Coding and Solutioning

Code Layout, Readability, Reusability

Date	19 November 2022
Team ID	PNT2022TMID51098
Project Name	Real-Time River Water Quality Monitoring and Control System
Maximum Marks	2 Marks

Code Layout, Readability, Reusability Key Ideas

KEY IDEA 1: Code must be simple to comprehend.

KEY IDEA 2: When writing code, try to think of as many applications as you can.

KEY IDEA 3:

Choosing specific terms

- Avoiding general names (or knowing when to use them)
- Using concrete names instead of abstract names
- Adding further information to names using suffixes or prefixes
- Determining how long names should be
- Using name formatting to add extra information.

Select Particular Words

You must avoid using "empty" terms and pick words that are very detailed. For instance, the word "obtain" is quite general, as shown in the example: def GetPage(url):

Searching for more colourful words

Don't be scared to consult a thesaurus or ask a buddy for ideas for better names. There are many terms to choose from in English because it is a rich language.

Example:

send ~ deliver, dispatch, announce, distribute, route find ~ search, extract, locate, recover start ~ launch, create, begin, open make ~ create, set up, build, generate, compose, add, new

KEY IDEA 4: It's preferable to be accurate and straightforward rather than cute.

- Avoid tmp and retval as generic names.
- Instead of using a name that is meaningless, such as this, choose a name that expresses the entity's values or goals.
- Sometimes using a generic name will help you find a bug. temp or tmp

Use concrete names instead of abstract names

Consider that your internal method ServerCanStart() checks the server's ability to listen on a specific TCP/IP port. However, ServerCanStart() is a bit of an ambiguous moniker. CanListenOnPort() would be a more specific moniker (). This name specifically states what the method does. Don't try to combine two concepts that are orthogonal to one another. Follow the Single Responsibility Rule to make naming methods easy.

Other Important Attributes Coding

Knowing that part of the data your application gets is not yet secure leads to many security attacks. You may want to utilise variables with names like unsafeMessageBody or untrustedUrl for this. The variables that are produced after running functions that clean up the dangerous input could be trustedUrl or safeMessageBody. Before further processing, a password should be encrypted because it is currently in "plaintext." Better name for the password: plaintext password.

The ideal length for a name

How do you choose between the names d and days for a variable? The precise application of the variable will determine the response.

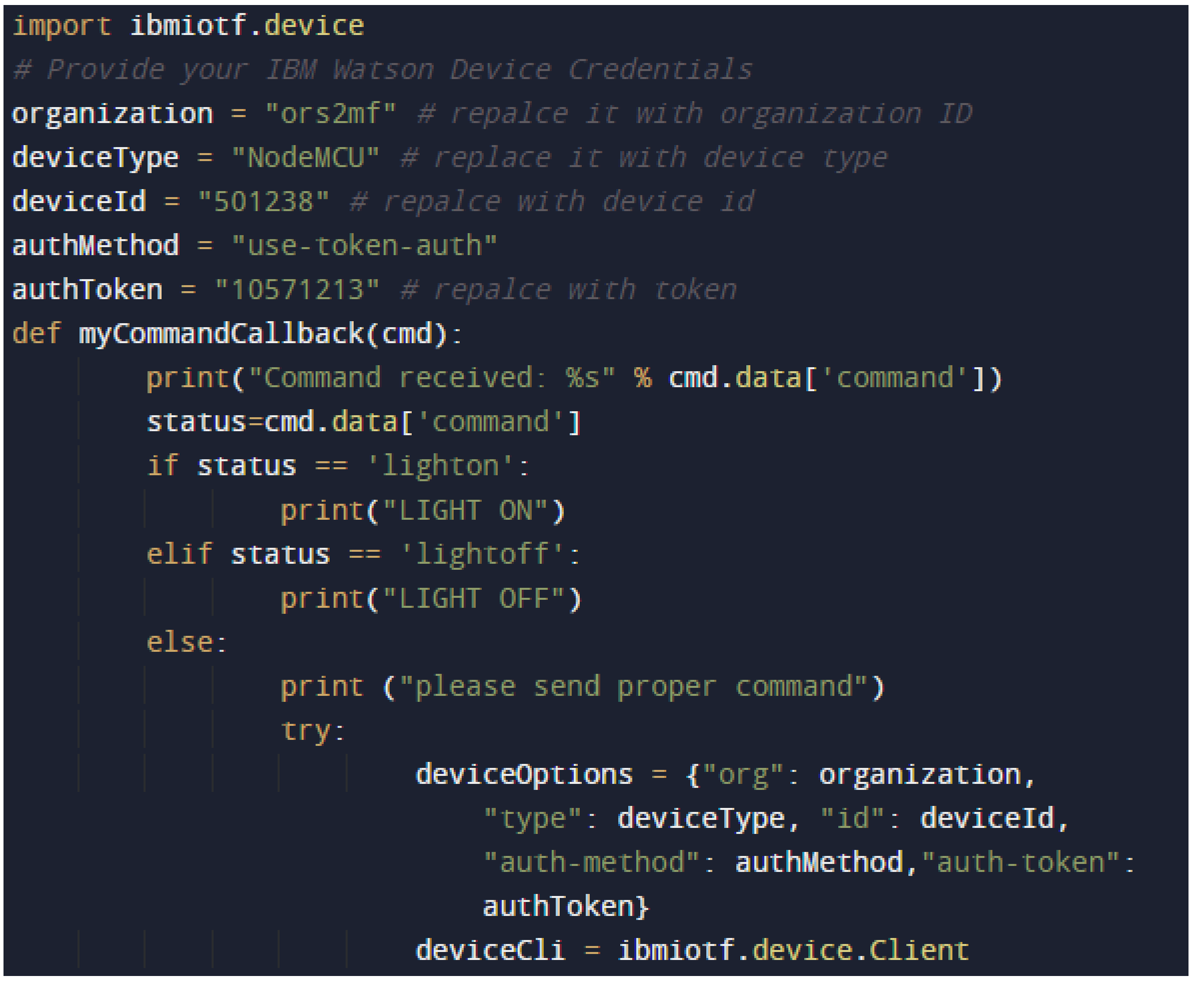
For a Limited Scope, Shorter Names Are Acceptable

The amount of information that identifiers need to contain depends on how many other lines of code

```
if (debug) { map<string,int> m; LookUpNamesNumbers(& m);
Print(m);
}
```

Despite the fact that m is devoid of any information, this is not an issue because the reader already has all the knowledge necessary to comprehend this code.

Code Layout



```
import ibmiotf.device
# Provide your IBM Watson Device Credentials
organization = "ors2mf" # repalce it with organization ID
deviceType = "NodeMCU" # replace it with device type
deviceId = "501238" # repalce with device id
authMethod = "use-token-auth"
authToken = "10571213" # repalce with token
def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data['command'])
        status=cmd.data['command']
        if status == 'lighton':
                print("LIGHT ON")
        elif status == 'lightoff':
                print("LIGHT OFF")
        else:
                print ("please send proper command")
                try:
                        deviceOptions = {"org": organization,
                            "type": deviceType, "id": deviceId,
                            "auth-method": authMethod,"auth-token":
                            authToken}
                        deviceCli = ibmiotf.device.Client
```

Readability & Reusability

- Readable code is code that clearly communicates its intention to the reader.
- This code is easy to read and understand everything faster.
- Pre-existing code can be recycled to perform the same function or repurposed to perform a similar but slightly different function. Code reusability **increases productivity reduces costs, and improves overall quality**.
- We can reuse every part of the code.