**IDX G9 CS S STUDY GUIDE ISSUE 2**

**By Joyce**

**Lesson 08: If Statement**

**Boolean Values**

* Boolean values have only two possible values: True or False.
* Examples:
  + '5' + '6' == '56' evaluates to False.
  + 1 != '1' evaluates to True.
  + 8 >= 8 evaluates to True.

**Relational Operators**

* Operators and their descriptions with examples:
  + < Less than: 2\*\*3 < 9 evaluates to True.
  + > Greater than: 0 > 1 evaluates to False.
  + <= Less than or Equal to: 4 <= 5 evaluates to True.
  + >= Greater than or Equal to: 9\*\*0.5 >= 3 evaluates to True.
  + == Equal to: 1 == 2 evaluates to False.
  + != Not Equal to: 'B' != 'B' evaluates to False.

**If Statement**

* Syntax:

if condition:

statements

**Example**

ph = float(input('Enter the pH level: '))

if ph < 7.0:

print(str(ph), "is acidic.")

if ph > 7.0:

print(str(ph), "is alkaline.")

**If … Else … Statement**

* Syntax:

if condition:

statements1

else:

statements2

**Example**

n = int(input())

line = "Today, I ate " + str(n) + " apple"

if n > 1:

line += "s."

else:

line += "."

print(line)

**If … Elif … Else … Statement**

Syntax:

if condition1:

statements1

elif condition2:

statements2

else:

statements3

**Example**

age = int(input("What is your age? "))

if age < 11:

print("You are a kid.")

elif age < 20:

print("You are a teenager.")

else:

print("You are an adult.")

**Lesson 09: Boolean Operators**

**Relational Operators**

* Operators and their descriptions with examples:
  + < Less than: 2\*\*3 < 9 evaluates to True.
  + > Greater than: 0 > 1 evaluates to False.
  + <= Less than or Equal to: 4 <= 5 evaluates to True.
  + >= Greater than or Equal to: 9\*\*0.5 >= 3 evaluates to True.
  + == Equal to: 1 == 2 evaluates to False.
  + != Not Equal to: 'B' != 'B' evaluates to False.

**Boolean Operators**

* There are three types of Boolean operators: not, and, or.
* Like relational operators, they evaluate expressions down to a Boolean value.

**Truth Table**

* A truth table shows all possible results of a Boolean operator.

**Truth Table of and Operator**

* The and operator evaluates an expression to True if both Boolean values are True; otherwise, it evaluates to False.
  + Expression | Evaluates to
  + True and True | True
  + True and False | False
  + False and True | False
  + False and False | False

**Truth Table of or Operator**

* The or operator evaluates an expression to True if either of the two Boolean values is True; otherwise, it evaluates to False.
  + Expression | Evaluates to
  + True or True | True
  + True or False | True
  + False or True | True
  + False or False | False

**Truth Table of not Operator**

* The not operator evaluates to the opposite Boolean value.
  + Expression | Evaluates to
  + not True | False
  + not False | True

**Precedence**

* Operator precedence follows this order: () > not > and > or.
* Example expression: (a > b and c > d) or (a > b and a !> c) or (not b == c)?

**Boolean Expression Example**

* Expressions that evaluate to True:
  + 2 >= 2 and 1 < 3
  + 1 <= 2 or 1 != 1
  + not “A” == “a”
* Expressions that evaluate to False:
  + not 1 == 1
  + “a” == “A” or “b” == “B”
  + 3 > 4 and 3 < 5

**Remainder**

* Returns the remainder after dividing the first operand by the second operand.
  + Example: 22 % 5
  + math.floor(123 / 10) % 10
  + Consider: -13 % 3?

**Integer Division**

* Returns the floor value of the quotient produced by dividing the two operands.
  + Example: (1234 // 100) % 10
  + Consider: -14//3?

**Exercise**

* Evaluate the following Boolean expressions:
  + “T” == “t” or 4 > 1
  + 13 % 5 > 3 and not 3 == 2
  + (3 \* 5) % (12 / 2) >= 9 \*\* (1 / 2)
  + 9 > 3 \* 3 or not 11 – 5 < 6 and 2 + 5 >= 7
  + 2 + 2 == 4 and not 2 + 2 == 5 and 2 \* 2 == 2 + 2
  + not 2 > 1 and 3 < 4 or 5 > 5 and 2 > 1 and 3 >= 3

**Precedence of Common Operators**

* Operator | Description
  + \*\* | Exponentiation (raise to the power)
  + \* / % // | Multiply, divide, modulo, and integer division
  + + - | Addition and subtraction
  + <= < > >= != == | Relational operators
  + not and or | Boolean operators

**Lesson 10: Number System**

**Decimal Number System**

* The decimal system is a positional system that uses 10 digits (0-9) to represent numbers.
* Base-10: Each place value in a decimal number is a power of 10.
* Example: 635210 assumes base-10 if the base is omitted.

**Binary Number System**

* The binary system is widely used in computer science.
* It consists of only two digits: 0 and 1.
* Base-2: Each place value in a binary number is a power of 2.
* Example: 1011102 requires the base number to be specified.

**Bits and Bytes**

* Each 0 or 1 is called a bit (binary digit).
* Transmission speeds are often measured in bits per second (b/s).
* A group of eight bits is called a Byte, usually abbreviated as an uppercase B.
* Storage space is typically measured in bytes.

**Decimal to Binary Conversion**

* Example conversion of 23 to binary:
  + Dividend: 23, Divisor: 2
  + Quotient: 11, Remainder: 1
  + Quotient: 5, Remainder: 1
  + Quotient: 2, Remainder: 1
  + Quotient: 1, Remainder: 0
  + Result: 10111B
  + Calculation:1x2^4+0x2^3+1x2^2+1x2^1+1x2^0

**Data Representation & Conversion**

* 1 byte = 8 bits
* 1 KiB = 1024 bytes, 1 MiB = 1024 KiB = 104104 bytes
* Conversion between Binary and Decimal:
  + B to D: 10011101B
  + D to B: 89 over 2, find “remainders”
    - 89 / 2 = 44 remainder 1
    - 44 / 2 = 22 remainder 0
    - 22 / 2 = 11 remainder 0
    - 11 / 2 = 5 remainder 1
    - 5 / 2 = 2 remainder 1
    - 2 / 2 = 1 remainder 0
    - 1 / 2 = 0 remainder 1
    - Result: 1011001B

**Hexadecimal Notation**

* Hexadecimal notation is based on 16.
* Each hexadecimal digit represents 4 bits.

**Hex Conversion**

* Binary to Hex (Hex to Binary):
  + Example: 1A 🡪 26
  + Example: 255 🡪 FF

**Base-N Number Conversion**

* 1234 = X10, what is X? (Solution: 4)
* 123 = X8, what is X? (Solution: 5)
* 678 = X16, what is X? (Solution: 2B in hexadecimal)
* 19x = 25, what is X? (Solution: 5)

**Lesson 12: While-Loop**

* **For Loop vs While Loop**
  + For loops are suitable when the number of iterations is known in advance.
  + While loops are used when the number of iterations is not known beforehand.
* **While Loop Syntax**

While condition:

Statements

* **Using While Loop for Accumulation**

i, s = 1, 0

n = int(input("Enter n: "))

while i <= n:

s += i

i += 1

print(s)

Be cautious to avoid infinite loops.

* **Augmented Assignment Statement**

These are shorthand notations for performing an operation and assignment.

Examples:

* + - x += 2 is equivalent to x = x + 2
    - x -= 2 is equivalent to x = x - 2
    - x \*= 2 is equivalent to x = x \* 2
    - x /= 2 is equivalent to x = x / 2
    - x //= 2 is equivalent to x = x // 2
    - x %= 2 is equivalent to x = x % 2

**Lesson 13: String Manipulation I**

* **strip() Function**
  + - str.strip(): Returns a new string with leading and trailing whitespace characters removed.
    - str.strip(s): Removes leading and trailing occurrences of characters in s.
* **lstrip() and rstrip()**
  + - str.lstrip(): Removes whitespace characters from the left end of a string.
    - str.rstrip(): Removes whitespace characters from the right end of a string.
* **String Indexing**
  + Syntax: str[index]
  + Example: str = 'Hello World!'
    - str[0] → 'H'
    - str[4] → 'o'
    - str[-1] → '!'
* **Membership Operators**
  + x in s: Returns True if x is found in s, otherwise False.
  + x not in s: Returns True if x is not found in s, otherwise False.
  + Example: '+' in '1+2' returns True.
* **len() Function**
  + len(str): Counts and returns the length of the string str.
* **Input Text Cleaning**
  + Clean a string of leading and trailing special characters, including whitespaces.
  + **Sample Input**: ----\*\*\*---- I am a student in SHSID ----\*\*\*------
  + **Sample Output**: I am a student in SHSID
* **String Slicing**
  + Syntax: str[start slice: end slice]
  + Example: str = 'python'
    - str[:-1] → 'pytho'
    - str[-1:] → 'n'
    - str[1:] → 'ython'
    - str[:1] → 'p'

**Lesson 14: String Manipulation II**

* **join() Function**
  + - delimiter.join(list): Concatenates strings in a list using the specified delimiter.
    - delimiter.join(str): Concatenates characters in a string using the specified delimiter.

**split() Function**

* + - str.split(delimiter): Splits a string into a list of substrings based on the delimiter.
    - str.split(): By default, splits on whitespace characters (space, tab, new line).
* **isalnum(), isdigit(), isalpha()**
  + str.isalnum(): Returns True if all characters in the string are alphanumeric, otherwise False.
  + str.isdigit(): Returns True if all characters are digits, otherwise False.
  + str.isalpha(): Returns True if all characters are alphabetic, otherwise False.