- Math Operators from Highest to Lowest Precedence
- Common Data Types
- Comparison operators
- String concatenation and replication
- Common functions
- Binary Boolean Operators
    - The not Operator
- Def Statements with Parameters
- List


### Math Operators from Highest to Lowest Precedence

| Operator | Operation | Example | Evaluates to... |
|---|---|---|---|
| ** | Exponent | 2 ** 3 | 8 |
| % | Modulus/remainder | 22 % 8 | 6 |
| // | Integer division/floored quotient | 22 // 8 | 2 |
| / | Division | 22 / 8 | 2.75 |
| * | Multiplication | 3 * 5 | 15 |
| - | Subtraction | 5- 2 | 3 |
| + | Addition | 2+ 2 | 4 |

### Common Data Types

| Data type | Examples |
|---|---|
| Integers | -2, -1, 0, 1, 2, 3, 4, 5 |
| Floating-point numbers | -1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25 |
| Strings | 'a', 'aa', 'aaa', 'Hello!', '11 cats' |

### Comparison operators

Comparison operators compare two values and evaluate down to a single Boolean value.

| Operator | Meaning |
|---|---|
| == | Equal to |
| != | Not equal to |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |

### String concatenation and replication

```
>>> 'Alice' + 'Bob'
'AliceBob'
```

```
>>> 'Alice' * 5
'AliceAliceAliceAliceAlice'
```

# Common functions

### The print() function

The print() function displays the string value inside the parentheses on the screen.

```
print('Hello world!')
print('What is your name?') # ask for their name
```

### The input() function

The input() function waits for the user to type some text on the keyboard and press enter.

```
myName = input()
```

### The len() function

You can pass the len() function a string value (or a variable containing a string), and the function evaluates to the integer value of the number of characters in that string.

```
>>> len('hello')
5
>>> len('My very energetic monster just scarfed nachos.')
46
>>> len('')
0
```

# Binary Boolean Operators

A truth table shows every possible result of a Boolean operator. Table 2-2 is the truth table for the and operator.

| Expression | Evaluates to... |
|---|---|
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |

## The not Operator

Unlike and and or, the not operator operates on only one Boolean value (or expression). The not operator simply evaluates to the opposite Boolean value.

| Expression | Evaluates to... |
|---|---|
| not True | False |
| not False | True |

# Def Statements with Parameters

When you call the len() function and pass it an argument such as 'Hello', the function call evaluates to the integer value 5, which is the length of the string you passed it. In general,

the value that a function call evaluates to is called the return value of the function. When creating a function using the def statement, you can specify what the return value should be with a return statement. A return statement con- sists of the following:

- The *return* keyword
- The value or expression that the function should return

When an expression is used with a return statement, the return value is what this expression evaluates to. For example, the following program defines a function that returns a different string depending on what num- ber it is passed as an argument.

```
import random

def getAnswer(answerNumber):
        if answerNumber == 1:
                return "It is certain"
        elif answerNumber == 2:
           return 'It is decidedly so'
       elif answerNumber == 3:
           return 'Yes'
       elif answerNumber == 4:
           return 'Reply hazy try again'
       elif answerNumber == 5:
           return 'Ask again later'
       elif answerNumber == 6:
           return 'Concentrate and ask again'
       elif answerNumber == 7:
           return 'My reply is no'
       elif answerNumber == 8:
           return 'Outlook not so good'
       elif answerNumber == 9:
           return 'Very doubtful'

r = random.randint(1, 9)
fortune = getAnswer(r)
print(fortune)
```

# List