

Control

Announcements

Print and None

(Demo)

Example: Printing and Returning

```
def f(x):  
    return x + 1
```

How are the following functions different?

(A)

```
def h(x):  
    f(x)
```

h(3)

Prints

(nothing)

(B)

```
def h(x):  
    print(f(x))
```

4

(C)

```
def h(x):  
    return print(f(x))
```

4

(D)

```
def h(x):  
    return f(x)
```

(nothing)

Returns

None

None

None

4

Default return value

Default return value

Return value of
print(f(x))

Example: Print Then Return



Implement a function `h(x)` that first prints, then returns, the value of `f(x)`.

```
def h(x):  
    return print(f(x))
```

(A)

```
def h(x):  
    print(f(x))  
    return f(x)
```

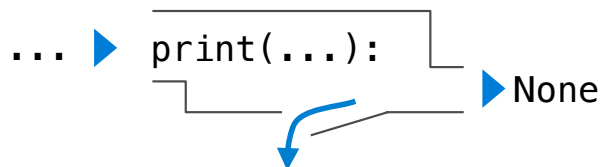
(B)

```
def h(x):  
    y = f(x)  
    print(y)  
    return y
```

(C)

pollev.com/denero

What's a function `f` for which implementations (B) and (C) would have different behavior?



Non-pure function has
side effects!
Happen twice for (C)

```
>>> h(2)  
...
```

(Demo)

```
>>> h(2)  
...
```

Control

Conditional Statements

Conditional statements (often called "If" Statements) contain statements that may or may not be evaluated.

		x=10	x=1	x=-1
<pre>if x > 2: print('big') if x > 0: print('positive')</pre>	Two separate (unrelated) conditional statements	big positive	positive	
<pre>if x > 2: print('big') elif x > 0: print('less big')</pre>	One statement with two clauses: if and elif Only one body can ever be executed	big	less big	
<pre>if x > 2: print('big') elif x > 0: print('less big') else: print('not pos')</pre>	One statement with three clauses: if, elif, else Only one body can ever be executed	big	less big	not pos

Boolean Values and Expressions

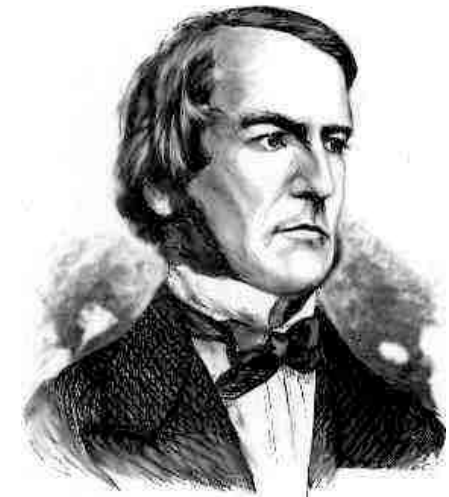
False values: `False`, `0`, `None`, `''`

Values that evaluate to false in a boolean context

Comparison operators: `>`, `<`, `>=`, `<=`, `==`, `!=`

Boolean operators: `or`, `and`, `not`

(Demo)



George Boole

While Statements

While statements contain statements that are repeated as long as some condition is true.

Important considerations:

- How many separate names are needed and what do they mean?
- The while condition **must eventually become a false value** for the statement to end (unless there is a return statement inside the while body).
- Once the while condition is evaluated, the entire body is executed.

Names and their initial values

```
1 i, total = 0, 0
```

```
2 while i < 3:
```

The while condition is evaluated before each iteration

A name that appears in the while condition is changing

```
    i = i + 1
```

```
    total = total + i
```

Executed even when i is set to 3

Example: Prime Factorization

Prime Factorization

Each positive integer n has a set of prime factors: primes whose product is n

...
 $8 = 2 * 2 * 2$
 $9 = 3 * 3$
 $10 = 2 * 5$
 $11 = 11$
 $12 = 2 * 2 * 3$
...

How can we determine whether a number is divisible by another?

One approach: Find the smallest prime factor of n , then divide by it

$$858 = 2 * 429 = 2 * 3 * 143 = 2 * 3 * 11 * 13$$

(Demo)