Python Curriculum

Part 02 - Logic Controls (2/2)

Error Controls and Controlled Errors

try/except

```
>>> 1 / 0
Traceback (most recent call last):
  ...
ZeroDivisionError: division by zero
```

```
>>> try:
...     1 / 0
... except:
...     pass
```

```
>>> a = 0
>>> b = '5'
>>> try:
...     b / a
... except ZeroDivisionError:
...     pass
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

```
>>> b = 'bob'
>>> a = '5'
>>> try:
...     c = b / a
... except ZeroDivisionError:
...     # handle zero division error when a == 0
...     c = a / b
... except TypeError:
...     # handle type error
...     try:
...         c = int(b) / int(a)
...     except ValueError:
...         print('we cannot perform {0} ({1}) / {2} ({3}'.format(
...             b, type(b), a, type(a)))
... except:
...     # theoretical last resort to catch any other errors
...     pass
...
we cannot perform bob (<class 'str'>) / 5 (<class 'str'>
```

Remember `hours_from()`?

```
'''module: utility.py'''
def hours_from(x, y):
    try:
        x = int(x)
        y = int(y)
    except ValueError:
        return None

    from_x = x + y  # unbound y hours from x
    from_x = str(from_x % 24)  # 24-hour capped hours from x, then cast to str
    z = from_x.zfill(2) + ':00'  # left-pad and format hours from x as HH:00
    return z  # return the value of z
```

```
>>> import utility
>>> utility.hours_from(16, 12345)  # utility module's hours_from() function
'01:00'
>>> utility.hours_from('16:00', 12345)
>>> # None, null, nil, nothing
```

Controlled Errors

```
'''module: utility.py'''
def hours_from(x, y):
    try:
        x = int(x)
        y = int(y)
    except ValueError:
        raise Exception('x and y need to be real numbers or base-10 number strings')

    from_x = x + y  # unbound y hours from x
    from_x = str(from_x % 24)  # 24-hour capped hours from x, then cast to str
    z = from_x.zfill(2) + ':00'  # left-pad and format hours from x as HH:00
    return z  # return the value of z
```

```
>>> from utility import hours_from  # cherry-pick hours_from() function from utility module
>>> hours_from('16:00', 123)
Traceback (most recent call last):
  ...
    x = int(x)
ValueError: invalid literal for int() with base 10: '16:00'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  ...
    raise Exception('x and y need to be real numbers or base-10 number strings')
Exception: x and y need to be real numbers or base-10 number strings
```

```
def hours_from(x, y):
    try:
        x = int(x)
        y = int(y)
    except ValueError:
        raise

    from_x = x + y  # unbound y hours from x
    from_x = str(from_x % 24)  # 24-hour capped hours from x, then cast to str
    z = from_x.zfill(2) + ':00'  # left-pad and format hours from x as HH:00
    return z  # return the value of z
```

```
def hours_from(x, y):
    x = int(x)
    y = int(y)

    from_x = x + y  # unbound y hours from x
    from_x = str(from_x % 24)  # 24-hour capped hours from x, then cast to str
    z = from_x.zfill(2) + ':00'  # left-pad and format hours from x as HH:00
    return z  # return the value of z
```

```
>>> from utility import hours_from
>>> hours_from('16:00', 12345)
Traceback (most recent call last):
  ...
    x = int(x)
ValueError: invalid literal for int() with base 10: '16:00'
```

Assertion

```
>>> from utility import hours_from
>>> assert hours_from(16, 12345) == '01:00'  # nothing, good
>>> assert hours_from(16, 12345) != '16:00'  # nothing, good
>>> assert hours_from(16, 12345) == '16:00'
Traceback (most recent call last):
  ...
AssertionError
```

if/else

vs

try/except

Questions?