What’s the alternative?

Code snippets .

Displays an instruction and prompts for input in one line.

| 1 | name = input("Enter your name:") |
| --- | --- |

Increments the score value by one.

| 1 | score += 1 |
| --- | --- |

Displays a variable within a print, variables are separated by commas.

| 1 | print("Name:", name, "Score:", score) |
| --- | --- |

Concatenates a string within a print statement. Casting is required on all non-string variables.

| 1 | print("Name: " + name + "Score: " + str(score)) |
| --- | --- |

Uses a call without the use of an additional variable.

| 1  2  3  4 | numbers = [3, 4, 5, 2, 3, 4, 5, 5]  for x in range(3, len(numbers)):  print(numbers[x]) |
| --- | --- |

# Challenge 1 . Adapt this program

A ‘guess the word’ game has been created. The game gives the user five attempts to guess the word.

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | x = "sandwich"  y = ""  z = x != y  a = 0  while z and a < 5:  print("Guess the word: ")  y = input()  if y == x:  a = a + 1  b = "CORRECT!"  z = False  else:  a = a + 1  b = "INCORRECT!"  print(f"You were {b}. Guesses: {a}") |
| --- | --- |

**Instructions**

* Copy this code into your development environment.
* Execute the code to make sure that you understand how it works.
* The code doesn’t use meaningful identifiers. Change the variable names so that they use meaningful identifiers.
* Use the code snippets above to provide an alternative solution to the same problem.
* Add informative commenting to the program.

**Paste your completed code below:**

**Note for assessor: this is just one example, there will be many variations from learners.**

|  | # variable declaration  word = "sandwich"  guess = ""  not\_guessed = word != guess  guesses = 0  # continue to guess whilst incorrect and number of guesses less than 5  while not\_guessed and guesses < 5:  guess = input("Guess the word: ")  if guess == word:  guesses += 1  result = "CORRECT!"  not\_guessed = False  else:  guesses += 1  result = "INCORRECT!"  # display the final result to the user  print("You were", result, "Guesses:", guesses) |
| --- | --- |

# Challenge 2 . Adapt this program

A ‘Simon says’ program has been created. The player types in as many Simon says instructions as they wish. The program will then iterate through the list, either saying “Simon says” or leaving that part blank.

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | from random import choice  from time import sleep  simonSays = []  AddingInstructions = True  while AddingInstructions:  print("Enter a Simon says instruction")  INSTRUCTION = input()  simonSays.append(INSTRUCTION)  print("Would you like to add another? Y/N")  ANSWER = input().upper()  if ANSWER == "N":  AddingInstructions = False  Intros = ["Simon says...", ""]  simonSaysLength = len(simonSays)  for x in range(simonSaysLength):  INSTRUCTION = simonSays[x]  Intro = choice(Intros)  print(f"{Intro}{INSTRUCTION}")  sleep(3) |
| --- | --- |

**Instructions**

* Copy the code into your development environment.
* Execute the code to make sure that you understand how it works.
* Fix the variable names. They have been written using inconsistent naming conventions. They should follow the Python naming conventions.
* Use the code snippets above to form an alternative approach to the solution.
* Add informative commenting to the program.

**Paste your completed code below:**

**Note for assessor: this is just one example, there will be many variations from learners.**

|  | # library imports  from random import choice  from time import sleep  simon\_says = []  adding\_instructions = True  # continue to ask for Simon says instructions until user is finished  while adding\_instructions:  instruction = input("Enter a Simon says instruction: ")  simon\_says.append(instruction)  answer = input("Would you like to add another? Y/N: ").upper()  if answer == "N":  adding\_instructions = False  intros = ["Simon says...", ""]  # play the game  for x in range(len(simon\_says)):  instruction = simon\_says[x]  print(choice(intros), instruction)  sleep(3) |
| --- | --- |

# Explorer task . Adapt any program

Look back over any program that you have made during this unit and adapt it using the methods covered in this lesson.