Return

Return Statements

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

- For a user-defined function f: switch to a new environment; execute f’s body
- return statement within f: switch back to the previous environment; f’s return value is now available

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)
        if d == last:
            return None

>>> end(34567, 5)
7
6
5
None
```

Returning a Function Using Its Own Name

```python
# Returning a function using its own name

def print_all(k):
    print(k)
    return_all(k)
def print_sums(n):
    print(n)
    def sum_sums(k):
        return print_sums(n=k)
    return sum_sums

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

If Statements and Call Expressions

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

- Each clause is considered in order.
- Evaluate the header’s expression (if present).
- If it is a true value (or an else header), execute the suite & skip the remaining clauses.
- This function doesn’t exist

```python
def if(x, y, z):
    if x:
        if y:
            "if" header expression
            "if" suite
        else:
            if z:
                "if" header expression
                "if" suite
            else:
                "else" suite
```

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

```python
# Evaluation rule for call expressions

def if(x, y, z):
    if x:
        if y:
            "if" header expression
            "if" suite
        else:
            if z:
                "if" header expression
                "if" suite
            else:
                "else" suite
```
Control Expressions

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>`.

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>`.

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