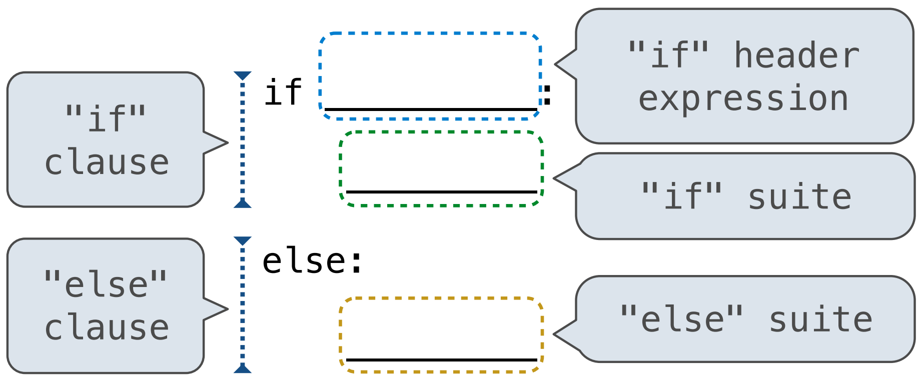
1. Evaluate the header's expression (if present).

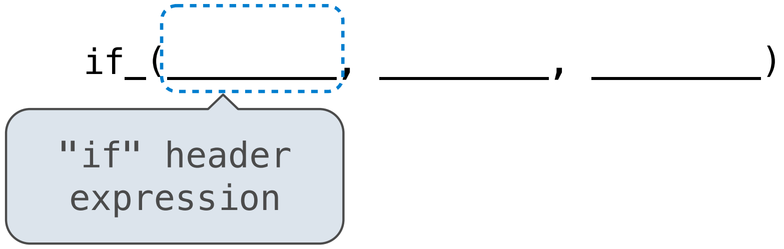
2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

```
if_(________, ________, ________)
```

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.





Execution Rule for Conditional Statements:

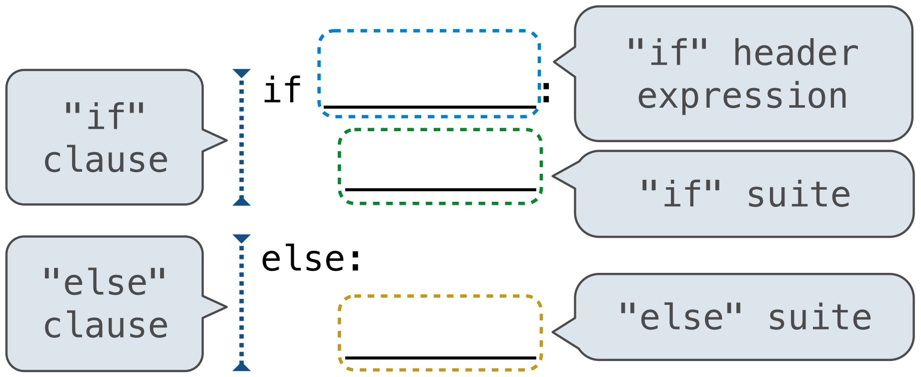
Each clause is considered in order.

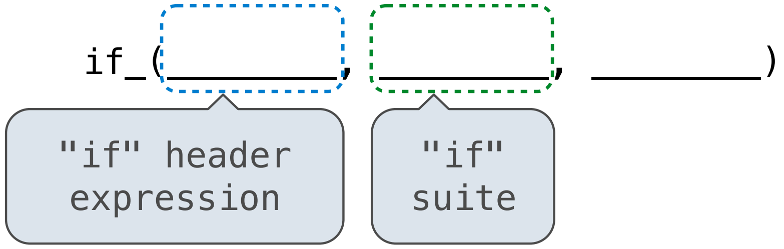
1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.





Execution Rule for Conditional Statements:

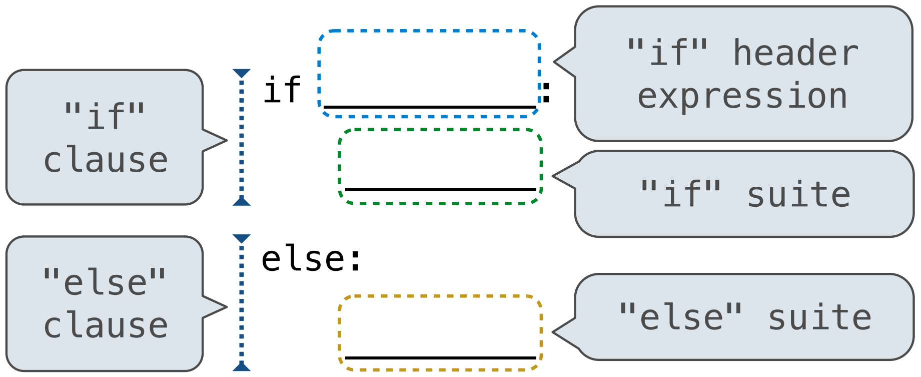
Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

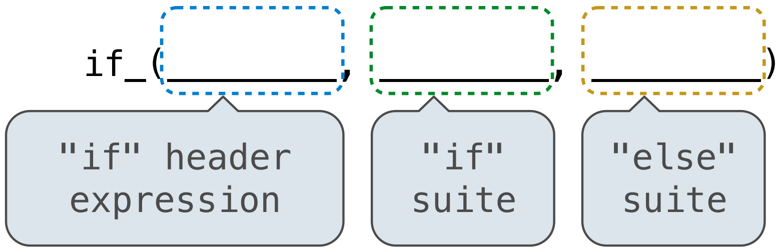


Execution Rule for Conditional Statements:

Each clause is considered in order.

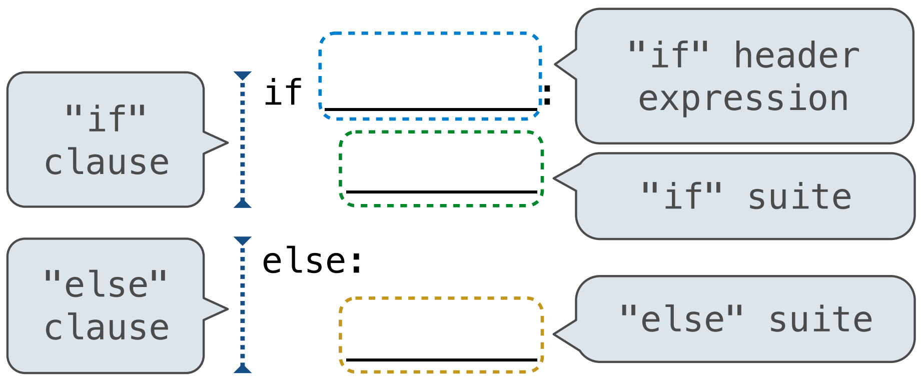
1. Evaluate the header's expression (if present).

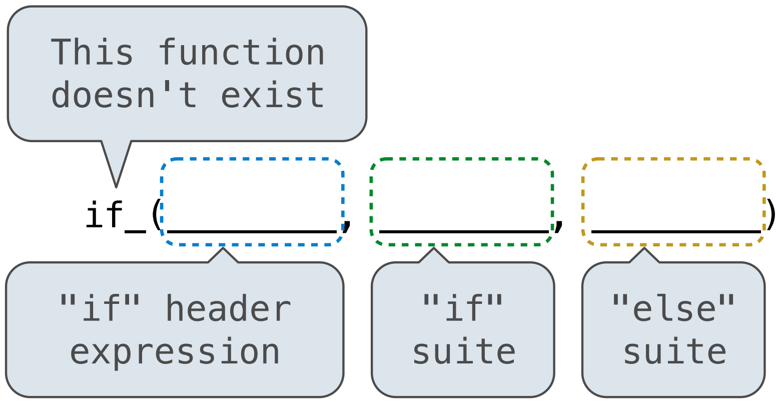
2. If it is a true value (or an else header),
   execute the suite & skip the remaining clauses.



If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.




This function doesn't exist
if_(__________, __________, __________)
"if" header expression
"if" suite
"else" suite

Execution Rule for Conditional Statements:

Each clause is considered in order.

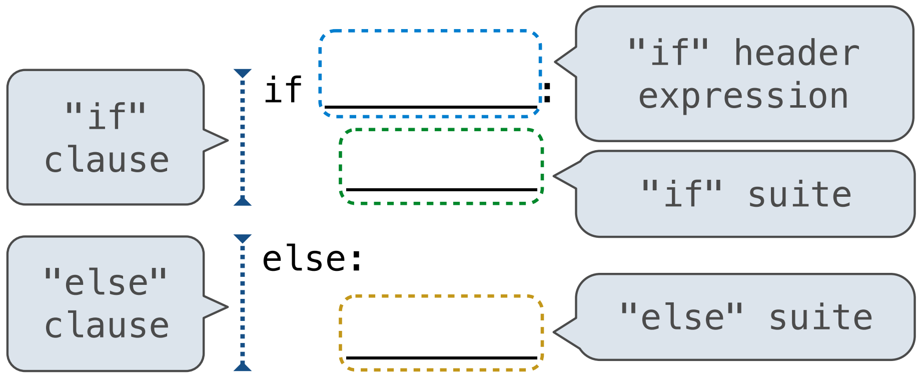
1. Evaluate the header's expression (if present).

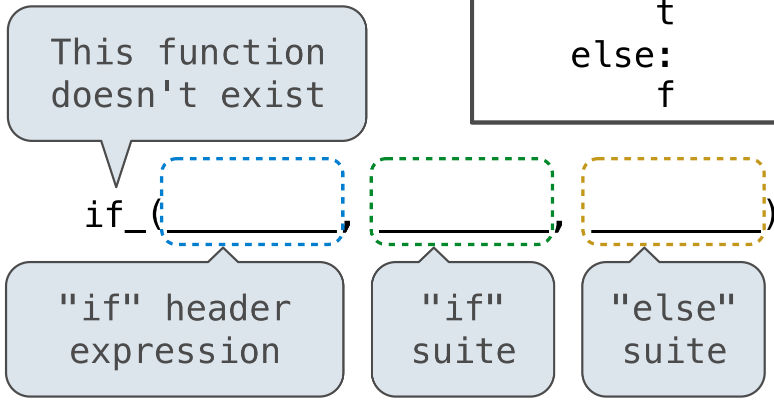
2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
def if_(c, t, f):
    if c:
        t
    else:
        f
```





Execution Rule for Conditional Statements:

Each clause is considered in order.

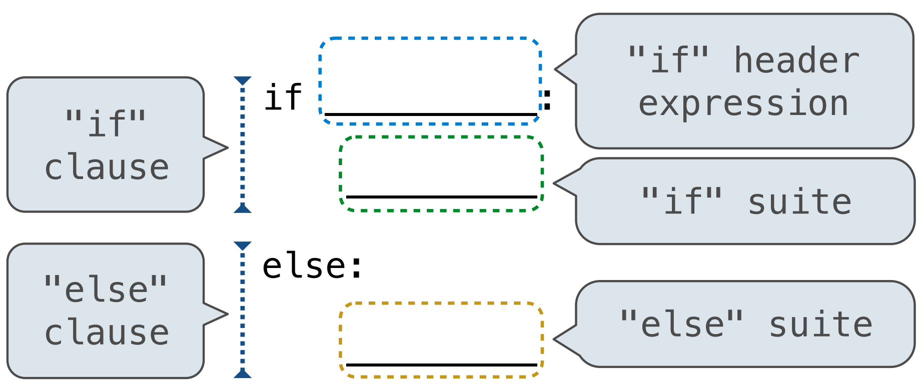
1. Evaluate the header's expression (if present).

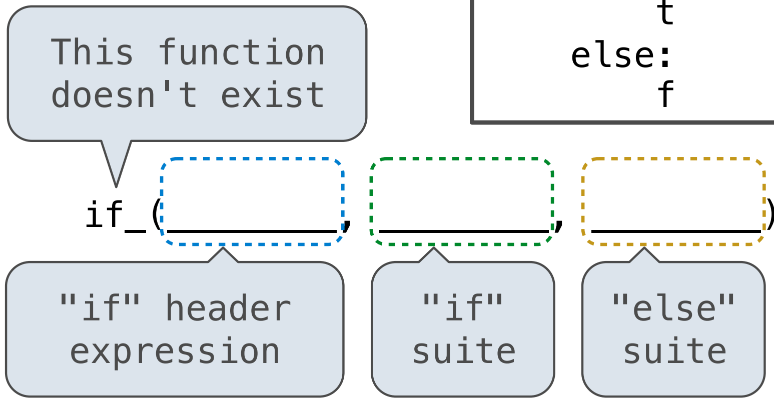
2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
def if_(c, t, f):
    if c:
        t
    else:
        f
```




This function doesn't exist
if_(________, ________, ________)
"if" header expression
"if" suite
"else" suite

Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression (if present).

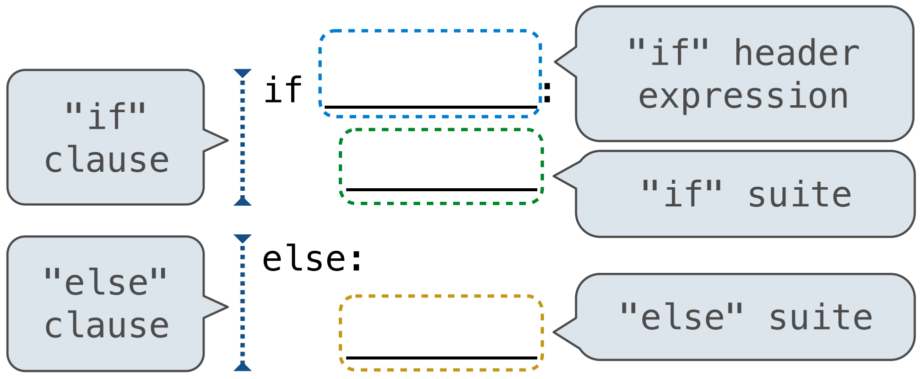
2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

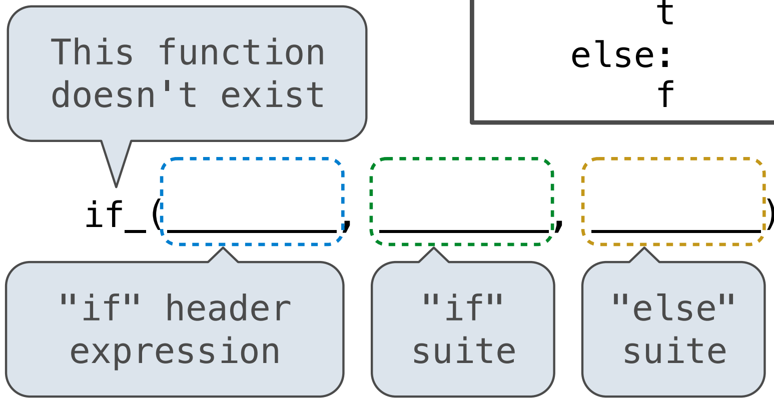
Evaluation Rule for Call Expressions:

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
def if_(c, t, f):
    if c:
        t
    else:
        f
```





This function doesn't exist
if_(___________, ___________, ___________)
"if" header expression
"if" suite
"else" suite

Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

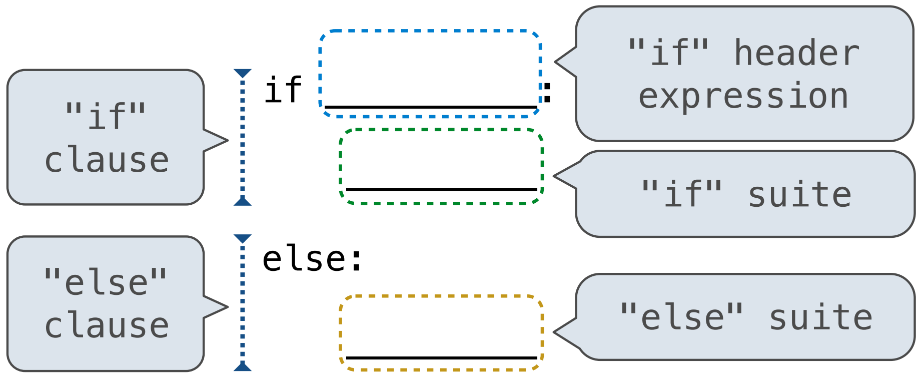
Evaluation Rule for Call Expressions:

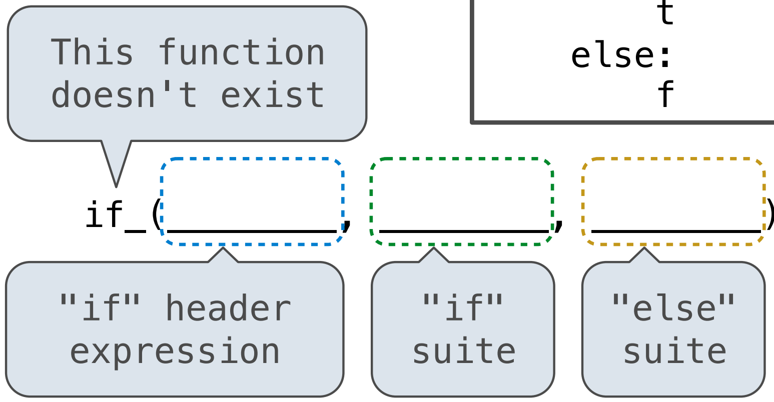
1. Evaluate the operator and then the operand subexpressions

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
def if_(c, t, f):
    if c:
        t
    else:
        f
```





Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

Evaluation Rule for Call Expressions:

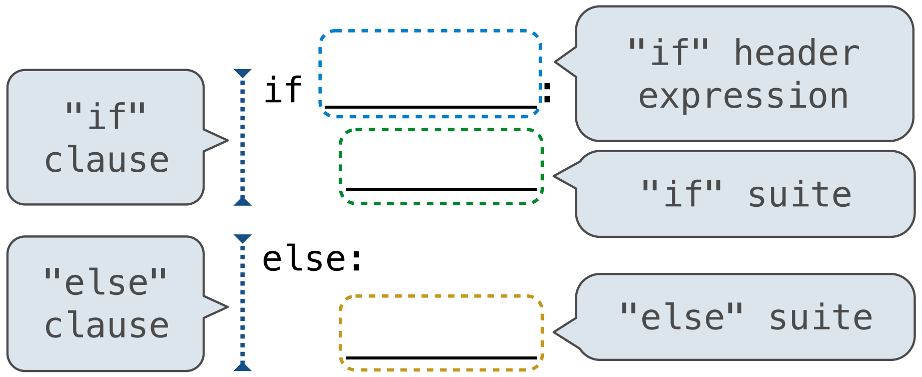
1. Evaluate the operator and then the operand subexpressions

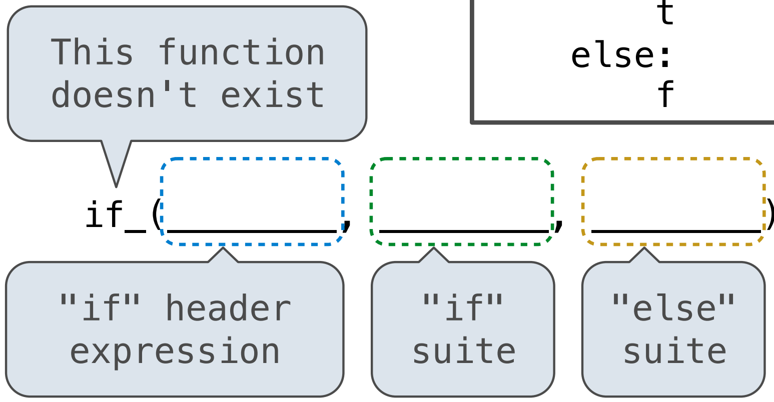
2. Apply the function that is the value of the operator to the arguments that are the values of the operands

If Statements and Call Expressions

Let's try to write a function that does the same thing as an if statement.

```
def if_(c, t, f):
    if c:
        t
    else:
        f
```





Execution Rule for Conditional Statements:

Each clause is considered in order.

1. Evaluate the header's expression (if present).

2. If it is a true value (or an else header), execute the suite & skip the remaining clauses.

(Demo)

Evaluation Rule for Call Expressions:

1. Evaluate the operator and then the operand subexpressions

2. Apply the function that is the value of the operator to the arguments that are the values of the operands

Control Expressions

Logical Operators

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>**.

2. If the result is a false value **v**, then the expression evaluates to **v**.

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a false value **v**, then the expression evaluates to **v.**
3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**

2. If the result is a false value **v**, then the expression evaluates to **v.**

3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

To evaluate the expression **<left>** **or** **<right>**:

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a false value **v**, then the expression evaluates to **v.**
3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

To evaluate the expression **<left>** **or** **<right>**:

1. Evaluate the subexpression **<left>.**

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a false value **v**, then the expression evaluates to **v.**
3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

To evaluate the expression **<left>** **or** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a true value **v**, then the expression evaluates to **v.**

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**

2. If the result is a false value **v**, then the expression evaluates to **v.**

3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

To evaluate the expression **<left>** **or** **<right>**:

1. Evaluate the subexpression **<left>.**

2. If the result is a true value **v**, then the expression evaluates to **v.**

3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

Logical Operators

To evaluate the expression **<left>** **and** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a false value **v**, then the expression evaluates to **v.**
3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

To evaluate the expression **<left>** **or** **<right>**:

1. Evaluate the subexpression **<left>.**
2. If the result is a true value **v**, then the expression evaluates to **v.**
3. Otherwise, the expression evaluates to the value of the subexpression **<right>.**

(Demo)

Conditional Expressions

Conditional Expressions

A conditional expression has the form

```
<consequent> if <predicate> else <alternative>
```

Conditional Expressions

A conditional expression has the form

```
<consequent> if <predicate> else <alternative>
```

Evaluation rule:

Conditional Expressions

A conditional expression has the form

```
<consequent> if <predicate> else <alternative>
```

Evaluation rule:

1. Evaluate the **<predicate>** expression.

Conditional Expressions

A conditional expression has the form

```
<consequent> if <predicate> else <alternative>
```

Evaluation rule:

1. Evaluate the **<predicate>** expression.

2. If it's a true value, the value of the whole expression is the value of the **<consequent>**.

Conditional Expressions

A conditional expression has the form

<consequent> if <predicate> else <alternative>

Evaluation rule:

1. Evaluate the **<predicate>** expression.

2. If it's a true value, the value of the whole expression is the value of the **<consequent>.**

3. Otherwise, the value of the whole expression is the value of the **<alternative>.**

Conditional Expressions

A conditional expression has the form

<consequent> if **<predicate>** else **<alternative>**

Evaluation rule:

1. Evaluate the **<predicate>** expression.

2. If it's a true value, the value of the whole expression is the value of the **<consequent>**.

3. Otherwise, the value of the whole expression is the value of the **<alternative>**.

```
>>> x = 0
>>> abs(1/x if x != 0 else 0)
0
```