**NOTE: This is an official document by Indexademics. Unless otherwise stated, this document may not be accredited to individuals or groups other than the club IDX, nor should this document be distributed, sold, or modified for personal use in any way.**

**IDX G9 COMPUTER SCIENCE H STUDY GUIDE**

**ISSUE 3**

**By Gavin 9-4**

Lesson 12 Nested Loop

In Python, **nested loops** refer to a loop that exists inside another loop.

The basic structure is:

for outer\_variable in outer\_sequence:

for inner\_variable in inner\_sequence:

# Statements

**Implementation 1**

This structure can be utilized to generate a simple multiplication chart:

for i in range(1, 6): # Outer loop

for j in range(1, 6): # Inner loop

print(i \* j, end='\t')

print() # New line

**Output:**

1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

4 8 12 16 20

5 10 15 20 25

The chart prints all possible values of i\*j, where {i,j∈N} and i∈[1,6), j∈[1,6).

**Implementation 2**

The loop structure can also be used to generate a list of stars, with *n* stars on the *nth* line,

rows = 5

i = 0

while i < rows: # Outer loop

j = 0

while j <= i: # Inner loop

print('\*', end=' ') # Print star, with one space after each

j += 1

print() # New line after each row

i += 1

**Escape Sequences**

In Python, *escape sequences* are special characters that allow you to represent certain formatting in strings. For instance:

\n represents a **newline**

**Example Usage**

print("Hello,\nWorld!")

**Result:**

Hello,

World!

\t inserts a **horizontal tab space**

**Example Usage**

print("Name:\tJohn")

print("Age:\t69")

**Result:**

Name: John

Age: 30

Lesson 13 Functions

In Python, functions are blocks of **reusable code** that perform a specific task. They can be categorized into two main types:

**Built-in Functions:** These are functions that come pre-defined in Python. Examples include print(), len(), max(), and many others. They provide essential functionality without the need for additional code.

**User-defined Functions:** These are functions that you create to perform specific tasks, defined using the def keyword followed by the function name and parentheses.

When defining a function, specify parameters (arguments) that allow you to pass data into the function.:

def function\_name(parameter1, parameter2):

# Function body

# Code to execute

**Example**

def greet(name):

print(f"Hello, {name}!")

greet("Alice") # Output: Hello, Alice!

The return statement is used to **exit a function** and **send a value back** to the caller. (If no return statement is specified, the function will return None by default.)

**Example**

def add(a, b):

return a + b

result = add(5, 3)

print(result) # Output: 8

**Implementation: isprime() function**

Now, let’s create a function called isprime that checks if a number is prime.

def isprime(n):

"""Check if a number is prime."""

if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

# Example usage

number = 29

if isprime(number):

print(f"{number} is a prime number.")

else:

print(f"{number} is not a prime number.")

**Explanation**

The code checks every single factor of a number *n*, from 2 to . If no numbers in this range is a factor of n, then the number n is prime. The usage of return True/False in the function gives the *isprime(number)* statement a Boolean value.

Lesson 14 Continue & Break

**Break Statement**

The break statement is used to terminate a loop **prematurely**. When break is executed, the loop stops immediately, and is transferred to the next statement following the loop.

for i in range(5):

if i == 3:

break # Exit the loop when i equals 3

print(i)

In this example, the value of i goes from 0 to 4, yet when the value was 3, the for loop was exited due to the break statement, therefore the output of the code would be:

0

1

2

**The continue Statement**

The continue statement is used to skip the current iteration of a loop and move to the next iteration. When continue is executed, the remaining code inside the loop for that iteration is skipped.

for i in range(5)

if i == 3:

continue # Skip the rest of the loop when i equals 3

print(i)

In this example, when i = 3, the continue statement merely exited the current loop iteration, or in other words, moved on to i = 4 without executing the rest of the loop (print statement), and therefore the output would be:

0

1

2

4

Lesson 15 Files

**File Manipulation in Python**

File manipulation in Python allows you to interact with files on your computer, enabling you to read from and write to them. Python provides built-in functions and methods to handle files efficiently.

**Opening a File**

To open a file, you use the open() function, which requires two arguments: the file name and the file mode. The syntax is as follows:

f = open('filename.txt', 'mode')

**File Modes:**

'r': Read mode (default).

Opens a file for reading. If the file does not exist, it raises an error.

'w': Write mode.

Opens a file for writing. If the file exists, it truncates the file (deletes its contents). If it does not exist, a new file is created.

'a': Append mode.

Opens a file for appending. Data is written at the end of the file without truncating it. If the file does not exist, a new file is created.

**Reading from a File**

Once a file is opened in read mode, you can read its contents using several methods:

read():

Reads the entire file content as a single string.

**Example:**

with open('example.txt', 'r') as f:

content = f.read()

print(content)

readline():

Reads the next line from the file. Each call to readline() reads one line.

**Example:**

with open('example.txt', 'r') as f:

line = f.readline()

print(line) # Reads the first line

readlines():

Reads all the lines in a file and returns them as a list of strings.

**Example:**

with open('example.txt', 'r') as f:

lines = f.readlines()

print(lines) # Prints a list of all lines

**Writing to a File**

To write data to a file, you can use the write() method.

This method requires a string argument.

**Example:**

with open('output.txt', 'w') as f:

f.write("Hello, World!\n")

f.write("This is a new line.")

**Closing a File**

It is important to close a file after you are done with it to free up system resources. You can do this using the close() method.

*(There won’t be an error if you don’t close it though. You won’t lose points on the test for this.)*

**Example:**

f = open('example.txt', 'r')

f.close() # Close the file

*Using the with statement (as shown in previous examples) automatically closes the file for you when the block is exited, making it a preferred method for file handling.*

**Splitting Lines**

You can split the contents of a string into a list of lines using the splitlines() method. This is useful when you want to process each line individually.

**Example:**

text = "Line 1\nLine 2\nLine 3"

lines = text.splitlines()

print(lines)

# Output: ['Line 1', 'Line 2', 'Line 3']

**Example Project:**

**Word Count in a .txt file**

This code reads a .txt file, counts the occurrences of each word, and prints the results.

def count\_words\_in\_file(filename):

word\_count = {} # Create an empty dictionary to store word counts

with open(filename, 'r') as file: # Open the file for reading

content = file.read() # Read the entire file content

words = content.split() # Split the content into words based on spaces

# Count the occurrences of each word

for word in words:

word = word.lower() # Convert to lowercase to ensure case insensitivity

if word in word\_count:

word\_count[word] += 1

else:

word\_count[word] = 1

# Print the word counts

for word, count in word\_count.items():

print(f"{word}, {count}")

# Example usage

filename = 'example.txt'

count\_words\_in\_file(filename)

**Example Usage**

The .txt format of “*The Picture of Dorian Gray”* is inputted, and the result is as follows:

(the actual output is too long to put here)

# Parts of the output

the, 2761

project, 84

gutenberg, 25

ebook, 8

of, 1672

picture, 38

dorian, 158

gray, 73

this, 134

is, 574

for, 394

use, 18

anyone, 4

anywhere, 2

in, 876

united, 15

states, 11

and, 1607

most, 49

other, 57

parts, 4

……………………………………..