Introduction to Coding

Tan Sein Jone

University of British Columbia

August 19, 2024

Table of contents

Table of Contents

What are Functions?

- Functions are used to organize code
- Functions are used to make code reusable
- Functions are used to make code easier to read

Defining Functions

- Functions are defined using the def keyword
- Functions can take arguments
- Functions can return values

```
def my_function(x):
    return x
```

Calling Functions

- Functions are called using the function name
- Functions can be called with or without arguments
- Functions can be called multiple times

```
def my_function(x):
    return x
print(my_function(10))
```

Pseudocode

- Pseudocode is used to plan out code
- Pseudocode is used to break down complex problems
- Pseudocode is used to make code easier to write

```
# Pseudocode
# Define a function that takes a list of numbers as
    an argument
# Iterate over the list of numbers
# If the number is even, add it to a new list
# Return the new list
```

```
def even_numbers(numbers):
    even_numbers = []
    for number in numbers:
        if number % 2 == 0:
            even_numbers.append(number)
    return even_numbers
```

Debugging

- Debugging is the process of finding and fixing errors in code
- Debugging is an important skill for programmers
- It may be frustrating to get an error message, but sometimes not getting one can be worse
- When we get an error message, we can use it to help us find the problem
- When the program runs without errors, but the output is not what we expect, have to use debugging techniques to find the problem

Read Error Messages

- Syntax Errors: Errors in the code structure
- Logic Errors: Errors in the code logic
- Runtime Errors: Errors that occur while the code is running

Rubber Duck Debugging

- Explaining the code to someone else
- Explaining the code to an inanimate object
- Explaining the code to yourself

Table of Contents

Simple Functions

- Let's write a simple function that does only one thing
- This function will take a number as an argument and return the square of that number

```
def square(x):
    return x * x
```

Simple Functions

- Functions don’t necessarily have to take arguments

- They don’t necessarily have to return values

```
def hello():
    print("Hello, World!")
```

When Will We Use Simple Functions?

- When we want to break down a complex problem into smaller parts
- When we want to make our code more readable

Table of Contents

Compound Functions

- You can write functions that call other functions
- This is called a compound function
- Say you want to write a function that first checks if a value is an integer, and then squares it

```
def is_integer(x):
    return type(x) == int

def square(x):
    return x * x

def square_integer(x):
    if is_integer(x):
        return square(x)
    else:
        return "Not an integer"
```

Importance of Modularity

- Modularity is the practice of breaking down code into smaller, more manageable parts
- Modularity makes code easier to read and understand
- Modularity makes code easier to maintain
- Modularity makes code easier to test