Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Fall 2022

Week 10

Let's Discuss

Key Questions

How do I use **dynamically generated streams** of data to implement **memory efficient** and **predictable** Python programs?

Learning Objectives

To **remember** and **understand** some the concept of a **monoid**, seeing how it connects to **practical applications** with strings and sequences

Summations of Large Lists

File: seq_nonGen.py

- Find the sum of all numbers between 1 and n
- Build a list of these numbers in memory and then find sum??
- Note: entire list must held in the memory!

```
# Build and return a list
# ref: https://wiki.python.org/moin/Generators
def listBuilder(n):
    num= 0; nums = []
    while num < n:
        nums.append(num)
        num += 1
    return nums
#end of listBuilder()
sum_of_first_n = sum(listBuilder(1000000))
print("\t The sum of first n :",sum_of_first_n)
```

Summations of Large Lists

File: seq_gen.py

- Suppose we *still* want to find the sum of all numbers between 1 and n but we do not want to use all our memory.
- Generator functions to build the list and get each value as requested

Part 1 of 2

```
# Using the generator pattern (an iterable)
# ref: https://wiki.python.org/moin/Generators
class listBuilder(object):
    def __init__(self, n):
        self.n = n
        self.num, self.nums = 0, []

    def __iter__(self):
        return self
```

Summations of Large Lists

File: seq_gen.py

Part 2 of 2

```
    def __next__(self):
        return self.next()

    def next(self):
        if self.num < self.n:
            cur, self.num = self.num, self.num+1
            return cur
        else:
            raise StopIteration()

sum_of_first_n = sum(listBuilder(1000000))
print("\t The sum of first n :", sum_of_first_n)
```

Let's Use Poetry

Let's Code!

Poetry

Project to Make

Project Guide Lines

- Want to make am infinite palindrome sequence generator.

Format

- Format: ABCBA

Examples of palindromes

- 11,22, 33, ...
- 2824282, 2825282, 2826282, 2827282, 2828282, ...
- 478874, 479974, 480084, 481184, 482284, 483384, ...,
- 6513156, 6514156, 6515156, 6516156, 6517156, ...

Setup Steps

Remember these slides from Week 5?!

Make a working directory

```
mkdir projects
cd projects
```

Use Poetry to create new project

```
poetry new makepal
cd makepal
```

Add Project Dependencies

```
poetry add typer
poetry add rich
```

Add Project Development Dependencies

```
poetry add -D black mypy
```

Mypy: `http://mypy-lang.org/`

Setup Steps

Add File: projects/makepal/makepal/main.py

```
"""Required docstring for an __init__ file."""

__version__ = "0.1.0"
```

Add File: projects/makepal/pyproject.toml

```
[tool.poetry] ...

[tool.poetry.scripts]
makepal = "makepal.main:cli"

[tool.poetry.dependencies] ...
```

Update Poetry

```
poetry install
```

Add File: projects/makepal/makepal/main.py

Use file located in sandbox: `main.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from rich.console import Console
import typer

# create a Typer object to support the command-line interface
cli = typer.Typer()
```

Add File: projects/makepal/makepal/main.py

Use file located in sandbox: `main.py`

```
@cli.command()
def main(upperbounds: str = ""):
    """Driver function. Upperbounds
    is how high we go to create palendromes"""

    for i in infinite_sequence():
        pal = is_palindrome(i)
        if pal:
            # print(f"\t {count}, {pal}")
            print(f"{pal}")

# end of main()
```

Add File: projects/makepal/makepal/main.py

Use file located in sandbox: `main.py`

```
def is_palindrome(num):
    # Skip single-digit inputs
    if num // 10 == 0:  # return an int, not a float
        return False
    temp = num
    reversed_num = 0

    while temp != 0:
        reversed_num = (reversed_num * 10) + (temp % 10)
        temp = temp // 10

    if num == reversed_num:
        return num
    else:
        return False

# end of is_palindrome()
```

Add File: projects/makepal/makepal/main.py

Use file located in sandbox: `main.py`

```
def infinite_sequence() -> None:
    """Infinite_sequence
    will eventually stop at an upperbounds"""

    num = 0
    count = 0
    while True:
        yield num
        num += 1

# end of infinite_sequence()
```

Basic Reformatting with Black

Clean up your code with Black

```
poetry run black hello_user tests
```

```
obonhamcarter@MacBookPro-2017 makepal % python3 -m black main.py
reformatted main.py

All done! ✨ 🍰 ✨
1 file reformatted.
```

Execute Project

What do you see?

```
# run from projects/makepal/
poetry run makepal --help
```

```
obonhamcarter@MacBookPro-2017 makePal % poetry run makepal --help

 Usage: makepal [OPTIONS]

 Driver function. Upperbounds is how high we go to create palendromes

 Options
 --upperbounds                   TEXT
 --install-completion            [bash|zsh|fish|powershell  Install completion for the
                                 |pwsh]                    specified shell.
                                                           [default: None]
 --show-completion               [bash|zsh|fish|powershell  Show completion for the
                                 |pwsh]                    specified shell, to copy it
                                                           or customize the
                                                           installation.
                                                           [default: None]
 --help                                                    Show this message and exit.
```

Execute Project

What do you see?

```
# run from projects/makepal/
poetry run makepal
```

```
780087
781187
782287
783387
784487
^C
Aborted.
```

- Our project could race on forever!
- How can we change the code to add an upperbound so that eventually, the code will terminate?