

# CSCA08F18

Albion Fung, Sonika Verma

[github.com/conanap](https://github.com/conanap)



# Basic Types

- int
- float
- bool
- NoneTypes



# Object Types

- str
- function / method
- Other objects types



# Operator & Operands



# Arithmetic Operators

- `**`
- `*`, `/`, `//`, `%`
- `+`, `-`



# Boolean Operators

- `()`
- `>`, `<`, `>=`, `<=`, `==`, `!=`, `in`
- `not`
- `and`, `or`



```
evaluate (not 4 / 2 > 3 + 5 ** 2) == (not (3+ 5) ** 2 / 4 + 54 / 3 - 10 < 2)
```



# Strings

- Single quotes (')
- double quotes(" not 2 single quotes)
- triple quotes (three of only single or double quotes)



# Strings

- `a = '01234'`
- `a[0]`
- `a[-1]`
- `a[:3]`
- `a[1:]`
- `a[:-1]`



# Operators on Strings

- Compare with <, <=, >, >=, ==, !=
- \*



# Functions & Methods

- A short hand name for a bunch of code
  - may or may not use input
  - may or may not have output
- Function: standalone object
- Method: function part of an object



# Function Design Recipe

- Examples
- Header with Type Annotation
- Description
- Body
- Test



# Docstring

- Description
- Precondition
- Examples



# Type Annotations

- Include type of each parameter
- Annotate return type
- `def summation(first: int, second: int) -> int:`



```
def area(width: float, length: float) -> float:
```

```
    """Return the product of width and length, and print the string 'Hi'"""
```

Examples:

```
>>> area(3.0, 2.0)
```

```
Hi
```

```
6.0
```

```
>>> area(0.1, 2.0)
```

```
Hi
```

```
0.2
```

```
"""
```

```
    print('Hi')
```

```
    return width * length
```



# Nested Functions Calls

- Can call a function in a function
- Can call a function as an argument



# Rules of Evaluation

- Left to right
- Variables and literals
- Operands then operators
- Arguments
- [Nested] functions / method calls



$$x = 24$$

$$\min(\max(x, 15 + 2), 44, 32 / 3) / 4 + 2$$



$\max(44, \sqrt{144} / 2 + 5) * \min(15, 5^{**2})$



# Range Function

- Used in for loops (usually)
- `range([start,] stop [, step])`



# For Loops

- `for i in range(start, stop, step)`
- Includes start but excludes end
- Changes i by step for each iteration
- Can be negative!



# if

- Give you an option to run or not run the code
- if boolean expression



# elif

- Requires an if statement to be used
- Only executed if preceding if statement is false
- elif boolean expression:



# else

- Must be used in conjunction with an if statement
- Code ran only if all if and elif statements are false
- else:



# Memory Model