## **Knowledge check**

The list below outlines the programming knowledge you will need to be able to complete the programming assessment task. Use these examples to check you are confident with each of these. If you're not, go back and relearn the relevant skills.

| **Use code to input and output data**  # using input() to take user input  num = input('Enter a number: ')  # using print() to output  print('You Entered: ', num)  Use **variables** to store data of different data types including strings, numbers (integers and floats), and booleans  # an integer variable  number = 10  # a string variable  name = 'Matua Matt'  # a boolean variable  raining = False  Use **variables** to demonstrate how a **variable value can change** during program execution  name = 'Matt'  age = 52  # assigning a new value to name  name = 'Jake'  # changing a variable using operators  age = age + 1 #or age += 1  Use **appropriate** variable names  # variables should have sensible names and use lower case, numbers, and underscore only  animal = 'Snake'  animal2 = 'Rat'  zoo\_keeper = 'Steve'  temp = 98  Use **basic maths operations** on variables like add, subtract, multiply, divide and modulus  a = 7  b = 2  # addition  print('Addition: ', a + b) #equals 9  # subtraction  print('Subtraction: ', a - b) #equals 5  # multiplication  print('Multiplication: ', a \* b) # = 14  # division  print('Division: ', a / b) #equals 3.5  # floor division  print('Floor Division: ', a // b) # = 3  # modulo  print('Modulo: ', a % b) #equals 1  # a to the power b  print('Power: ', a \*\* b) #equals 49 | Use **comparison operations** which could include less than, greater than, less than or equal to, greater than or equal to and equal to.  a = 5  b = 2  c = 5  print (a > b) # returns TRUE  print (a < c) # returns FALSE  print (a <= c) # returns TRUE  Use **selection statements** like ‘if’ and ‘if-else’ that allows code to be optionally executed based on certain conditions  number = 10  # if statement  if number > 0:  # remember the indent  print('Number is positive')  print('This statement always executes')  # if...else statement  number = 10  if number > 0:  print('Positive number')  else:  print('Negative number')  print('This statement always executes')  Use **loops** (**iterative code)** that could include ‘repeat’ loops, ‘while’ loops and ‘for’ loops to repeat blocks of code based on conditions.  # while loop repeats until a condition is met  number = 1  while number <= 5:  print(number)  number = number + 1 #or number += 1  print('This prints when the loop ends')  # for loop cycles through a list  languages = ['C++', 'Python', 'Java']  # access elements of the list one by one  for i in languages:  print(i)  # for can cycle through a range too  number = 8  for i in range(number):  print(i)  # for can iterate over each character too  language = ('Python')  for x in language: # x is a temp variable  print(x) | Use **nested code** within loops or selections  # nested code is a loop within a loop  # outer loop (using a for loop)  for i in range(1, 11):  # nested loop to iterate from 1 to 10  for j in range(1, 11):  # print multiplication  print(i \* j, end=' ')  print()  # outer loop (using a nested if-else loop)  num = int(input('Number?'))  if num% 2 == 0:  if num% 3 == 0:  print('Divisible by 3 and 2')  else:  print('divisible by 2 but not divisible by 3')  Else:  if num% 3 == 0:  print('divisible by 3 but not divisible by 2')  else:  print('not Divisible by 2 and not divisible by 3')  Use **collections** such as **lists** and **dictionaries** to store, access and edit values.  # a list of three elements  languages = ['Python', 'Swift', 'C++']  # access the first and third elements  print(languages[0]) # output Python  print(languages[2]) # output C++  # access information using a dictionary  country\_capitals = {’Germany’: ‘Berlin’ , ‘Canada’: ‘Ottawa’ , ‘England: ‘London’}  print(country\_capitals[‘Germany’])  # Output: Berlin  print(country\_capitals[‘England’])  # Output: London  # Python has many useful list methods too.  # e.g. add(append) items to a list  fruits = ['apple' , 'banana' , 'orange']  fruits.append('cherry')  print('Updated List:' , fruits)  Use **comments** to explain sections of your code  # you should use a comment before every section of code to explain what it does  Use logical code **sequences** for programs and sub-sections  # remember code runs from top to bottom. If it encounters a loop subsection it completes that before moving on  Use accurate and consistent **syntax and structure** in your code  # use 4 spaces per indentation group, use spaces not tab  if num%2 == 0:  if num%3 == 0:  print('Divisible by 3 and 2')  # limit lines to a max of 79 characters  # use blank lines sparingly, just to indicate logical sections  # python is case-sensitive. mynumber is different to Mynumber  # use spaces around operands and dividers |
| --- | --- | --- |