Computational Structures in Data Science

Lecture 3:
Functions and Loops

Berkeley
UNIVERSITY OF CALIFORNIA

Announcements

- Lab Attendance:
  - Please review your autograder feedback to make sure the attendance choice is correct
- Earning points is based on *correctness*
  - Applies to all labs, self-checks, homework, projects.
  - You get as many tries as you need, but the results must work, at the end of the day.
  - If you need an extension, you can ask for one, but be careful with time. ☺
- My "Tea" Hours: Weds 5pm, 784 Soda – may move occasionaly.

Computational Structures in Data Science

Learning Process & Debugging

Process NOT Memorization

- This is not a class about memorization.
- This is a class about *problem solving* and *process.*
- You will not know everything, but you will be able to figure it out.
- Focus on building intuition!
  - **Predict** what will happen **first**
  - Then **try and inspect**
  - Now**,** Figure out **why**!
  - Was your prediction correct or incorrect?

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 …`**
	`while …`
	`for …`
•Comments	`# Text after the # is ignored.`

Computational Structures in Data Science

Python: Functions (Again)

Learning Objectives

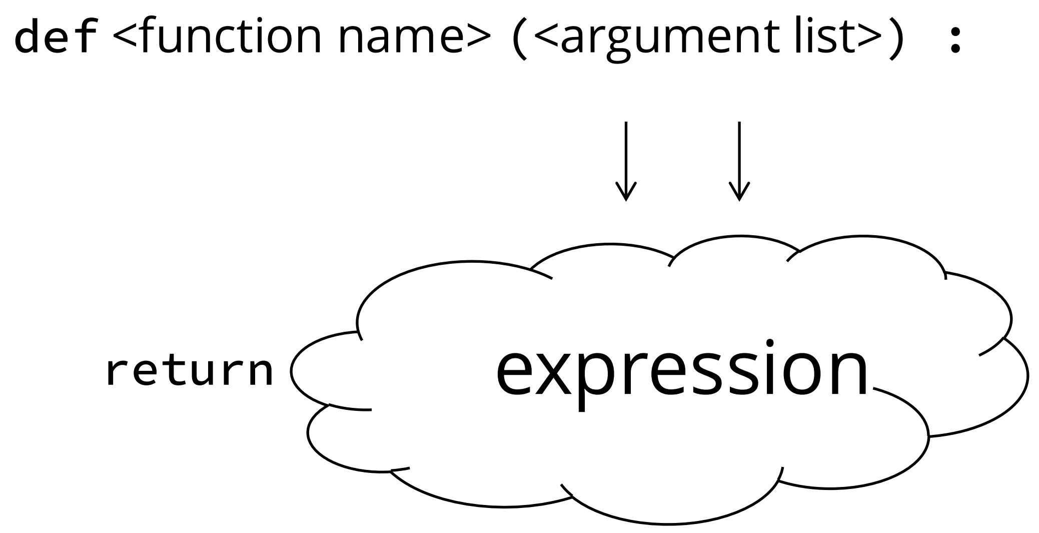
- Create your own functions.
- Write a loop to run the same code multiple times
- Use conditionals to control when a loop stops

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 "`buttface`" ☺ (This actually happened!)
- `my_favorite_class = 'C88C'`

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
- arguments become accessible inside the function body.



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.
  - If you don't specify a return statement, the value is None

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

Aside: String and Text

- Strings, or sequences of text are incredibly common!
- In Python we use ' or "
- We combine strings with +, or by using *string interpolation:*
- f-strings allow us to embed an expression inside some text!

```
def print_greet(name):
    # print("Hello, " + name)
    print(f"Hello, {name}")
```

What happens?

```
def print_greet(name):
    #Same as: print("Hello, " + name)
    print(f"Hello, {name}")

x = print_greet('C88C')
x
# What's x?
y = print_greet(c88c)
y
# What's y?
```

Computational Structures in Data Science

Python: Control Flow

Berkeley
UNIVERSITY OF CALIFORNIA

Conditional Statements

- 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

```python
course = 'C88C'
time = '3:00'
if time == '2:00':
    print(f"Go to {course}")
else:
    print("Go get some ☕")
```

What is shown?

Live Coding Demo

```
course = 'C88C'
time = '3:00'
if time == '2:00':
    print(f"Go to {course}")
else:
    print("Go get some ☕")

Go to C88C
```

Extending Conditional Statements

- Only 1 set of true statements OR the else body will be executed.
- The else body is optional.

```
if <predicate A>:
       <predicate A statements>
elif <predicate B>:
       <predicate B statements>
elif <predicate C>:
       <predicate C statements>
…
else:
       <false statements>
```

Consider these two cases

```python
year_in_school = 4
if year_in_school >= 4:
     print('Senior')
elif year_in_school >= 3:
    print('Junior')
elif year_in_school >= 2:
    print('Sophomore')
else:
    print('Freshmen')
```

```python
year_in_school = 4
if year_in_school >= 4:
     print('Senior')
if year_in_school >= 3:
    print('Junior')
if year_in_school >= 2:
    print('Sophomore')
if year_in_school >= 1:
    print('Freshmen')
```

Conditional Expression Shorthand

- Return a Value Based on some condition

```
<true expression> if <predicate> else <false expression>
```

- Example:

```
status = "it's hot!" if temperature > 85 else 'not hot…'
```

Computational Structures in Data Science

Python: Functions

(with conditionals)

Berkeley
UNIVERSITY OF CALIFORNIA

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
```

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]

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

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>

x = 1
while x < 10:
   print(x)
   x = x + 1 # or commonly x += 1

x
```

while Statement – Iteration Control

- Consider these two programs.
- What's different?

```
x = 1
while x < 10:
    print(x)
    x = x + 1 # or commonly x += 1
x

x = 1
while x <= 10:
    print(x)
    x += 1
x
```

Sum The Numbers

- This is a task we'll see many times!
- Do these do the same thing?

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

```
total_backwards = 0
z = 10
while z > 0:
    total_backwards += z
    z -= 1
print(total_backwards)
```

What happens if we mess up?

- What will happen here?

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

Computational Structures in Data Science

Iteration With `for` Loops

Berkeley
UNIVERSITY OF CALIFORNIA

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 broken down into smaller parts.
- Common sequences:
  - range() – give me all the numbers
  - Strings, e.g, "Hello, C88C!"
    - What is it a sequence of? Characters!
  - 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 > ]
```