SLDC (Software Development Life Cycle)

Requirements Analysis:
Design:
Development:
Testing:
Deployment:
Maintenance

Python Coding Random Notes:

Miscellaneous

- ** - To the power of. E.g 3**3 = 9
- / - Divides the number completely e.g 100 / 3 = 33.33333333336
- // - Divides the numbers and rounds down to the floor integer e.g 27/4 = 6
- sep=', ' - Separates the variables in the string (e.g with a comma and space)
- f" - Recommended way to print , formats everything in string that has {} within it
- Can multiply print statement e.g print("Hi!" * 10) would print "Hi!" 10 times without using loop
- pass - #do nothing. Function or statement will execute but have return of None
- type- type(variable) will return type for variable. Useful for checking input
- Strings behave as lists, every char is an element
- Startswith (useful)

Sys (argc, argv)

- Must *import sys* in order to use argv
- sys.argv - returns a list with all stuff (strings) entered into terminal (after python3)
- argc = len(sys.argv) sets variable named argc to be the length of list of argv

Tuples

- Behave like lists but cannot modify elements
- E.g (1,2) or (1,) or (1, 2, 3) or ("hi", "hello", "hey")

Slicing and indexing (lists or strings)

E.g : lst = [1, 2, 3, 4]
slice = lst[0:3]

- Variable slice will be a list contain elements from range index:0 element to index:3 element
- slice = [1, 2, 3]
- Slicing does not raise error for slicing in range greater than length of the list

- Can negative index (will loop back) e.g lst[-1] = 4 (useful for finding final element)
- We can remove trailing space or whitespace with slices e.g sentence = " Hi hello"
  sentence = sentence[1:] (take everything from 1th index onwards)
  sentence = "Hi hello"

Dictionaries

- Are not ordered like strings,lists,tuples, rather a key maps to a value
- dict.get(key) returns None if key not found , whereas dict["key"] gives keyerror
- dict["key"] = dict.get(key, 0) would set key to be 0 initally although if key is updated (not initial value) calling dict.get(key, 0) would just return the updated value (useful)

JSON

- A text file
- Has the form of a dictionary although all whitespace is ignored and no trailing commas
- Uses double quotes
-

Class and Objects

- Attributes are *values* inside objects
- Methods are *functions* inside objects
- Classes are *blueprints for objects* like how def is blueprint for function ???
- Have two ways to invoke it
  - e.g C = Circle(3) (from lab)
  - C.circumference() = Circle.circumference(C)
- *Writing Classes e.g*
  - Class enter_name:
    - def __init__(self, param1, param2):
      - self.value1 = param1 + param2
      - self.value2 = param2
    - def enter_method_name(self):
      - return self.value1 + something

Flask

- **Writing a flask server in python**

    - App = Flask(__name__)
    - *Writing routes:*
      @APP.route('/enter_some_rout_name', methods = ['enter_valid_method']
      def some_function():
          return 'enter_some_value'
    - if __name__ == '__main__':
          APP.run(port = 0)
- Methods : GET, POST, PUT, DELETE
- Only routes using GET method work when trying to run using web browser URLS

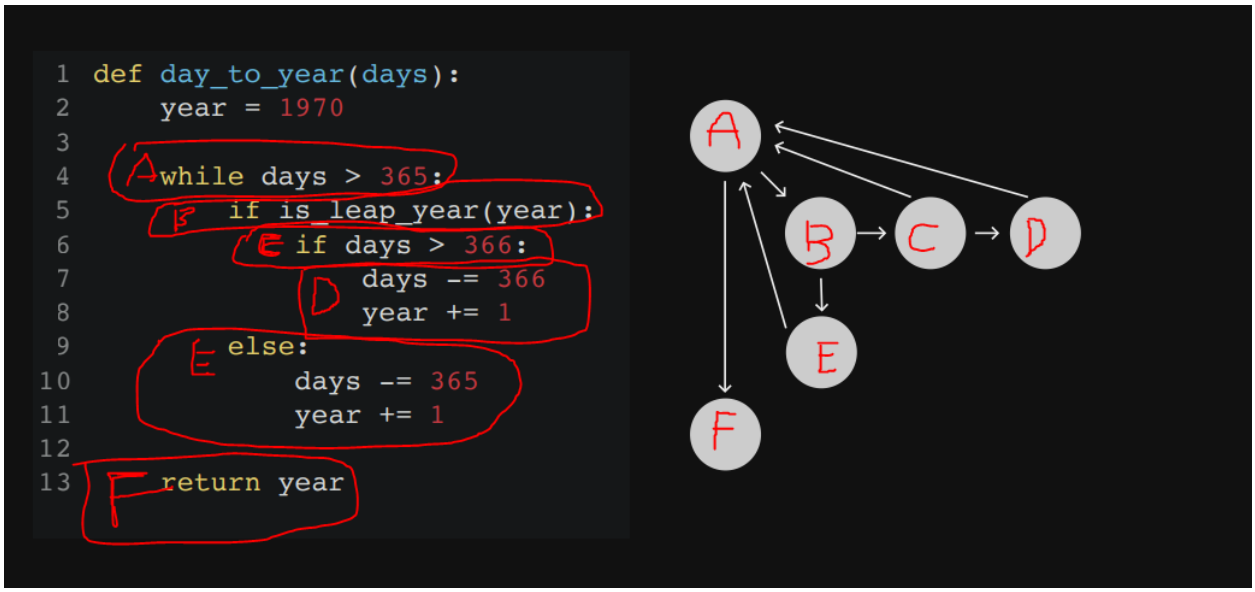
- **Running routes on web URL**
    - Copy paste the url given in the terminal
    - Type in the route which is follow by '/'
    - *Inputting parameters for functions corresponding to the route:*
        - Parameters inputted will be of type string, and return needs to be of type dict, tuple or string (hence will need to convert return value)
        - If parameter(s) listed in the function as an argument we can just type in the url
          ?parameter1='some_value'&parameter2='some_value'
        - If there are no arguments in the defined function, we will have to use the *request library* and write the following code in the function
          E.g request.args.get('parameter1')
          Can input parameters in the url query as normal after that as
        - request.args.getlist("parameter"): If taking in parameters to be put into a list
    - *Outputting (returning):* A good way to do this is to use dumps(some_return) , using from json import dumps

- **Running routes on ARC Client (FOR FINAL EXAM)**
- Since we cannot use routes using methods POST, PUT, DELETE on web url, we use ARC client
- Using the body:
    - For request.form.get("parameter") switch the body encoded type to form and input parameter and its value
    - For data = request.get_json switch body encoded type to application json and input json
        - Json string must use double commas, cannot use whitespace or trailing commas

- **Testing routes**
  - Import requests
  - resp = requests.enter_route_method.(f"{url}enter_route", data = {"parameter": "value"}
  - resp should be 200 if working, 400 for input error, 404 for access error, 500 for not working
  - Payload = resp.json()
  - Use assert to check that payload is the return value of the route

Cyclomatic Complexity

- Cyclomatic complexity = number_edges - number_nodes + 2
- Separate the code into buckets or chunks of code ran
- E.g



Decorators

Generators