Harrisburg University of Science & Technology

CISC 504 Principles of Programming Languages

Assignment 6: The Python Standard Library

Instructions:

- Use as many code cells as you need to implement the tasks in below.
- Submit a Jupyter Notebook (iPython) doc including a 5 minutes walk-through recording (a YouTube recordings is highly recommended.)
- **DO NOT JUST SUBMIT THE NOTEBOOK**

(1) Calculating the Time Elapsed to Run a Loop

In this assignment, you discovered a line of code that was pushed to production that is causing a major delay in the code. The line of code is as follows: `l = [random.randint(1, 999) for _ in range(10 * 3)]` .

For this assignment, given the line of code above and the import statements below, create a way to track the amount of time it takes to perform that line of code. We've used `time.time()` before to track run times. But the time will be so small it iwll be hard to read. Try to find a better function that tracks the number of nanoseconds that have past instead. Try looking at the `time` (https://docs.python.org/3/library/time.html) documentation from Python.

In [8]:
```
import random
import time
```

(2) Testing Python Code

You are given the following code snippet that appears to be crashing the local system that your code is running on. `compile("1" + "+1" * 10 ** 6, "string", "exec")` .

You need to try running this code directly on your system using the Python `sys` and `subprocess` modules. You will need to **run** a **subprocess** and find the system **executable** to execute the code. Try looking up the `sys` (https://docs.python.org/3/library/sys.html) and `subprocess` (https://docs.python.org/3/library/subprocess.html) modules and find the required methods to execute this code as a subprocess on your system.

In [11]:
```
import sys
import subprocess
code = 'compile("1" + "+1" * 10 ** 6, "string", "exec")'
```

(3) Using partial on class Methods

We learned about using `partial` 's in our exercise, but they seem to fall short on class **method**s. You need to discover why the partial on line 11 fails and fix it. Use the `functools` (https://docs.python.org/3/library/functools.html) documentation to find out what needs to be changed.

In [14]:
```
import functools

class Hero:
    DEFAULT_NAME = "Superman"
    def __init__(self):
        self.name = self.DEFAULT_NAME

    def rename(self, new_name):
        self.name = new_name

    reset_name = functools.partial(rename, DEFAULT_NAME)

    def __repr__(self):
        return f"Hero({self.name!r})"
```

In [16]:
```
hero = Hero()
assert hero.name == "Superman"
hero.rename("Batman")
assert hero.name == "Batman"
hero.reset_name()
assert hero.name == "Batman"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-bb67c39af065> in <module>
      3 hero.rename("Batman")
      4 assert hero.name == "Batman"
----> 5 hero.reset_name()
      6 assert hero.name == "Batman"

TypeError: rename() missing 1 required positional argument: 'new_name'
```