Lecture 14 (2-16-26)

Python (Data Structures, Arguments, I/O)

Reviewing and adding to `fizzbuzz.py` from last lecture:

```
#!/usr/bin/env python3

def fizzbuzz(start=1, end=100): #setting default parameters
    #docstring and "unit test":
    ''' Print fizzbuss from start to end.

    >>> fizzbuzz(1,5).
    1
    2
    Fizz
    4
    Buzz
    '''
    #the above is essentailly a unit test, you can do:
        # `python3 -m doctest - vfizzbuzz.py` to run the test

    for number in range(start, end + 1):
        if number % 15 == 0:
            print('FizzBuzz')
        elif number % 3 == 0:
            print('Fizz')
        elif number % 5 == 0:
            print('Buzz')
        else:
            print(number)

def main():
    fizzbuzz(90)

# this conditional does not run the module when you import
if __name__ == '__main__':
    main()
```

- if you call `fizzbuzz(90)` then you will get 90 to 100
- now you can do `import fizzbuzz` to use this in other scripts
- `type(fizzbuzz)` will return `<class 'module'>`
- you can do `help(fizzbuzz.fizzbuzz)`
    - this returns the docstring
    - first `fizzbuzz` is for the module
    - second `fizzbuzz` is for the function
- once you import the module, then you can call the `fizzbuzz` function by:
    - `fizzbuzz.fizzbuzz(5,15)`
- you can also do `from fizzbuzz import fizzbuzz`
    - now you can just do `fizzbuzz(5,15)` to call the function now

Static analysis

```
#!/usr/bin/env python3

import sys

def fizzbuzz(start int=1, end: int=100): -> None: #setting default
parameters
    #docstring and "unit test":
    ''' Print fizzbuss from start to end.

    >>> fizzbuzz(1,5).
    1
    2
    Fizz
    4
    Buzz
    '''
    #the above is essentailly a unit test, you can do:
        # `python3 -m doctest - vfizzbuzz.py` to run the test

    for number in range(start, end + 1):
        if number % 15 == 0:
            print('FizzBuzz')
        elif number % 3 == 0:
            print('Fizz')
        elif number % 5 == 0:
            print('Buzz')
```

```
            else:
                print(number)

def main(arguments:  list[str]=sys.argv[1:]) -> None: #example of slicing
the argv list and using the
    '''print fizzbuzz from first two command line arguments

    >>>main(['1','5'])
    1
    2
    Fizz
    4
    Buzz
    '''
    start = int(arguments[0])
    end = int(arguments[1])

    print(f'Start is {start}')
    print(f'End is {end}')

    fizzbuzz(start,end)

# this conditional does not run the module when you import
if __name__ == '__main__':
    main()
```

- `mypy` never runs your code, it simply statically evaluates it
  - `mypy fizzzbuzz.py`
- this is static analysis, it can save a lot of resources
- when python sees an error is gives an exception
- you can catch these exception errors
- its basically like this:
  - "try doing this, if this error is thrown do this instead"

```
#!/usr/bin/env python3

import sys

def fizzbuzz(start int=1, end: int=100): -> None: #setting default
parameters
    #docstring and "unit test":
```

```
    ''' Print fizzbuss from start to end.

    >>> fizzbuzz(1,5).
    1
    2
    Fizz
    4
    Buzz
    '''
    #the above is essentailly a unit test, you can do:
        # `python3 -m doctest - vfizzbuzz.py` to run the test

    for number in range(start, end + 1):
        if number % 15 == 0:
            print('FizzBuzz')
        elif number % 3 == 0:
            print('Fizz')
        elif number % 5 == 0:
            print('Buzz')
        else:
            print(number)

def main(arguments:  list[str]=sys.argv[1:]) -> None: #example of slicing
the argv list and using the
    '''print fizzbuzz from first two command line arguments

    >>>main(['1','5'])
    1
    2
    Fizz
    4
    Buzz
    '''

    try:
        start = int(arguments[0])
    except IndexError:
        start = 1

    try:
        end = int(arguments[1])
    except:
        end = 100


    fizzbuzz(start,end)
```

```
# this conditional does not run the module when you import
if __name__ == '__main__':
    main()
```

explicitly checking to bypass try and except using ternary operator

```
#!/usr/bin/env python3

import sys

def fizzbuzz(start int=1, end: int=100): -> None: #setting default
parameters
    #docstring and "unit test":
    ''' Print fizzbuss from start to end.

    >>> fizzbuzz(1,5).
    1
    2
    Fizz
    4
    Buzz
    '''
    #the above is essentailly a unit test, you can do:
        # `python3 -m doctest - vfizzbuzz.py` to run the test

    for number in range(start, end + 1):
        if number % 15 == 0:
            print('FizzBuzz')
        elif number % 3 == 0:
            print('Fizz')
        elif number % 5 == 0:
            print('Buzz')
        else:
            print(number)

def main(arguments:  list[str]=sys.argv[1:]) -> None: #example of slicing
the argv list and using the
    '''print fizzbuzz from first two command line arguments

    >>>main(['1','5'])
    1
```

```
    2
    Fizz
    4
    Buzz
    '''

    #ternary operators
    start = int(arguments[0]) if len(arguments) >= 1 else 1
    end = int(arguments[1]) if len(arguments) >= else 100


    fizzbuzz(start,end)

# this conditional does not run the module when you import
if __name__ == '__main__':
    main()
```

Unit tests

- we will be provided unit tests for each assignment
  - usually executed via `./fizzzbuzz.test -v`
- will learn how to write unit tests in future courses

---

returning a list in `fizzbuzz.py`

```
#!/usr/bin/env python3

import sys

def fizzbuzz(start int=1, end: int=100): -> list[str]: #setting default
parameters
    #docstring and "unit test":
    ''' Print fizzbuss from start to end.

    >>> fizzbuzz(1,5)
    ['1','2','Fizz', '4', 'Buzz']
    '''
```

```
    #the above is essentailly a unit test, you can do:
        # `python3 -m doctest - vfizzbuzz.py` to run the test
    results = []
    for number in range(start, end + 1):
        if number % 15 == 0:
            results.append('FizzBuzz')
        elif number % 3 == 0:
            results.append('Fizz')
        elif number % 5 == 0:
            results.append('Buzz')
        else:
            results.append(str(number))

    return results

def main(arguments:  list[str]=sys.argv[1:]) -> None: #example of slicing
the argv list and using the
    '''print fizzbuzz from first two command line arguments

    >>>main(['1','5'])
    1
    2
    Fizz
    4
    Buzz
    '''

    #ternary operators
    start = int(arguments[0]) if len(arguments) >= 1 else 1
    end = int(arguments[1]) if len(arguments) >= else 100


    for number in fizbuzzz(start,end):
        print(number)

    '''
        #or you could have done this instead of the for loop above

        print('\n'.join(fizzbuzz(start,end)))
    '''
# this conditional does not run the module when you import
if __name__ == '__main__':
    main()
```