|  |  |  |
| --- | --- | --- |
| Document name | : | HR.Skills.A2W5LA |
| Document description | : | This document contains the Learning Activities: “Code Analysis”. |
| Document Version | : | V 1.0 |
| Written by | : | Danny de Snoo |
| Date(s) written | : | 07/06/2024 |

Learning Activities description:

Code Analysis

1. Analyze the two given codes below without executing them. What will be the result of the programs?

|  |
| --- |
| a\_tuple = ('Never', 'gonna', 'give', 'you', 'up')  counter = 0  for x in a\_tuple:  if x[0] == 'g':  counter = counter + 1  else:  counter = counter + 2  print(counter) |

Analysis:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| This piece of code preforms a for-loop over the items in a tuple called a\_tuple The code has a counter which will be printed at the end of the line.  In the for-loop it loops over each item in the tuple and checks if the first character in the string is a lowercase g if it is the it just adds one to the counter else it adds a two to the counter.  if we use the logic of this loop and the items in the tuple we should get a result of 8   |  |  |  | | --- | --- | --- | | String |  | Point amount | | Never | : | 2 | | gonna | : | 1 | | give | : | 1 | | you | : | 2 | | up | : | 2 | | Total |  | 8 | |

|  |
| --- |
| def do\_something(x):  rtuple = x,  for i in range(2,11):  rtuple = rtuple + ((x\*i),)  return rtuple  print(do\_something(6)) |

Analysis:

|  |
| --- |
| This function creates a tuple with 10 integers in it based on the inputted number  first we create a tuple in which we add our first number. After this we enter a for-loop in which we loop by using the range function with its parameters as 2 and 11. Which means we will be going through the loops about 9 before it ends.  per time we loop through the for-loop we perform a calculation which does the x (input number) times the current iteration of our for-loop.  So if we would run this function with the current input number (6) Then we would get something like this as its result: (6, 12, 18, 24, 30, 36, 42, 48, 54, 60) |

|  |
| --- |
| def process\_strings(strings):  processed\_strings = []  for string in strings:  processed\_string = ""  for char in reversed(string):  processed\_string += char  processed\_strings.append(processed\_string)  return processed\_strings  def main():  names = ["Alice", "Bob", "Charlie", "Dave"]  processed\_names = process\_strings(names)  for name in processed\_names:  print(name)  main() |

Analysis:

|  |
| --- |
| This script contains 2 functions:  process\_strings with a required input called strings (the input needs to be a list object) and main which doesn’t require an input.  The main function is called when the script is run.  When main is run a list of names will be processed by using the process\_strings function. This function first makes a new list object called “processed\_strings” after that it iterates through the list object that has been inputted. In the iteration it creates an new string variable called “processed\_string”. |