Homework 9

Due Nov 2nd, 2020, 4PM

In [100]:
```
first_name = "Scott"
last_name = "Urista"

assert(len(first_name) != 0)
assert(len(last_name)  != 0)
```

1) Point

Modify class Point defined below to provide working versions of **str**() and **eq**().

Edit the class so that two Points with the same x and y are the same, and so that points are printed as tuples.

Printing

```
one = Point(3, 4)
print(one)
```

Should produce:

```
(3, 4)
```

Double Equals

```
one = Point(3, 4)
two = Point(3, 4)
print(one == two)
```

Should produce:

```
True
```

In [101]:
```
class Point(object):
    """Represents a point in 2-D space."""

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return '({self.x}, {self.y})'.format(self = self)


    def __eq__(self, other):
        return (self.x, self.y) == (other.x, other.y)
```

Unit Test for Point

In [102]:
```
def test_point():
    p = Point(3, 4)
    q = Point(3, 4)

    assert p.__str__() == '(3, 4)', "Should yield (3 4)"
    assert p == q, "Should be equal"

    print('Success')

test_point()
```

```
Success
```

2) Collatz sequence

The Collatz sequence, also know as the Hailstone sequence, is a sequence of numbers.

If the current number is n, the next number is n / 2 if n is even, and 3n + 1 if n is odd.

It has not been shown that there isn't a sequence which never repeats.
All known sequences end by repeating 4, 2, 1, 4, 2, 1, ...

Write a generator collatz(n) **that starts at n** and generates the rest of the sequence down to 1.
Your generator should raise a StopIteration exception after yielding 1.

In [103]:
```
def collatz(n):
    "generates the Collatz sequence"

    if n == 1:
        raise StopIteration
    while n != 1:
        yield n
        n = (n * 3 + 1) if n % 2 else (n // 2)
    yield 1
```

Unit Tests

In [104]:
```
import string
def test_collatz():
    g = collatz(4)
    lst = [n for n in g]
    assert lst == [4, 2, 1]

    g = collatz(11)
    lst = [n for n in g]
    assert lst == [11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]

    g = collatz(29)
    lst = [n for n in g]
    assert lst == [29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5,

    print('success!')

test_collatz()
```

```
success!
```

3) Next Month

Write a generator that will return a sequence of month names. Thus

```
gen = next_month('October')
```

creates a generator that generates the strings 'November', 'December', 'January' and so on.
If the caller supplies an illegal month name, your function should raise a ValueError exception with text explaining the problem.

In [127]:
```
month_names = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'Dec

def next_month(name: str) -> str:
    "Return a stream of the following months"
    if name not in month_names:
        raise ValueError("Not a valid month name")
    month = month_names.index(name) + 1
    while True:
        month = (month % 12)
        yield month_names[month]
        month += 1
```

Unit Tests

In [125]:
```
def test_months():
    gen = next_month('October')
    lst = [next(gen) for i in range(15)]
    assert(lst == ['November', 'December', 'January', 'February', 'March',

    gen = next_month('December')
    assert next(gen) == 'January'
test_months()
```

The following should raise a ValueError with text explaining the problem

In [126]:
```
try: # This should throw a Value Error
    gen = next_month('Thermador')

    m = next(gen)

    print(1/0)
except ValueError:
    print('Success!')
```

```
Success!
```

4) Phone Numbers

Modify the class below that takes a string and returns an object holding a valid NANP phone number. You will need to fill in the three methods listed, but underfined, below: __str__(), area_code(), and normalize().

The "North American Numbering Plan" (NANP) is a telephone numbering system used by many countries in North America. All NANP-countries share the same international country code: `1`.

NANP numbers are ten-digit numbers consisting of a three-digit area code and a seven-digit local number. The first three digits of the local number are the "exchange code", and the four-digit number which follows is the "subscriber number".

The format is usually represented as (NXX)-NXX-XTooXXX where `N` is any digit from 2 through 9 and `X` is any digit from 0 through 9.

Your task is to clean up differently formatted telephone numbers by removing punctuation, such as '(', '-', and the like, and removing and the country code (1) if present.

Start by stripping non-digits, and then see if the digits match the pattern. If you are asked to create a phone number that does not meet the pattern above, you should throw a ValueError with a string explaining the problem: too many or too few digits, or the wrong digits.

For example, the strings below

+1 (617) 495-4024

617-495-4024

1 617 495 4024

617.495.4024

should all produce an object that is printed as (617) 495-4024

ValueErrors

Each of the following strings should produce a ValueError exception.

+1 (617) 495-40247 has too many digits

(617) 495-402 has too few digits

+2 (617) 495-4024 has the wrong country code

(017) 495-4024 has an illegal area code

(617) 195-4024 has an illegal exchange code

In [108]:
```
class Phone:
    "A Class defining valid Phone Numbers"

    def __init__(self, raw):
        "Create new instance"
        self.number = self._normalize(raw)

    def __str__(self) -> str:
        "Create printable representation"
        return '('+self.number[0:3]+') '+self.number[3:6]+'-'+self.number


    def area_code(self) -> str:
        "Return the area code"
        return self.number[0:3]


    def _normalize(self, raw: str) -> str:
        """Take string presented and return string with digits
            Throws a ValueError Exception if not an NANP number"""
        pnumber = ''.join([char for char in raw if char.isdigit()])
        if len(pnumber) == 11 and pnumber[0] != '1':
            raise ValueError("invalid country code")
        if pnumber[0] == '1':
            pnumber = pnumber[1:]
        if len(pnumber) > 10:
            raise ValueError("invalid - too many digits")
        if len(pnumber) < 10:
            raise ValueError("invalid - too few digits")
        if pnumber[0] < '2':
            raise ValueError("invalid area code")
        if pnumber[3] < '2':
            raise ValueError("invalid exchange code")
        return pnumber
```

Unit Tests for Phone Number

In [109]:
```
def test_phone():
    p = Phone('+1 (617) 495-4024')
    assert(p.__str__() == '(617) 495-4024')

    p = Phone('617-495-4024')
    assert(p.__str__() == '(617) 495-4024')

    p = Phone('1 617 495 4024')
    assert(p.__str__() == '(617) 495-4024')

    p = Phone('617.495.4024')
    assert(p.__str__() == '(617) 495-4024')
    assert(p.area_code() == '617')


    p = Phone('+1 (508) 495  4024')
    assert(p.__str__() == '(508) 495-4024')

    p = Phone('508 - 495 - 4024')
    assert(p.__str__() == '(508) 495-4024')

    p = Phone('1 508 (495) [4024]')
    assert(p.__str__() == '(508) 495-4024')

    p = Phone('508!495?4024')
    assert(p.__str__() == '(508) 495-4024')
    assert(p.area_code() == '508')


    print("Success!")

test_phone()
```

```
Success!
```

Unit Tests for invalid numbers - each should raise a ValueError with a different string

There are 6 different problems below: you should throw Value errors with 6 different explainations

In [110]:
```
p = Phone('+1 (617) 495-40247')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
st)
<ipython-input-110-db4ce11c1cff> in <module>
----> 1 p = Phone('+1 (617) 495-40247')

<ipython-input-108-e1b453be7980> in __init__(self, raw)
      4     def __init__(self, raw):
      5         "Create new instance"
----> 6         self.number = self._normalize(raw)
      7
      8     def __str__(self) -> str:

<ipython-input-108-e1b453be7980> in _normalize(self, raw)
     25             pnumber = pnumber[1:]
     26         if len(pnumber) > 10:
---> 27             raise ValueError("invalid - too many digits")
     28         if len(pnumber) < 10:
     29             raise ValueError("invalid - too few digits")

ValueError: invalid - too many digits
```

In [111]:
```
p = Phone('(617) 495-402')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
st)
<ipython-input-111-692bc6319a88> in <module>
----> 1 p = Phone('(617) 495-402')

<ipython-input-108-e1b453be7980> in __init__(self, raw)
      4     def __init__(self, raw):
      5         "Create new instance"
----> 6         self.number = self._normalize(raw)
      7
      8     def __str__(self) -> str:

<ipython-input-108-e1b453be7980> in _normalize(self, raw)
     27             raise ValueError("invalid - too many digits")
     28         if len(pnumber) < 10:
---> 29             raise ValueError("invalid - too few digits")
     30         if pnumber[0] < '2':
     31             raise ValueError("invalid area code")

ValueError: invalid - too few digits
```

In [112]:
```
p = Phone('+2 (617) 495-4024')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
st)
<ipython-input-112-233260c5c685> in <module>
----> 1 p = Phone('+2 (617) 495-4024')

<ipython-input-108-e1b453be7980> in __init__(self, raw)
      4     def __init__(self, raw):
      5         "Create new instance"
----> 6         self.number = self._normalize(raw)
      7
      8     def __str__(self) -> str:

<ipython-input-108-e1b453be7980> in _normalize(self, raw)
     21         pnumber = ''.join([char for char in raw if char.isdigit(
)])
     22         if len(pnumber) == 11 and pnumber[0] != '1':
---> 23             raise ValueError("invalid country code")
     24         if pnumber[0] == '1':
     25             pnumber = pnumber[1:]

ValueError: invalid country code
```

In [113]:
```
p = Phone('(017) 495-4024')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
st)
<ipython-input-113-225915713b41> in <module>
----> 1 p = Phone('(017) 495-4024')

<ipython-input-108-e1b453be7980> in __init__(self, raw)
      4     def __init__(self, raw):
      5         "Create new instance"
----> 6         self.number = self._normalize(raw)
      7
      8     def __str__(self) -> str:

<ipython-input-108-e1b453be7980> in _normalize(self, raw)
     29             raise ValueError("invalid - too few digits")
     30         if pnumber[0] < '2':
---> 31             raise ValueError("invalid area code")
     32         if pnumber[3] < '2':
     33             raise ValueError("invalid exchange code")

ValueError: invalid area code
```

In [114]:
```
p = Phone('(617) 195-4024')
```

```
---------------------------------------------------------------------------
---
ValueError                                Traceback (most recent call la
st)
<ipython-input-114-895781462c15> in <module>
----> 1 p = Phone('(617) 195-4024')

<ipython-input-108-e1b453be7980> in __init__(self, raw)
      4     def __init__(self, raw):
      5         "Create new instance"
----> 6         self.number = self._normalize(raw)
      7
      8     def __str__(self) -> str:

<ipython-input-108-e1b453be7980> in _normalize(self, raw)
     31             raise ValueError("invalid area code")
     32         if pnumber[3] < '2':
---> 33             raise ValueError("invalid exchange code")
     34         return pnumber

ValueError: invalid exchange code
```

In [ ]: