UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Loops and Functions
Q&A
Fall 2020 Lecture #3

Announcements

- Waitlist: We expect 10% to get off, but it's unknown.
- This lecture will start off as Q&A, we *don't* need to get through these slides.

- Gradescope: We'll do 1 question at a time.
  - Answers will be public after lecture.

Things You Can Do Now

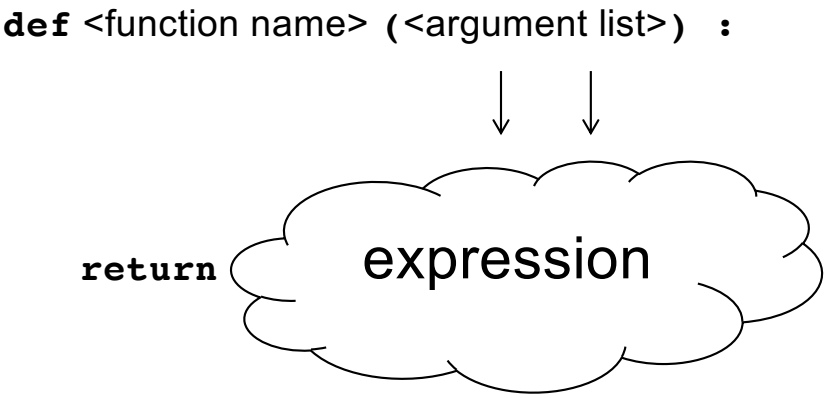
- Write a program that makes a decision.
- Write your own functions
- Use loops so you can process lots of data.

Let's Talk About Python

- Expression `3.1 * 2.6`
- Call expression `max(0, x)`
- Variables
- Assignment Statement `x = <expression>`
- Define Function: `def <function name> (<parameter list>):`
- Control Statements: `if …`
  `while …`

Q&A

Defining Functions



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

Doctests

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

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

Returns and Values

- All functions always return SOME value.
- If you don't specify `return`, the value is None.

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")
```

`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>
```

```
def first_primes(k):
    """ Return the first k primes.
    """
    primes = []
    num = 2
    while len(primes) < k :
        if prime(num):
            primes = primes + [num]
        num = num + 1
    return primes
```

Functions and Arguments

```
>>> x = 3
>>> y = 4 + max(17, x + 4) * 0.5
>>> z = x + y
>>> print(z)
15.5
```

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

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