Good Software Design

Beyond just delivering functionalities

yangsu.dev

What is considered ***good***

- Readability
- Flexibility
- Etc.

How?

- Understand the context
- Break it down
- Build and Test
- Iterate

Example

- Your manager wants to be alerted via text message when dogecoin price goes above 1 CAD.

Possible Implementation

```
def send dogecoin alert to boss():
    crypto url = "https://crypto-price.com/doge-coin"
    twilio_client = TwilioClient("secret_sdk_key")
    while True:
        page = requests.get(crypto url)
        soup = BeautifulSoup(page.content, "html.parser")
        result = soup.find(id="dogecoin")
        doge_coin_price = float(result.text.strip())
        if doge coin price > 1:
            twilio client.send text msg("boss-number", "Wake
up, dogecoin is on fire, buy buy buy")
        time.sleep(1)
```

Possible Iteration

```
def send_doge_coin_alert_to_boss(finder:
CryptoPriceFinderInterface, messenger: MessengerInterface):
    while True:
        if finder.get_price('dogecoin') > 1:
            messenger.send("boss-number", "Wake up, dogecoin is on
fire, buy buy buy")
        time.sleep(1)


class TwilioMessenger(MessengerInterface):
    def send(self, number, msg):
        …
        return self.client.send_text_message(number, msg)
class HTMLCryptoPriceFinder(CryptoPriceFinderInterface):
    def get_price(self, abbreviation):
        …
        return float(results.text.strip())
```

In conclusion

Good design is **iterative** and **relative**