CMPT 423/820

Assignment 1 Question 2 - Model Solution and Grading

Python Basics

- Data: integers, floating point, strings, Boolean, None
  - All data in Python is stored as objects, even numeric values
  - integers in Python are not limited in size
  - floating point values in Python are standard IEEE floating point values, and are limited in range.
  - These basic data types are immutable
  - type conversion functions `int()` `float()` `str()`
- Variables and expressions:
  - Every variable in Python stores a reference to an object
  - assignment statements
  - Arithmetic: `+ * - / // %`
  - Relational `== < > <= >= !=`
  - Boolean: `and or not`
- Sequential data structures: lists, tuples
  - Lists are mutable, tuples are immutable
  - Integer indexing for lists and tuples
  - Slicing lists creates new list object
- Dictionaries (associational)
  - Mutable
  - indexable by integers, strings, or tuples containing immutable data
- Python Control Flow
  - if-statements
  - for loops
  - while loops
  - function definitions
    - Since all variables store references, all Python parameters are pass-as-reference
    - By default, a function returns the value `None`.
- Console output
  - `print()`
  - String method `format()`

Tasks

Task 1.

Write Python code to calculate Fibonacci numbers. Use this code to produce the following output ("console output" using the function `print()`):

```
The 111 th Fibonacci number is 70492524767089125814114
As a float, this number is 7.049252476708912e+22
The integer difference between the two values is: 1634146
```

For the sake of this question, assume that the Fibonacci sequence starts with 0, 1, 1, 2, ... Also assume that the 0th Fibonacci number is 0, and the 3rd fibonacci number is 2.

Notice that the difference is non-zero. The integer value is exact. The floating point value is inexact due to truncation).

Model Solution

The purpose of this question is to demonstrate Python's infinite precision integer values. Most languages limit integers to a large but finite range. In Python, an integer has as many bits as you need to store your integer exactly. However, Python floating point values are standard. There is a finite range, and a finite precision. The difference between a largish Fibonacci number as an integer and as a float highlights this difference.

Several Fibonacci implementations are possible, including recursive functions. However, the implementation derived from a plain reading of the mathematical definition, i.e., as follows:

```
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n-1) + fib(n-2)
```

This implmentation is highly inefficient, and cannot return an answer in time!

Good, plausible solutions can use recursion (but not the above implementation). The implementation is not important, as long as it works.

The implementation below uses two variables to remember two values in the sequence, and simply advances the two variables along the sequence.

In [2]:
```
# which Fib number do you want?
n = 111

# Storing two of the numbers in the sequence
f1 = 0
f2 = 1

# just walk down the sequence, advancing f1, f2
i = 1
while i < n:
    # tuple assignment!
    f1, f2 = f2, f1 + f2
    i += 1

print('The', n, 'th Fibonacci number is', f2)
g = float(f2)
print('As a float, this number is', g)
print('The integer difference between the two values is:', f2 - int(g)
)
```

```
The 111 th Fibonacci number is 70492524767089125814114
As a float, this number is 7.049252476708912e+22
The integer difference between the two values is: 1634146
```

Task 2. Lists and slicing

Write Python to

1. Store the first 50 Fibonacci numbers in a list.
2. Use this list, and slicing, to construct a list containing only 30th-34th Fibonacci numbers.
3. Display the sublist
4. Using list indexing, modify the sublist making the first entry `12345`
5. Display the sublist again
6. Display the corresponding sequence from the original list.

Here's an example of the output:

```
The sublist of Fibonacci numbers (before): [832040, 1346269, 2178309, 3524
578, 5702887]
The sublist of Fibonacci numbers (after): [12345, 1346269, 2178309, 352457
8, 5702887]
The original sequence of Fibonacci numbers: [832040, 1346269, 2178309, 352
4578, 5702887]
```

Slicing is really really useful. This requirement is a very simple use of the concept.

Note: The Numpy module has a different, but similar operation, called a view. Python slicing creates a new object, always. Numpy views are not new objects.

Model Solution

This exercise shows that Python slicing creates new lists.

The algorithm below also shows that Python lists grow as long as they need to be, using `append()` .

In [4]:
```python
# the list of Fib numbers
fibs = [0,1]

# how many?
n = 49

# accumulating them
i = 1
while i < n:
    fibs.append(fibs[i-1]+fibs[i])
    i += 1

# slicing the list
subfibs = fibs[30:35]
print('The sublist of Fibonacci numbers (before):', subfibs)

# modifying the sublist
subfibs[0] = 12345
print('The sublist of Fibonacci numbers (after):', subfibs)

# show the original list unchanged
print('The original sequence of Fibonacci numbers:', fibs[30:35])
```

```
The sublist of Fibonacci numbers (before): [832040, 1346269, 2178309
, 3524578, 5702887]
The sublist of Fibonacci numbers (after): [12345, 1346269, 2178309,
3524578, 5702887]
The original sequence of Fibonacci numbers: [832040, 1346269, 217830
9, 3524578, 5702887]
```

What to hand in

Your version of this notebook named A1Q2.pdf, containing completed work above, and your name and student number at the top.

Evaluation:

- 2 marks: Your code cell for Task 1 uses Python (only -- no imported modules) to show the given output.
- 2 marks: Your code cell for Task 2 uses Python (only -- no imported modules) to show the given output

Grading:

Since Fibonacci numbers can be indexed in different ways (some people start with $F_0 = 1$, and that's okay), the actual value of the Fibonacci numbers doesn't matter. Off-by-one results here are good enough!

- **Task 1**: 2 marks: Give full marks unless:
  - The implementation used an imported module
  - The implementation uses the bad version of Fibonacci.
  - The output is missing from the PDF.
- **Task 2**: 2 marks: Give full marks unless:
  - The implementation used an imported module
  - The implementation uses the bad version of Fibonacci.
  - The output is missing from the PDF.