



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Lecture 3: Functions and Loops



Announcements

- We are working to expand the course!
 - Next week the waitlist will start to clear.
 - At least 60 seats on the waitlist
 - All concurrent enrollment students will get in (a separate waitlist)
- We'll be adding 2 new lab sections
 - Friday 11am
 - Friday 1pm
 - We will (re)release sections.c88c.org soon!
 - *If you already signed up no need to do anything.*



Let's talk Python

- Expression `3.1 * 2.6`
- *Call* expression `max(0, x)`
- Variables `my_name`
- Assignment Statement `my_name = <expression>`
- Define Statement: **`def function_name(<arguments>):`**
- Control Statements: **`if ...`**
`for ...`
`while ...`
- Comments `# Text after the # is ignored.`



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Python: Definitions and Control



Learning Objectives

- Create your own functions.
- Write a loop to run the same code multiple times
- Use conditionals to control when a loop stops



Variables In Python

- Variables “bind” (or assign) a *name* to a *value* (or expression)
- Variables can also come from *function arguments*
- Python has some specific rules about names...
 - Don’t memorize them all!
 - Mostly: No spaces, use _
- **Important:** Use meaningful names!
 - It’s a bit embarrassing to come to OH and try to explain the purpose of `butt` 😊
- `my_favorite_class = 'C88C'`



Functions in Python

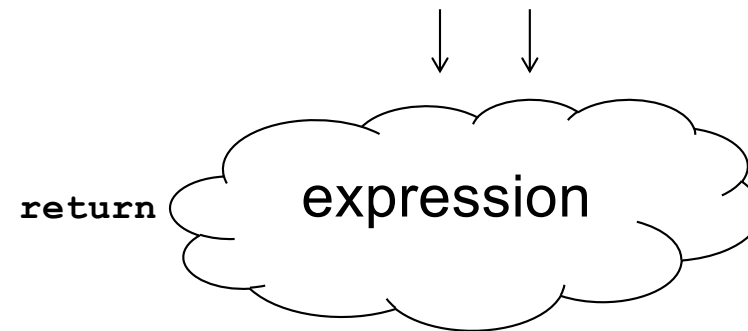
- We "define" them with `def`
- We typically name them using underscores ("Snake case")
- The first line ends in a `:`
- The body is indented by 4 spaces
- Arguments (parameters) create 'names' that exist only in our function
- Most functions will return a value, but some do not.

```
def print_greet(name):  
    print("Hello, " + name)  
def greet(name):  
    return "Hello, " + name
```

Defining Functions



```
def <function name> (<argument list>) :
```



- Abstracts an expression or set of statements to apply to lots of instances of the problem
- A function should *do one thing well*



Functions: Example

```
>>> y = 5
```

```
>>> x = 3
```

```
>>> z = max(3, 5) * 10
```

```
>>> z
```

```
50
```

```
def max(x, y):  
    if x > y:  
        return ( x )  
    else:  
        return ( y )
```

A diagram consisting of two arrows. A red arrow originates from the '5' in the function call 'max(3, 5)' in the line '>>> z = max(3, 5) * 10' and points to the 'y' parameter in the function definition 'def max(x, y):'. A blue arrow originates from the '3' in the same function call and points to the 'x' parameter in the function definition.



Returns and Values

- All functions always return SOME value.
- If you don't specify `return`, the value is `None`.
- Using `print` does not change how the function works, but does affect the output.



Functions: Calling and Returning Results

Python Tutor

```
# This style is shorthand.  
def max(x, y):  
    return x if x > y else y  
  
x = 3  
y = 4 + max(17, x + 6) * 0.1  
z = x / y
```



Doctests

- Write the docstring to explain *what* it does
 - What does the function return? What are corner cases for parameters?

```
def max(x, y):  
    """Returns the larger value of arguments x and y  
    >>> max(6, 0)  
    6  
    """"  
  
    return x if x > y else y
```

- Write doctest to show what it should do
 - Before you write the implementation.
 - `python3 -m doctest [-v] file.py`



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Python: Control Flow



Conditional Statement

- Do some statements, conditional on a *predicate* expression

```
if <predicate>:  
    <>true statements>  
else:  
    <>false statements>
```

- Example:

```
if temperature > 98.6:  
    print("fever!")  
else:  
    print("no fever")
```



Live Coding Demo

```
course = 'C88C'  
time = '2:00'  
if time == '2:00':  
    print("Go to " + course)  
else:  
    print("Go get some ☕")
```

Go to C88C



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Iteration with **while** Loops



Learning Objectives

- Use a while loop to repeat some task.
- Write an expression to control when a while loop stops executing



while Statement – Iteration Control

- Repeat a block of statements until a predicate expression is satisfied

```
<initialization statements>
```

```
while <predicate expression>:
```

```
    <body statements>
```

```
<rest of the program>
```



Sum The Numbers

- This is a task we'll see many times!

```
total = 0
n = 1
while n <= 10:
    total += n
    n += 1
print(total)
```



UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science



Iteration With **for** Loops



Learning Objectives

- Compare a for loop and a while loop.
- Learn to use range ()
- Use a string as a sequence of letters



for Statement – Iteration Control

- Repeat a block of statements for a structured sequence of variable bindings

```
<initialization statements>
```

```
for <variables> in <sequence expression>:
```

```
    <body statements>
```

```
<rest of the program>
```



<sequence expression> — What's that?

- Sequences are a *type* of data that can be broken down into smaller parts.
- Common sequences:
 - `range()` – gimme all the numbers
 - strings
 - lists (next!)
- We'll start with two basic facts:
 - `range(10)` is the numbers 0 to 9, or `range(0, 10)`
 - `[]` means "indexing" an item in a sequence.
 - `"Hello"[0] == "H"`



Data-Driven Iteration

- describe an expression to perform on each item in a sequence
- let the data dictate the control

```
[ <expr with loop var> for <loop var> in <sequence expr > ]
```