Exam 1 Summary Sheet

Annabelle Adachi

Chapter 1: Introduction to Python

- **High level language** - easier to program and read, can run on different types of computers, must be interpreted first before being processed by computer
- **Program** - instructions for performing a computation
  - Input - data from keyboard, file, etc.
  - Output - display or send out data
  - Math and logic - programs can perform basic mathematical and logical operations
  - Conditional execution - check for conditions and execute statements
  - Repetition - perform actions repeatedly
- **Bug** - programming error
  - Debugging: fixing programming errors
  - Types of bugs:
    - Syntax errors - incorrect structure, like a grammatical error
    - Runtime errors - doesn't appear until program is run, rare
    - Semantic errors - program will run successfully, but results aren't what you wanted
- **Comment** - text meant for other people reading the code, doesn't get interpreted by the computer

Chapter 2: Simple Python Data

- **Value/object** - word or number that program can manipulate
  - Categorized into classes: integers, strings, floats - type function tells what class an object is
  - Can convert from one class to another using type conversion functions
- **Variable** - name that refers to a value, assigned using =
  - No spaces in variable names
  - Have to begin with letter or underscore
  - Can include numbers
  - Can't have other characters like $
  - Can't be same as a Python keyword(words used to define Python rules)

    - Variables can be reassigned or updated
- **Statement** - instruction for Python interpreter to execute
- **Expression** - combination of values, variables, operators, calls to functions
- **Operators** - mathematical tokens, follows regular order of operations
    - * to multiply, ** for exponents
    - // integer division, result is rounded down to an integer
    - / division operator, result is floating point value
    - % remainder, provides integer remainder
- **Operand** - values the operator works on
- **Prompt string** - user input
    - Input function returns string, if you want a different class you have to use type conversion function

Chapter 3: Debugging

- **Debugging** is one of the most important skills in programming
- Much of debugging is avoidable:
    - Understand the problem you are trying to solve
    - Start small and keep improving
- Error messages
    - **ParseError**: syntax error - missing punctuation, etc.
    - **TypeError**: two objects that aren't compatible get combined (e.g. adding an integer and a string)
    - **NameError**: variable is used before it has been given a value, often caused by typos
    - **ValueError**: parameter passed to a function is outside function limitations/incompatible with function
- Tips for understanding error messages:
    - Comment out lines with error messages
    - Add in print statements to check values and types as you go
    - Work backwards to find the first occurrence of a problem

Chapter 4: Python Turtle Graphics

- Each different turtle is an instance with its own name
- **Iteration**: use a **for loop** for repetitive motions to simplify program

    - **Loop variable**: variable whose value cycles through the items in the given list
    - **Loop body**: indented, performed once for each iteration/item in list
    - **Terminating condition**: if no items left in list to cycle through, ends the loop and continues to next statement after loop
    - Flow of control not just top to bottom in loop
    - For loop is a compound statement
    - Use **range** function instead of list to further simplify for loop
        - For x in range(4) = for x in [0, 1, 2, 3]
- Negative angles or distances used to express opposite motion: forward(-100) = back(100) and right(-90) = left(270)
- Use penup()/up() and pendown()/down() to stop turtle drawing to move turtle around without leaving a trail
- Turtles can have different shapes: arrow, blank, circle, etc.
- Speed of turtle can be edited using speed() - between 1(slowest) and 10(fastest)
    - Setting speed to 0 will turn off animation and just make the turtle draw as fast as possible
- Color and fill can be edited
- Turtle can stamp its shape onto the screen using stamp()

Chapter 5: Python Modules

- **Module**: file containing definitions and statements for use in other Python programs
- Many modules come with Python in the **standard library**(e.g. turtle!)
- **Import** a module to use
- **Global Module Index** - alphabetical list of all modules available in the standard library
- Math module: contains mathematical functions and constants
- Random module: generate random numbers, pick random items from list, anything to simulate random-ness
- Create modules to use in other python files - import modules using name of file
    - Leave comments in module to explain what it does
    - Use **dot notation** to call functions from modules

Chapter 6: Functions

- **Function**: sequence of statements grouped together; a type of **compound statement**

- Functions help organize programs, break code into logical parts where each function is responsible for a specific task
- Syntax for function definition:
  - def name(parameter1, parameter2):
            statements
- **Function call/function invocation**: run a function with chosen parameters
- **Return** statement tells function what to return when it's called, terminates call to function
- **Global variables**: variables defined outside of any functions
- **Local variables**: temporary variables that are defined inside a function, only exist within a function
  - Parameters of a function are local variables
- Unit Testing
  - **Unit test**: test to check if parts of code/functions are working properly, collection of unit tests called a test suite
  - **Test case**: check certain requirements for a program using example parameters for a function
  - **Assert** checks whether expression returns the provided correct value, stops the program and produces an error message if expression evaluates to False, continues program if assert evaluates to True
  - Use assert with for loop to run checks on items in a list
- **Accumulator pattern**: loop that redefines variable according to a pattern(e.g. x increases by 1 every iteration of the loop)
- Functions can call other functions - **composition** is process of building functions using other functions