



CPSC 103

Introduction to Systematic Program Design 2021S

Lecture: Module 2- How to Design Functions

Ashish Chopra

13 May, 2021

1



designed by freepik

Announcements

1. Post-Lecture office hours (1hr after every lecture) starting today. 5-6pm
2. Withdraw Deadline Tomorrow. May 14
- ✓ 3. Code Review and Worksheet (Module 1 and Module 2) due on Sunday May 16th 10pm PDT.
- ✓ 4. Syllabus is updated with new deadlines for Project.
- ✓ 5. Project Module will be released on Sunday 12 am PDT.
6. Module 3 and Module 4 released for pre-class readings. Pre-lecture Assignments due by Mon ~~13~~ May, 10pm PDT.

17 May

Tips to succeed!

1. After the lecture, give a reading to HtDF recipe page, and Worked Example in Module 2.
2. Keep practicing problems on your own. HtDF Design Recipe will become your second nature!
3. With all that work, do something to relax over the weekend!

Learning Goals

1. How to design Functions that operate on primitive data using HtDF recipe.
2. Read complete function design and identify its different elements.
3. Evaluate different elements of function design for clarity, simplicity and consistency with each other.
4. You can explain the purpose of HtDF recipe's steps in a way that generalizes to other design problems.
5. Evaluate entire design for how well it solves a given problem.

Last Lecture: Summary

1. We looked at different building blocks of a program:
 1. Values and Data Types (primitive data)
 2. Variables ✓
 3. Operations $+$ $/$ $-$ $*$ $>$ $<$ $=$ $!$ and or
 4. Statements
 5. Functions
2. Conditional statements using if, if-else, if-elif-else.
3. How to call a function. ✓
4. How to write your own functions (simple ones)

Today, we will
design
functions!

Systematic Design Process

1. It helps in making clearer what the problem means. ✓
2. It is a process to go about solving a problem systematically.
3. The outcome (function or program) is well-structured, and well-tested.
4. The code/functions are easy to read, understand, and maintain.

You will be graded primarily on following our design process for the remainder of the term! The correctness of your code is just one small part of the whole design process.

Designing Functions: HtDF Recipe

HtDF recipe is about designing functions which are correct, easy to read, understand and maintain.

HtDF recipe consists of the following steps:

1. Signature, purpose, and stub.
2. Examples
3. Template
4. Code the function body
5. Test and Debug until correct

```
2
3 @typecheck
4 def declare_result(score: float) -> str: ✓
5     """
6     returns "Pass" if the given score is greater
7     than or equal to 50, "Fail" otherwise. ✓
8     """
9     # return "Pass" #stub ✓
10    # return ...(score) #template
11    if score >= 50:
12        return "Pass"
13    else:
14        return "Fail"
15
16
17 start_testing()
18 expect(declare_result(89), "Pass")
19 expect(declare_result(34), "Fail")
20 expect(declare_result(50), "Pass")
21 summary()

3 of 3 tests passed
```

Handwritten annotations: Blue circles 1, 3, 4, 2, and 5 are placed next to the code blocks. Blue boxes highlight the function signature, the template, the function body, the test cases, and the test results.

1.1 Signature

A signature of a function defines the name of the function, the input type(s) and output type of the function.

function

Problem: Design a ~~program~~ that returns the double of a given number.

def double_number(n : float) → float:

Problem: Design a function that takes a person's name and returns a greeting message.

def greet_me(name : str) → str:

Style Guide: The Function name, parameter name should be lower case (and multiple words are separated by underscore)


1.2 Purpose

The purpose of a function describes what the function does. It includes what the function takes and produces in a readable form.

It is written in triple quotes, which is also another way of commenting in a code.

Problem: Design a program that returns the double of a given number.

```
@typecheck
def double_number(n : float) → float:
    """
    returns the double of given
    number n.
    """
```



PURPOSE

1.3 Stub

Stub is a line of code which returns a value of the right type. It doesn't need to be the correct, any value from the right type will do.

Problem: Design a program that returns the double of a given number.

```
@typecheck
def double_number(n: float) -> float:
    """ Returns the double of the
        given number n
    """
    return 0.0 # stub
```

STUB



2. Examples

Imagine when you have a function ready, you or someone else going to use it by calling it.

So, Examples are collection of different calls to a function and the result that the function is expected to return for that call.

EXAMPLES



Problem: Design a program that returns the double of a given number.

```
@typecheck
def double_number(n: float) -> float:
    """
    returns the double of given number n.
    """
    return 0    # stub
```

```
start_testing()
expect(double(0), 2 * 0)
expect(double(7), 2 * 7)
expect(double(-1.5), 2 * -1.5)
summary()
```

3. Template

Comment out the body of the stub and copy the body of template from the data definition to the function design.

In case there is no data definition (for example in today's lecture, we are dealing only with primitive types), just copy all parameters.

Problem: Design a program that returns the double of a given number.

```
@typecheck
def double_number(n: float) -> float:
    """
        returns the double of given number n.
    """
    # return 0      # stub
    # return ... (n) # template
```

```
start_testing()
expect(double(0), 2 * 0)
expect(double(7), 2 * 7)
expect(double(-1.5), 2 * -1.5)
summary()
```

4. Code the Function Body

Complete the function body by filling in the template,
note that you have a lot of information written already.
Remember to comment the template.

Problem: Design a program that
returns the double of a given number.

```
@typecheck
def double_number(n: float) -> float:
    """
    returns the double of given number n.
    """
    # return 0      # stub
    # return ...(n)  # template
    return 2 * n
```

```
start_testing()
expect(double(0), 2 * 0)
expect(double(7), 2 * 7)
expect(double(-1.5), 2 * -1.5)
summary()
```

5. Test and Debug until correct

You should run your functions until all tests pass.

Problem: Design a program that returns the double of a given number.

```
@typecheck
def double_number(n: float) -> float:
    """
    returns the double of the given number n
    """
    # return 0      # stub
    # return ...(n) # template
    return 2*n
```

```
start_testing()
expect(double(0), 2 * 0)
expect(double(7), 2 * 7)
expect(double(-1.5), 2 * -1.5)
summary()
```

TEST RESULTS



3 of 3 tests passed

Let's Critique!

WITHOUT HTDF

```
In [6]: 1 def double_number(n):  
2         return 2*n  
3  
4 double_number(8)  
5 double_number(2)
```

Out[6]: 4

- ① Error-prone
- ② Unreliable
- ③ Not easy to read
- ④ Difficult to maintain

WITH HTDF

```
1 @typecheck  
2 def double_number(n: float) -> float:  
3     ...  
4     returns the double of a given number n.  
5     ...  
6     # return 0 #stub  
7     # return ...(n) #template  
8     return 2 * n  
9  
10 start_testing()  
11 expect(double_number(7), 7*2)  
12 expect(double_number(-10.5), -10.5*2)  
13 expect(double_number(0), 0)  
14 summary()  
  
3 of 3 tests passed
```


- ① Well-tested
- ② Well-documented (easy to read)
- ③ Clear definition
- ④ Easy to maintain & evolve.

Worksheet Activity Time!

Let's do
Question 1 – 7

Module 2 (HtDF): Worksheet



Upload a scanned version of your [How to Design Functions worksheet](#) . (For help on how to scan, see [Creating a PDF](#).)

You can also find the Jupyter version of this worksheet on Syzygy in your [module-2-htdf/Worksheet directory](#).

If you choose to not use the Jupyter version of the worksheet, please be aware of the following:

- We reserve the right to refuse to grade non-PDF submissions.
- In order to receive marks for your worksheet submission, we must be able to see the text you have written on the page. If we cannot make out what has been written, you will receive a 0 for your worksheet.

Rest on Jupyter Notebook at:

module2-htdf > Lecture > Ashish > Lecture Python Notebook – Module 2 (HtDF) – Blank.ipynb