HW2_Part1

April 24, 2020

1 Stats 21 - HW 2 - Part 1 - Charles Liu (304804942)

This is your second homework assignment.

There are two parts:

- HW2 Part 1 (this document)
- HW2 Part 2 (the other with turtle graphics)

The questions have been entered into this document. You will modify the document by entering your code.

Make sure you run the cell so the requested output is visible. Download the finished document as either a PDF or an HTML file.

You will submit:

- the rendered HTML/PDF file of HW Part 1
- this ipynb file with your answers of HW Part 1
- the ipynb file with your answers of HW Part 2
- a document with the screenshots requested in HW Part 2

1.1 Textbook Chapter 5 Problems

1.1.1 Exercise 5.1

[1]:
```
import time
```

[2]:
```
time.time()
```

[2]:
```
1587726773.0248725
```

Write a function `now()` that reads the current time and prints out the time of day in hours, minutes, and seconds, plus the number of days since the epoch. The function does not need to return a value, just print output to the screen.

The result should look like:

"Current time is: 15:25:47. It has been 18370 days since the epoch."

Use `int()` to drop decimal values. You do not need to try to find the date with years and months.

Tip: build your function incrementally. Start by finding how many days have passed since the epoch. (check your answer at the bottom of the page: https://www.epochconverter.com/seconds-days-since-y0) From there find how many hours, etc. Keep in mind the hours will be UTC time.

[3]:
```
def now(current_time):
    days_epoch = current_time // (24 * 60 * 60)
    full_day = days_epoch * (24 * 60 * 60)
    seconds_remain = current_time - full_day
    hours = seconds_remain // (60 * 60)
    minutes = ((seconds_remain) - (hours * 60 * 60)) // 60
    seconds = seconds_remain - ((hours * 60 * 60) + (minutes * 60))
    time_hms = "%d: %d: %d" % (hours, minutes, seconds)
    print("Current time is: ", time_hms, ". It has been", days_epoch, "days␣
 ↪since the epoch.")
```

[4]:
```
now(time.time())
```

```
Current time is:  11: 12: 53 . It has been 18376.0 days since epoch.
```

[5]:
```
now(1587683939.1453102)
```

```
Current time is:  23: 18: 59 . It has been 18375.0 days since epoch.
```

1.2 Textbook Chapter 6 Problems

1.2.1 Exercise 6.2

The Ackermann function, A(m, n), is defined:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

See http://en.wikipedia.org/wiki/Ackermann_function . Write a function named `ack` that evaluates the Ackermann function. Use your function to evaluate a few test cases. Don't test with $m \geq 4$ as it grows very fast very quickly.

[6]:
```
def ack(m,n):
    if(m == 0):
        return(n + 1)
    if((m > 0) and (n == 0)):
        return(ack(m - 1, 1))
    return(ack(m - 1, ack(m, n - 1)))
```

[7]:
```
# test case, should be 61
ack(3, 3)
```

[7]:
```
61
```

[8]:
```
# test case, should be 125
ack(3, 4)
```

[8]:
```
125
```

1.2.2 Exercise 6.3

A palindrome is a word that is spelled the same backward and forward, like "noon" and "redivider". Recursively, a word is a palindrome if the first and last letters are the same and the middle is a palindrome.

[9]:
```
def first(word):
    return word[0]
def last(word):
    return word[-1]
def middle(word):
    return word[1:-1]
```

Create a recursive function `is_palindrome(word, verbose)` that accepts two arguments:

- `word` is a string and the function returns `True` if it is a palindrome and `False` otherwise.
- `verbose` is a boolean value (default value False). If `verbose` is True, the function will print out all of the recursive calls that are being made. See the example on page 60 in section 6.9 for an example.

You can use the built-in function `len()` to check the length of a string.

[10]:
```
def is_palindrome(word, verbose = False):
    if(len(word) <= 1):
        return(True)
    if(first(word) != last(word)):
        return(False)
    return(is_palindrome(middle(word)))
```

[11]:
```
is_palindrome("racecar")
```

[11]:
```
True
```

[12]:
```
is_palindrome("stats")
```

[12]:
```
True
```

[13]:
```
is_palindrome("abcdefghigfedcba", True)
```

[13]:
```
False
```

[14]:
```
is_palindrome("abcdefghgfedcba", True)
```

[14]:
```
True
```

1.2.3 Exercise 6.4

A number, `a`, is a power of `b` if it is divisible by `b` and `a/b` is a power of `b`. Write a function called `is_power` that takes parameters `a` and `b` and returns `True` if a is a power of b. Note: you will have to think about the base case.

[15]:
```
def is_power(a, b):
    c = a/b
    if(((a % b) == 0) and ((c % b) == 0)):
        return(True)
    else:
        return(False)
```

[16]:
```
is_power(1024, 2)
```

[16]: True

[17]:
```
is_power(6561, 3)
```

[17]: True

[18]:
```
is_power(4374, 3)
```

[18]: True

[19]:
```
is_power(768, 2)
```

[19]: True

1.3 Exercise 6.5

The greatest common divisor (GCD) of a and b is the largest number that divides both of them with no remainder.

One way to find the GCD of two numbers is based on the observation that if `r` is the remainder when `a` is divided by `b`, then `gcd(a, b) = gcd(b, r)`.

As a base case, we can use `gcd(a, 0) = a`.

Write a function called `gcd` that takes parameters `a` and `b` and returns their greatest common divisor.

[20]:
```
def gcd(a, b):
    if(b == 0):
        return(a)
    r = (a % b)
    return(gcd(b, r))
```

[21]:
```
gcd(21, 7)
```

[21]: 7

[22]:
```
gcd(42, 28)
```

[22]: 14

[23]:
```
gcd(105, 140)
```

[23]: 35