🐍

Revise Python By Harshit Dabas (Beginner)

Section 1 : Variables

Band Name Generator

```
city=input("Enter your fav City: \n")
animal=input("Enter your pet's name: \n")
print(city+" "+animal)
```

Section 2 : Data Types, String Manipulation

Data Types

```
# Subscripting
print("Hello"[0])

# String
print("123" + "345")

# Integer = Whole number
print(123 + 345)

# Large Integers
print(123_456_789)

# Float = Floating Point Number
print(3.14159)

# Boolean
```

```
print(True)
print(False)
```

Type Error, Checking and Conversion

```
# TypeError
# len(123)

# No TypeError
len("Hello")

# Type Checking
print(type("abc"))
print(type(123))
print(type(3.14))
print(type(True))

# Type Conversion
str()
int()
float()
bool()

name_of_the_user = input("Enter your name")
length_of_name = len(name_of_the_user)

print(type("Number of letters in your name: "))  # str
print(type(length_of_name))  # int

print("Number of letters in your name: " + str(length_of_name))
```

Mathematical Operations

```
print("My age: " + str(12))
print(123 + 456)
print(7 - 3)
print(3 * 2)
print(5 / 3)
print(5 // 3)
print(2 ** 3)

# PEMDASLR Order
# ()
# **
# * OR /
# + OR -

# Outputs 7
print(3 ∗ 3 + 3 / 3 - 3)

# Outputs 3
print(3 ∗ 3 + 3 / 3 - 3)
```

Number Manipulation

```
bmi = 84 / 1.65 ** 2

# Original Float with decimal places
print(bmi)

# Flooring the number by converting it into int
print(int(bmi))

# Rounding the number into a whole number
print(round(bmi))

# Rounding only to 2 decimal places
print(round(bmi, 2))


## Accumulate
score = 0

# User scores a point
score += 1
print(score)
```

```
#Also
score -= 1
score *= 2
score /= 2

score = 0
height = 1.8
is_winning = True

print(f"Your score is = {score}, your height is {height}. You are winning is {is_w
```

Tip Calculator

```
print("Welcome to the tip calculator!")
bill = float(input("What was the total bill? $"))
tip = int(input("What percentage tip would you like to give? 10 12 15 "))
people = int(input("How many people to split the bill? "))
tip_as_percent = tip / 100
total_tip_amount = bill * tip_as_percent
total_bill = bill + total_tip_amount
bill_per_person = total_bill / people
final_amount = round(bill_per_person, 2)
print(f"Each person should pay: ${final_amount}")
```

Section 3: Control Flow, Logical Operators

If Else

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))

if height >= 120:
    print("You can ride the rollercoaster")
```

```
else:
    print("Sorry you have to grow taller before you can ride.")
```

Modulo

```
number_to_check = int(input("What is the number you want to check? "))

if number_to_check % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Nesting and Elif

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))

if height >= 120:
    print("You can ride the rollercoaster")
    age = int(input("What is your age? "))
    if age <= 12:
        print("Please pay $5.")
    elif age <= 18:
        print("Please pay $7.")
    else:
        print("Please pay $12.")
else:
    print("Sorry you have to grow taller before you can ride.")
```

Multiple Ifs

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))
bill = 0

if height >= 120:
    print("You can ride the rollercoaster!")
    age = int(input("What is your age? "))
    if age < 12:
        bill = 5
        print("Child tickets are $5.")
    elif age <= 18:
        bill = 7
        print("Youth tickets are $7.")
    else:
        bill = 12
        print("Adult tickets are $12.")

    wants_photo = input("Do you want a photo taken? Y or N. ")
    if wants_photo == "Y":
        bill += 3

    print(f"Your final bill is ${bill}")

else:
    print("Sorry, you have to grow taller before you can ride.")
```

Python Pizza

```
print("Welcome to Python Pizza Deliveries!")
size = input("What size pizza do you want? S, M or L: ")
pepperoni = input("Do you want pepperoni on your pizza? Y or N: ")
extra_cheese = input("Do you want extra cheese? Y or N: ")

# todo: work out how much they need to pay based on their size choice.

bill = 0

if size == "S":
    bill += 15
elif size == "M":
    bill += 20
elif size == "L":
    bill += 25
else:
    print("You have chosen an invalid size.")
```

```
# todo: work out how much to add to their bill based on their pepperoni choice
if pepperoni == "Y":
    if size == "S":
        bill += 2
    else:
        bill += 3

# todo: work out their final amount based on whether if they want extra chees
if extra_cheese == "Y":
    bill += 1

print(f"Your final bill is: ${bill}.")
```

Logical Operators

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))
bill = 0

if height >= 120:
    print("You can ride the rollercoaster!")
    age = int(input("What is your age? "))
    if age < 12:
        bill = 5
        print("Child tickets are $5.")
    elif age <= 18:
        bill = 7
        print("Youth tickets are $7.")
    elif age >= 45 and age <= 55:
        # Or
        # 45 <= age <= 55
        print("Everything is going to be ok. Have a free ride on us!")
    else:
        bill = 12
        print("Adult tickets are $12.")

    wants_photo = input("Do you want a photo taken? Y or N. ")
    if wants_photo == "Y":
        bill += 3

    print(f"Your final bill is ${bill}")

else:
    print("Sorry, you have to grow taller before you can ride.")
```

Treasure Island

```
print(r'''
*******************************************************************************
          |                   |                  |                     |
 _________|________________.=""_;=.______________|_____________________|_______
|                   |  ,-"_,=""     `"=.|                  |
|___________________|__"=._o`"-._        `"=.______________|___________________
          |                `"=._o`"=._      _`"=._                     |
 _________|_____________________:=._o "=._."_.-="'"=.__________________|_______
|                   |    __.--" , ; `"=._o." ,-"""-._ ".   |
|___________________|_._"  ,. .` ` `` ,  `"-._"-._   ". '__|___________________
          |           |o`"=._` , "` `; .". ,  "-._"-._; ;              |
 _________|___________| ;`-.o`"=._; ." ` '`."\ ` . "-._ /_______________|_______
|                   | |o ;    `"-.o`"=._``  '` " ,__.--o;   |
|___________________|_| ;     (#) `-.o `"=.`_.--"_o.-; ;___|___________________
____/______/______/___|o;._    "      `".o|o_.--"    ;o;____/______/______/____
/______/______/______/_"=._o--._        ; | ;        ; ;/______/______/______/_
____/______/______/______/__"=._o--._   ;o|o;     _._;o;____/______/______/____
/______/______/______/______/____"=._o._; | ;_.--"o.--"_/______/______/______/_
____/______/______/______/______/_____"=.o|o_.--""___/______/______/______/____
/______/______/______/______/______/______/______/______/______/______/_____ /
*******************************************************************************
''')
print("Welcome to Treasure Island.")
print("Your mission is to find the treasure.")
choice1 = input('You\'re at a crossroad, where do you want to go? '
                'Type "left" or "right".\n').lower()

if choice1 == "left":
    choice2 = input('You\'ve come to a lake. '
                    'There is an island in the middle of the lake. '
                    'Type "wait" to wait for a boat. '
                    'Type "swim" to swim across.\n').lower()
    if choice2 == "wait":
        choice3 = input("You arrive at the island unharmed. "
                        "There is house with 3 doors. One red, "
                        "one yellow and one blue. "
                        "Which colour do you choose?\n").lower()
        if choice3 == "red":
            print("It's a room full of fire. Game Over")
        elif choice3 == "yellow":
            print("You found the treasure. You Win!")
        elif choice3 == "blue":
            print("You enter a room of beasts. Game Over.")
        else:
            print("You chose a door that doesn't exist. Game Over.")
    else:
        print("You got attacked by an angry trout. Game Over.")

else:
    print("You fell in to a hole. Game Over.")
```

Section 4: Randomization, Lists

Random Module

```
import random

random_integer = random.randint(1, 10)
print(random_integer)

random_number_0_to_1 = random.random() * 10
print(random_number_0_to_1)

random_float = random.uniform(1, 10)
print(random_float)
```

```
random_heads_or_tails = random.randint(0, 1)
if random_heads_or_tails == 0:
    print("Heads")
else:
    print("Tails")
```

Lists

```
states_of_america = ["Delaware", "Pennsylvania", "New Jersey", "Georgia", "C

states_of_america[1] = "Pencilvania"

states_of_america.append("Angelaland")

states_of_america.extend(["Angelaland", "Jack Bauer Land"])

print(states_of_america)
```

Banker Roulette

```
import random
friends = ["Alice", "Bob", "Charlie", "David", "Emanuel"]

# 1st Option
print(random.choice(friends))

# 2nd Option
random_index = random.randint(0, 4)
print(friends[random_index])
```

IndexError

```
states_of_america = ["Delaware", "Pennsylvania", "New Jersey", "Georgia", "C
                     "South Carolina", "New Hampshire", "Virginia", "New York", "Nort
                     "Vermont", "Kentucky", "Tennessee", "Ohio", "Louisiana", "Indian
                     "Alabama", "Maine", "Missouri", "Arkansas", "Michigan", "Florida"
                     "California", "Minnesota", "Oregon", "Kansas", "West Virginia", "N
```

```
                    "North Dakota", "South Dakota", "Montana", "Washington", "Idaho
                    "New Mexico", "Arizona", "Alaska", "Hawaii"]

print(states_of_america[49])  # No error
print(states_of_america[50])  # IndexError

# Using len() to find the number of items in a List
num_states = len(states_of_america)
print(states_of_america[num_states - 1])



# dirty_dozen = ["Strawberries", "Spinach", "Kale", "Nectarines", "Apples", "Gr
# "Tomatoes", "Celery", "Potatoes"]

fruits = ["Strawberries", "Nectarines", "Apples", "Grapes", "Peaches", "Cherrie
vegetables = ["Spinach", "Kale", "Tomatoes", "Celery", "Potatoes"]

dirty_dozen = [fruits, vegetables]
print(dirty_dozen)
```

Rock Paper Scissors

```
import random

rock = '''
    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)
'''

paper = '''
    _______
---'   ____)____
          ______)
          _______)
          _______)
---.__________)
'''

scissors = '''
    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)
'''

game_images = [rock, paper, scissors]

user_choice = int(input("What do you choose? Type 0 for Rock, 1 for Paper or
# Note: it's worth checking if the user has made a valid choice before the next
# If the user typed somthing other than 0, 1 or 2 the next line will give you an e
# You could for example write:
if user_choice >= 0 and user_choice <= 2:
    print(game_images[user_choice])

computer_choice = random.randint(0, 2)
print("Computer chose:")
print(game_images[computer_choice])

if user_choice >= 3 or user_choice < 0:
    print("You typed an invalid number. You lose!")
elif user_choice == 0 and computer_choice == 2:
    print("You win!")
elif computer_choice == 0 and user_choice == 2:
    print("You lose!")
elif computer_choice > user_choice:
    print("You lose!")
elif user_choice > computer_choice:
    print("You win!")
elif computer_choice == user_choice:
    print("It's a draw!")
```

Section 5: Loops

For Loops

```
fruits = ["Apple", "Peach", "Pear"]
for fruit in fruits:
    print(fruit)
    print(fruit + " pie")

print(fruits)
```

Highest Score

```
student_scores = [150, 142, 185, 120, 171, 184, 149, 24, 59, 68, 199, 78, 65, 89,

max_score = 0
for score in student_scores:
    if score > max_score:
        max_score = score

print(max_score)
```

For Loops with Range

```
print(range(1, 10))  # Doesn't do anything

for number in range(1, 10):  # Prints 1 to 9
    print(number)

for number in range(1, 11):  # Prints 1 to 10
    print(number)



# Gauss challenge
total = 0
for number in range(1, 101):
    total += number
print(total)
```

Password Generator

```
import random
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', '
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

print("Welcome to the PyPassword Generator!")
nr_letters = int(input("How many letters would you like in your password?\n"))
nr_symbols = int(input(f"How many symbols would you like?\n"))
nr_numbers = int(input(f"How many numbers would you like?\n"))

# Easy Level
# password = ""
# for char in range(0, nr_letters):
#     password += random.choice(letters)
#
# for char in range(0, nr_symbols):
#     password += random.choice(symbols)
#
# for char in range(0, nr_numbers):
#     password += random.choice(numbers)
#
# print(password)

# Hard level
password_list = []
for char in range(0, nr_letters):
    password_list.append(random.choice(letters))

for char in range(0, nr_symbols):
    password_list.append(random.choice(symbols))

for char in range(0, nr_numbers):
    password_list.append(random.choice(numbers))

print(password_list)
random.shuffle(password_list)
```

```
print(password_list)

password = ""
for char in password_list:
    password += char

print(f"Your password is: {password}")
```

Section 6: Functions

Functions

```
def my_function():
    print("Hello")
    print("Bye")



my_function()



def turn_right():
    turn_left()
    turn_left()
    turn_left()



while not at_goal():
    if right_is_clear():
        turn_right()
        move()
    elif front_is_clear():
        move()
    elif wall_in_front() and wall_on_right():
        turn_left()
```

Section 7: Hangman

Word Guesser

```
# TODO-1 - Randomly choose a word from the word_list and assign it to a vari

# TODO-2 - Ask the user to guess a letter and assign their answer to a variabl

# TODO-3 - Check if the letter the user guessed (guess) is one of the letters i
#  is, "Wrong" if it's not.

import random
word_list = ["aardvark", "baboon", "camel"]
print("You Have 5 lives")
lives = 5
chosen_word = random.choice(word_list).lower()
original_word = chosen_word

print(chosen_word)
placeholder=""
correct_letter = []
for position in range(len(chosen_word)):
    placeholder += "_"
print("Word to guess: " + placeholder)

while lives > 0:
    check = input("Enter a letter to check: ").lower()
    blanks = ""
    for i in chosen_word:
        if i == check:
            blanks += i
            correct_letter.append(i)
        elif i in correct_letter:
            blanks += i
        else:
            blanks += "_"
    print("Word to guess: " + blanks)
    if check not in chosen_word:
        lives-=1
        print(f"Wrong Choice!! \nYou have {lives} lives left.\nThe word still is:  {bla

    if "_"  not in blanks:
        print("You Won!!!")

if lives == 0:
    print(f"You Lost!!\nThe Word was {original_word}")
```

Section 8: Function Parameters, Caesar Cipher

Functions with inputs

```
# Simple Function that packages code into a named block
def greet():
    print("Hello Angela")
    print("How do you do Jack Bauer?")
    print("Isn't the weather nice?")



greet()



# Function that allows for inputs
def greet_with_name(name):
    print(f"Hello {name}")
    print(f"How do you do {name}?")



greet_with_name("Billie")
```

Positional vs keyword args

```
# Functions with input

# def greet_with_name(name):
#     print(f"Hello {name}")
#     print(f"How do you do {name}?")
#
#
# greet_with_name("Jack Bauer")

# Functions with more than 1 input
def greet_with(name, location):
    print(f"Hello {name}")
    print(f"What is it like in {location}")

# Positional arguments
# greet_with("Jack Bauer", "Nowhere")
# greet_with("Nowhere", "Jack Bauer")


# Keyword arguments
greet_with(location="London", name="Angela")
```

Caesar Cipher

```
import art

print(art.logo)

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't
            'v', 'w', 'x', 'y', 'z']


def caesar(original_text, shift_amount, encode_or_decode):
    output_text = ""
    if encode_or_decode == "decode":
        shift_amount *= -1
    for letter in original_text:

        if letter not in alphabet:
            output_text += letter
        else:
            shifted_position = alphabet.index(letter) + shift_amount
            shifted_position %= len(alphabet)
            output_text += alphabet[shifted_position]
    print(f"Here is the {encode_or_decode}d result: {output_text}")


should_continue = True

while should_continue:

    direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n").lo
    text = input("Type your message:\n").lower()
    shift = int(input("Type the shift number:\n"))

    caesar(original_text=text, shift_amount=shift, encode_or_decode=direction)

    restart = input("Type 'yes' if you want to go again. Otherwise, type 'no'.\n")
    if restart == "no":
        should_continue = False
        print("Goodbye")
```

Section 9: Dictionaries

Dictionaries

```
# Creating a dictionary
programming_dictionary = {
    "Bug": "An error in a program that prevents the program from running as ex
    "Function": "A piece of code that you can easily call over and over again.",
}

# Retrieving a value from a dictionary
print(programming_dictionary["Function"])

# Adding more items to a dictionary
programming_dictionary["Loop"] = "The action of doing something over and o

# Creating an empty dictionary
empty_dictionary = {}

# Wipe an existing dictionary
# programming_dictionary = {}
# print(programming_dictionary)

# Edit an item in a dictionary
programming_dictionary["Bug"] = "A moth in your computer."
# print(programming_dictionary)

# Loop through a dictionary
for key in programming_dictionary:
    print(key)
    print(programming_dictionary[key])
```

Nested Lists and Dict

```
capitals = {
    "France": "Paris",
    "Germany": "Berlin",
}

# Nested List in Dictionary

# travel_log = {
#     "France": ["Paris", "Lille", "Dijon"],
#     "Germany": ["Stuttgart", "Berlin"],
# }
# print Lille
# print(travel_log["France"][1])

nested_list = ["A", "B", ["C", "D"]]
# print(nested_list[2][1])

# Nested dictionary in a dictionary
travel_log = {
  "France": {
    "cities_visited": ["Paris", "Lille", "Dijon"],
    "total_visits": 12
   },
  "Germany": {
    "cities_visited": ["Berlin", "Hamburg", "Stuttgart"],
    "total_visits": 5
   },
}

print(travel_log["Germany"]["cities_visited"][2])
```

Blind Auction

```
from art import logo
print(logo)



def find_highest_bidder(bidding_record):
    highest_bid = 0
    winner = ""
    for bidder in bidding_record:
        bid_amount = bidding_record[bidder]
        if bid_amount > highest_bid:
            highest_bid = bid_amount
            winner = bidder
    print(f"The winner is {winner} with a bid of ${highest_bid}")
```

```
bids = {}
continue_bidding = True
while continue_bidding:
    name = input("What is your name?: ")
    price = int(input("What is your bid?: $"))
    bids[name] = price
    should_continue = input("Are there any other bidders? Type 'yes or 'no'.\n")
    if should_continue == "no":
        continue_bidding = False
        find_highest_bidder(bids)
    elif should_continue == "yes":
        print("\n" * 20)
```

Section 10: Functions with Outputs

Functions with Outputs

```
def format_name(f_name, l_name):
    formated_f_name = f_name.title()
    formated_l_name = l_name.title()
    return f"{formated_f_name} {formated_l_name}"


print(format_name("AnGeLa", "YU"))


def function_1(text):
    return text + text


def function_2(text):
    return text.title()


output = function_2(function_1("hello"))
```

```
print(output)
```

Multiple Return Values

```
def format_name(f_name, l_name):
    if f_name == "" or l_name == "":
        return "You did not provide valid inputs"
    formated_f_name = f_name.title()
    formated_l_name = l_name.title()
    return f"Result: {formated_f_name} {formated_l_name}"


print(format_name(input("What is your first name?"), input("What is your last n
```

Doc Strings

```
def format_name(f_name, l_name):
    """Take a first and last name and format it to return the
    title case version of the name."""
    formated_f_name = f_name.title()
    formated_l_name = l_name.title()
    return f"{formated_f_name} {formated_l_name}"


formatted_name = format_name("AnGeLa", "YU")

length = len(formatted_name)
```

Calculator

```
import art


def add(n1, n2):
    return n1 + n2


def subtract(n1, n2):
    return n1 - n2


def multiply(n1, n2):
    return n1 * n2


def divide(n1, n2):
    return n1 / n2


operations = {
    "+": add,
    "-": subtract,
    "*": multiply,
    "/": divide,
}

# print(operations["*"](4, 8))


def calculator():
    print(art.logo)
    should_accumulate = True
    num1 = float(input("What is the first number?: "))

    while should_accumulate:
        for symbol in operations:
            print(symbol)
        operation_symbol = input("Pick an operation: ")
        num2 = float(input("What is the next number?: "))
        answer = operations[operation_symbol](num1, num2)
        print(f"{num1} {operation_symbol} {num2} = {answer}")

        choice = input(f"Type 'y' to continue calculating with {answer}, or type 'r

        if choice == "y":
            num1 = answer
        else:
            should_accumulate = False
            print("\n" * 20)
            calculator()


calculator()
```

Section 11: Blackjack Game

```
import random
from art import logo



def deal_card():
    """Returns a random card from the deck"""
    cards = [11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10]
    card = random.choice(cards)
    return card



def calculate_score(cards):
    """Take a list of cards and return the score calculated from the cards"""
    if sum(cards) == 21 and len(cards) == 2:
        return 0

    if 11 in cards and sum(cards) > 21:
        cards.remove(11)
        cards.append(1)

    return sum(cards)


def compare(u_score, c_score):
    """Compares the user score u_score against the computer score c_score."""
    if u_score == c_score:
        return "Draw 🙃"
    elif c_score == 0:
        return "Lose, opponent has Blackjack 😱"
    elif u_score == 0:
        return "Win with a Blackjack 😎"
    elif u_score > 21:
        return "You went over. You lose 😭"
    elif c_score > 21:
        return "Opponent went over. You win 😁"
    elif u_score > c_score:
        return "You win 😃"
    else:
        return "You lose 😤"


def play_game():
    print(logo)
    user_cards = []
    computer_cards = []
    computer_score = -1
    user_score = -1
    is_game_over = False

    for _ in range(2):
        user_cards.append(deal_card())
        computer_cards.append(deal_card())

    while not is_game_over:
        user_score = calculate_score(user_cards)
        computer_score = calculate_score(computer_cards)
        print(f"Your cards: {user_cards}, current score: {user_score}")
        print(f"Computer's first card: {computer_cards[0]}")

        if user_score == 0 or computer_score == 0 or user_score > 21:
            is_game_over = True
        else:
            user_should_deal = input("Type 'y' to get another card, type 'n' to pass
            if user_should_deal == "y":
                user_cards.append(deal_card())
            else:
                is_game_over = True

    while computer_score != 0 and computer_score < 17:
        computer_cards.append(deal_card())
        computer_score = calculate_score(computer_cards)

    print(f"Your final hand: {user_cards}, final score: {user_score}")
    print(f"Computer's final hand: {computer_cards}, final score: {computer_sc
    print(compare(user_score, computer_score))


while input("Do you want to play a game of Blackjack? Type 'y' or 'n': ") == "y
    print("\n" * 20)
    play_game()
```

Section 12:Scope and Number Guessing Game

Namespaces and Scope

```
enemies = 1



def increase_enemies():
    enemies = 2
    print(f"enemies inside function: {enemies}")
```

```
increase_enemies()
print(f"enemies outside function: {enemies}")


# Local Scope
def drink_potion():
    potion_strength = 2
    print(potion_strength)


drink_potion()
# Can't access this potion_strength outside of its scope
# print(potion_strength)

# Global Scope
player_health = 10


def game():
    def drink_potion():
        potion_strength = 2
        print(player_health)

    drink_potion()


print(player_health)
```

Block Scopes

```
game_level = 10
enemies = ["Skeleton", "Zombie", "Alien"]


def create_enemy():
    new_enemy = ""
    if game_level < 5:
        new_enemy = enemies[0]

    print(new_enemy)
```

Global Variables

```
# Modifying Global Scope

enemies = 1

# def increase_enemies():
#     global enemies
#     enemies += 1
#     print(f"enemies inside function: {enemies}")


def increase_enemies(enemy):
    print(f"enemies inside function: {enemy}")
    return enemy + 1


enemies = increase_enemies(enemies)
print(f"enemies outside function: {enemies}")
```

Global Constraints

```
from random import randint
from art import logo


EASY_LEVEL_TURNS = 10
HARD_LEVEL_TURNS = 5
```

```
# Function to check users' guess against actual answer
def check_answer(user_guess, actual_answer, turns):
    """Checks answer against guess, returns the number of turns remaining."""
    if user_guess > actual_answer:
        print("Too high.")
        return turns - 1
    elif user_guess < actual_answer:
        print("Too low.")
        return turns - 1
    else:
        print(f"You got it! The answer was {actual_answer}")


# Function to set difficulty
def set_difficulty():
    level = input("Choose a difficulty. Type 'easy' or 'hard': ")
    if level == "easy":
        return EASY_LEVEL_TURNS
    else:
        return HARD_LEVEL_TURNS


def game():
    print(logo)
    # Choosing a random number between 1 and 100.
    print("Welcome to the Number Guessing Game!")
    print("I'm thinking of a number between 1 and 100.")
    answer = randint(1, 100)
    print(f"Pssst, the correct answer is {answer}")

    turns = set_difficulty()

    # Repeat the guessing functionality if they get it wrong.
    guess = 0
    while guess != answer:
        print(f"You have {turns} attempts remaining to guess the number.")
        # Let the user guess a number
        guess = int(input("Make a guess: "))
        # Track the number of turns and reduce by 1 if they get it wrong
        turns = check_answer(guess, answer, turns)
        if turns == 0:
            print("You've run out of guesses, you lose.")
            return
        elif guess != answer:
            print("Guess again.")

game()
```

Number Guessing Game

```
from random import randint
from art import logo


EASY_LEVEL_TURNS = 10
HARD_LEVEL_TURNS = 5


# Function to check users' guess against actual answer
def check_answer(user_guess, actual_answer, turns):
    """Checks answer against guess, returns the number of turns remaining."""
    if user_guess > actual_answer:
        print("Too high.")
        return turns - 1
    elif user_guess < actual_answer:
        print("Too low.")
        return turns - 1
    else:
        print(f"You got it! The answer was {actual_answer}")


# Function to set difficulty
def set_difficulty():
    level = input("Choose a difficulty. Type 'easy' or 'hard': ")
    if level == "easy":
        return EASY_LEVEL_TURNS
    else:
        return HARD_LEVEL_TURNS


def game():
    print(logo)
    # Choosing a random number between 1 and 100.
    print("Welcome to the Number Guessing Game!")
    print("I'm thinking of a number between 1 and 100.")
    answer = randint(1, 100)
    print(f"Pssst, the correct answer is {answer}")

    turns = set_difficulty()

    # Repeat the guessing functionality if they get it wrong.
    guess = 0
    while guess != answer:
        print(f"You have {turns} attempts remaining to guess the number.")
        # Let the user guess a number
        guess = int(input("Make a guess: "))
        # Track the number of turns and reduce by 1 if they get it wrong
        turns = check_answer(guess, answer, turns)
        if turns == 0:
            print("You've run out of guesses, you lose.")
            return
        elif guess != answer:
            print("Guess again.")

game()
```

Section 13 : Debugger

Theory Only.

Section 14: Higher Lower Game

```
# Display art
from art import logo, vs
from game_data import data
import random


def format_data(account):
    """Takes the account data and returns the printable format."""
    account_name = account["name"]
    account_descr = account["description"]
    account_country = account["country"]
    return f"{account_name}, a {account_descr}, from {account_country}"


def check_answer(user_guess, a_followers, b_followers):
    """Take a user's guess and the follower counts and returns if they got it righ
    if a_followers > b_followers:
        return user_guess == "a"
    else:
        return user_guess == "b"


print(logo)
score = 0
game_should_continue = True
# Generate a random account from the game data
account_b = random.choice(data)

# Make the game repeatable.
while game_should_continue:

    # Making account at position B become the next account at position A.
    account_a = account_b
    account_b = random.choice(data)

    if account_a == account_b:
```

```
    account_b = random.choice(data)

print(f"Compare A: {format_data(account_a)}.")
print(vs)
print(f"Against B: {format_data(account_b)}.")

# Ask user for a guess.
guess = input("Who has more followers? Type 'A' or 'B': ").lower()

# Clear the screen
print("\n" * 20)
print(logo)

# - Get follower count of each account
a_follower_count = account_a["follower_count"]
b_follower_count = account_b["follower_count"]

# Check if user is correct.
is_correct = check_answer(guess, a_follower_count, b_follower_count)

# Give user feedback on their guess.
# score keeping.
if is_correct:
    score += 1
    print(f"You're right! Current score {score}")
else:
    print(f"Sorry, that's wrong. Final score: {score}.")
    game_should_continue = False
```

Notion Link

https://www.notion.so/Revise-Python-By-Harshit-Dabas-Beginner-18b44adb142b80778e70c09cebc00741?pvs=4