# Midterm Topics List

Midterm Date: 24-March-2022 9:00 a.m.

## Important

- You may bring a memory aid: 1 sheet, can be double-sided, letter size (8.5"x11") **Typed and printed** in English. Your name and ID must be typed on the sheet. Each student is expected to prepare their own cheat sheet and submit it with exam booklet at the end of the exam.
- Simple Calculators approved by the college are allowed

## List of Material Covered in class and labs

- Lecture 1: Computer Data & Numbering Systems.
- Data Representation Examples: Additional Examples on Binary and Hex conversion
- Lecture 2: Python 3 Basics
- Lecture 3: while Loop & if statement
- Lecture 4: strings
- Lecture 5: Lists & for loop
- Lecture 6: Review of Iterables
- Lecture 7: Booleans
- Lecture 8: Functions
- Labs 1-7 & T1-Review-Lab

## Tokens

- Recognize and know how to use these reserved words:

| | | | | | |
|---|---|---|---|---|---|
| `and` | `def` | `else` | `if` | `None` | `return` |
| `break` | `del` | `False` | `in` | `not` | `True` |
| `continue` | `elif` | `for` | `is` | `or` | `while` |

- Integers - Decimal digits *only*.
- Floats - Decimal digits with decimal point and optional exponent.
- Remember valid formats for integer and float
- Strings - Single, double, and triple quoted. Know the sequence '\n' in a string.
- Variable Names - Consist of letters, digits, or underscores, must begin with letter or underscore, case-sensitive.
- Lists - Lists of integers, floats, strings, mixed lists, 2-dimensional lists.

## Expressions

- Binary math operators: addition (+), subtraction (-), multiplication (*), remainder (%), floor division (//), real division (/), and exponentiation (**).
- Unary math operator: negation (-).
- Parentheses.
- Rule for promotion of integer to float.
- Comparison operators: ==, !=, <, >, <=, >=. result is `True` or `False`.
- chaining comparisons, i.e. 2 <= x < 5
- Basic operator precedence: exponentiation is evaluated first, then multiplication/division/remainder, then addition/subtraction.
- Index expressions: `lst[0]`, `lst[-1]`, `string[j-1]`, etc.
- Slice expressions: `lst[i:j + 1]`, `string[1:]`, `lst[::-1]`, etc.
- Other operators: `is`, `in`.
- Builtin Function calls, e.g. `len(x)`, `max(x, y)`, `min(x, y)`, `sum(lst)`
- Method calls, e.g. index, find, append, extend, insert
- Boolean expressions, e.g. (x>=5) and usage with while and if

## Statements

- Assignment (e.g. =, +=, //=, *=).
- `if`/`elif`/`else`
- `break` - Exit enclosing loop.
- `continue` - Skip to next iteration.
- Expression (e.g. call to `print()`)
- `for` - For every item in an iterable, repeat statements.
- `if`/`elif`/`else` - Choose one of several actions.
- `while` - If a condition is true, repeat statements.
- nested `while` and nested `for`

## Types

- Understand the terms *sequence* and *iterable*.
- Basic operators on sequences (* for repetition and + for concatenation, comparison operators).
- Mutable (lists) vs. immutable (all other types so far).
- Know the types and type functions *covered so far*:

  `bool()` - `True` or `False`. Conversion of other types to boolean values.

  `float()` - Convert integer or string to float.

  `int()` - Convert float or string to integer.

  `bin()` - Convert decimal or hex to binary.

  `hex()` - Convert decimal or binary to hexadecimal.

  `list()` - Convert other iterable to list.

  Using range to create a list. List can also be created with comma-separated list of expressions surrounded by square brackets.

  `str()` - Convert other values to string.

  `tuple()` - Convert iterable to tuple. Can be also created from comma-separated list of expressions inside parentheses.

- `bool` - `True` or `False`. Conversion of other types to boolean values, rules for results of boolean expressions.
- `float` - A floating-point number, `float()` will convert a `str` or `int` to a `float`.

- `int` - An integer, `int()` will convert a `str` or `float` to an `int`.
- `list` - A mutable sequence (or array) of values. A literal list is a comma-separated sequence of expressions surrounded by square brackets. Use of operators for comparison >, <, >=, <=, ==, !=. Concatenation with +, repetition with *. Important methods:

| | | | | | |
|---|---|---|---|---|---|
| `append()` | `copy()` | `extend()` | `insert()` | | `remove()` |
| `clear()` | `count()` | `index()` | `pop()` | `reverse()` | `sort()` |

- `str` - Immutable text. `str()` will convert any type to a string for output. Important methods:
  - `format()` - Basics of width, precision, and conversion types `'d'`, `'e'`, `'f'`, and `'s'`.
  - `split()`, `join()` - Convert to or from a list of strings.
  - `index()`, `rindex()`, `find()`, `rfind()` - Search for a substring. Know the difference.
  - `upper()`, `lower()`: convert to upper or lower case.
  - `isupper()`, `islower()`, `isalpha()`, `isdigit()`, `isalnum()`: check for type of string
  - Comparison rules, e.g. know why these are both `True`:

```
"Apple" != "apple", "grape" < "grapefruit"
```

- `tuple` - An immutable list. Literal tuples are normally enclosed inside parentheses. Single-element tuple requires trailing comma. Operators same as with `list`. Two methods: `index()` and `count()`

## Important Note about String and List Methods

- Must be careful with the type of argument(s) that need to be passed to each method
- Always check if the method returns a value. Many methods do not have a return value. They return **None**.

## Built-in functions

Basics of these functions as covered in class.

| | | | | |
|---|---|---|---|---|
| `abs()` | `input()` | `max()` | `range()` | `sorted()` |
| `all()` | `len()` | `min()` | `reversed()` | `sum()` |
| `any()` | `list()` | `print()` | `round()` | `type()` |

## User-defined functions

- parameters and arguments
- calling a function
- return statement or statements
- Scope rules: local variables and global variables

## Algorithms

- Know the algorithm (manual) to convert a non-negative decimal integer to binary or hexadecimal.
- Use of a list of lists to represent a simple 2-dimensional matrix and how to access matrix elements.
- Understand the concept of a validation algorithm with many conditions, e.g. password check.