Python user-defined functions

Midterm review

Computers do not solve problems - computers carry out solutions, specified by people, to problems.

An Algorithm for Succeeding in CIS 122

0. Skim the assigned reading

1. Reading and class prep work:
   - Read the assigned reading closely.
   - Try Python code from assigned reading.
   - Try the exercises at the end of the chapter.

2. Class and Lab
   - Come with questions from other steps.
   - Take notes.
   - Review posted notes and try the examples on your own.

3. Project work outside of class:
   - Review solution key to prior problem set.
   - Start working on the weekly problem set.
   - Attend lab help hours.
Python Functions

A function names an operation.

Python has built in functions and user-defined functions

Why functions?
Readability/Clarity/Information-hiding
Reuse/Flexibility

→ Abstraction

program building blocks

“fix it and forget it”
Functions – built in

For example,

```python
>>> type(x)
>>> int(x)
>>> float(x)
>>> min(x, y, ...)
>>> max(x, y, ...)
```

Python User-defined functions

User-defined functions associate a name with a user-defined operation/process.
User-defined functions associate a name with a user-defined operation/process.

\[ x = \]

```python
def add_one():
    
    z = y + 1
    print(z)
    return None
```

User-defined functions associate a name with a user-defined operation/process.

\[ x = 99 \]

```python
def add_one(y):
    
    z = y + 1
    print(z)
    return None
```
User-defined functions associate a name with a user-defined operation/process.

```python
x = 99

def add_one(y):
    
    z = y + 1
    print(z)
    return None

A function definition is not an expression and is not evaluated.
```

User-defined functions associate a name with a user-defined operation/process.

```python
x = 99

>>> x
99

def add_one(y):
    
    z = y + 1
    print(z)
    return None

>>> add_one
<function add_one at 0x103dec680>

A function definition is not an expression and is not evaluated.
As with built-in functions, a user-defined function must be called to execute:

```python
>>> add_one(4)
5
```
Parameters and Arguments

>>> def add_one(y):
    
    z = y + 1
    print(z)
    return None

y is a parameter
Parameters and Arguments

```python
>>> def add_one(y):
    """
    z = y + 1
    print(z)
    return None

y is a parameter - part of the function definition

>>> add_one(101)
>>> add_one(102)

101 and 102 are arguments – input values specified when the function is called
```

When a Python function is called:
```python
>>> add_one(101)
```

- Evaluate arguments left to right
- Create a namespace to hold the function’s local variables
- Assign argument values to parameters
- Execute the function body until return
- Return value specified in return
Parameters and Arguments

```python
>>> def add_one(y):
    """ """
    z = y + 1
    print(z)
    return None

>>> add_one(101)

y is assigned the value 101 when the function is called
```

Local Variables

```python
>>> def add_one(y):
    """ """
    z = y + 1
    print(y)
    return None

y and z are local variables
```
What is the result when this code is executed?

def calculate(w, x):
    
    a = x + 2
    b = w + 1
    print(a + b + 3)
    return None

>>> calculate
>>> calculate(1)
>>> calculate(1, 5)
>>> calculate(2, 0)

What value is returned after the following code is entered into the Python shell? (same code as prior)

>>> def calculate(w, x):
    w = 1
    x = 5
    a = x + 2
    b = w + 1
    print(a + b + 3)
    return None

>>> calculate(2, 0)
>>> calculate(3, 7)
Python Functions

✓ Python built-in functions
✓ combining functions
✓ user-defined functions
✓ keywords (def, return)
✓ defining v. calling
✓ parameters, arguments
✓ what happens when a function is executed
✓ local variables
• functions can call functions
• functions return a value
• function design recipe

Functions can call functions:

def calculate(w, x):
    """: ""
    a = x + 2
    b = w + 1  # b = add_one(w)
    print(a + b + 3)
    return None

>>> calculate(1, 5)
Functions can call functions:

```python
def calculate(w, x):
    a = x + 2
    b = add_one(w)
    print(a + b + 3)
    return None

def add_one(x):
    y = x + 1
    print(y)
    return None

>>> calculate(1, 5)
? ???
```
def add_one(x):
    """ """
    y = x + 1
    print(y)
    return y

>>> add_one(99)
??

Recall:
Functions are expressions and are evaluated.

# def add_one(x):
#     y = x + 1
#     print(y)
#     return y
# >>> add_one(99)
# 100
# when function executes, y is printed
# when function executes, y is returned
# and printed by Python REPL

Functions are expressions and are evaluated.

They evaluate to the value in the return statement.
def add_one(x):
    
    y = x + 1
    print(y)
    return y

>>> add_one(99)
100
100

Functions are expressions and are evaluated.

They evaluate to the value in the return statement.

Functions can call functions:

def calculate(w, x):
    
    a = x + 2
    b = add_one(w)
    print(a + b + 3)
    return None

>>> calculate(2, 6)
Functions can call functions:

```python
def calculate(w, x):
    """
    a = x  + 2
    b = add_one(w)
    print(a + b + 3)
    return None
    """

add_one(x):
    """
    y = x + 1
    #print(y)
    return 0
    """

>>> calculate(2, 6)
????
def add_one(x):
    ''' (number) -> number  # type contract
    Add one to x; print and return
    the result.
    
    >>> add_one(99)  # examples of use
    100
    >>> add_one(1.1)
    2.1
    '''
    y = x + 1  # function body
    return y  # return statement

Writing programs and functions
using the function design recipe
Cricket chirps: Nature's Thermometer

According to the Farmer's Almanac, counting cricket chirps is a reliable guide to determining the outdoor temperature (when the temperature is between 55 and 100 degrees Fahrenheit, as crickets do not chirp outside of this range).

The formulae are as follows:

To convert cricket chirps to degrees Fahrenheit, count number of chirps in 14 seconds then add 40 to get temperature.

Example: 30 chirps + 40 = 70° F

Using the function design recipe, define a function, `chirps_to_f temp`, that takes as input the number of cricket chirps in a 14 second interval, and prints the outdoor temperature in Fahrenheit degrees. `chirps_to_f temp` will return the value None.

Test your function on the following examples:

```python
>>> chirps_to_f_temp(30)
70
>>> chirps_to_f_temp(55)
95
>>> chirps_to_f_temp(15)
55
```
This is how you start ...

Write the docstring first:

– Examples

– Type contract

– Brief description of function
Cricket chirps: Nature's Thermometer

In this case, the examples were given in the project specification:

```python
>>> chirps_to_f$temp(30)
70
>>> chirps_to_f$temp(55)
95
>>> chirps_to_f$temp(15)
55
```

Type contract

What parameters are needed?

What value will the function return?
Type contract

What parameters are needed?
Number of cricket chirps in a 14-second interval

What value will the function return?
None (see project specification)
Write a brief description of what the function will do.

Estimate Fahrenheit temperature based on number of cricket chirps in a 14 second interval; print the estimated temperature.
def chirps_to_ftemp(chirps):
    '''
    (int) -> None

    Print estimated fahrenheit temperature based on number of chirps in a 14-second interval.
    
    >>> chirps_to_ftemp(30)
    Estimated temp is 70 degrees Fahrenheit.
    >>> chirps_to_ftemp(55)
    Estimated temp is 95 degrees Fahrenheit.
    '''
    pass
    return None

Develop an algorithm:
Develop an algorithm to solve the problem:

\[ f_{temp} \leftarrow \text{add 40 to } \# \text{ cricket chirps} \]

Develop an algorithm:

\[ f_{temp} \leftarrow \text{add 40 to } \# \text{ cricket chirps} \]

Review Python tools and implement algorithm:
Develop an algorithm:

\[ f_{\text{temp}} \leftarrow \text{add 40 to } \# \text{ cricket chirps} \]

Review Python tools and implement algorithm:

\[ f_{\text{temp}} = \text{chirps} + 40 \]

def chirps_to_ftemp(chirps):
    """(int) -> None
    # type contract
    # brief description
    Print estimated fahrenheit temp based on cricket chirps.
    >>> chirps_to_ftemp(30) # example(s) of use
    Estimated temp is 70 degrees Fahrenheit.
    ""
    ftemp = chirps * 40 # function body
    print('Estimated temp is', ftemp, 'degrees Fahrenheit.')
    return None # return
Test the code using examples:

```python
>>> chirps_to_ftemp(30)
Estimated temp is 1200 degrees Fahrenheit.
```

Revise and retest as needed
Thinking:
   – example(s) of use
   – type contract
   – brief description
   – algorithm

Turn it into Python code:
   – write docstring (+ pass and return None)
   – review Python toolkit
   – write Python code

Test code using examples
Revise and retest

Python Functions
✓ Python built-in functions
✓ combining functions
✓ user-defined functions
✓ keywords (def, return)
✓ defining v. calling
✓ parameters, arguments
✓ what happens when a function is called
✓ local variables
✓ functions can call functions
✓ functions return a value
✓ function design recipe
Python toolkit so far

Data types:
numbers (int and float)
strings
functions (executable data type)

Built-in functions and operators:
+ - * / // % ** [ ] [:] int, float, str for
pow, round, abs print, input min, max
type, id help range

Errors:
SyntaxError Syntax
TypeError Runtime NameError
IndexError Logic

CIS 122 Introduction to Programming Spring 2017  Week 4.2.

string methods – count, find, index

import
modules – math, turtle

function definition (def)
CIS 122 Introduction to Programming Spring 2017

Bug Collection So Far

>>> x === 5
SyntaxError: invalid syntax

>>> y
NameError: name 'y' is not defined

>>> 9 + 'planets'
TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> greeting[len(greeting)]
IndexError: string index out of range

CIS 122 Introduction to Programming Spring 2017

Bug Collection So Far

>>> abs()
Traceback (most recent call last): File "<pyshell#0>" line 1, in <module>  
abs() TypeError: abs() takes exactly one argument (0 given)

>>>