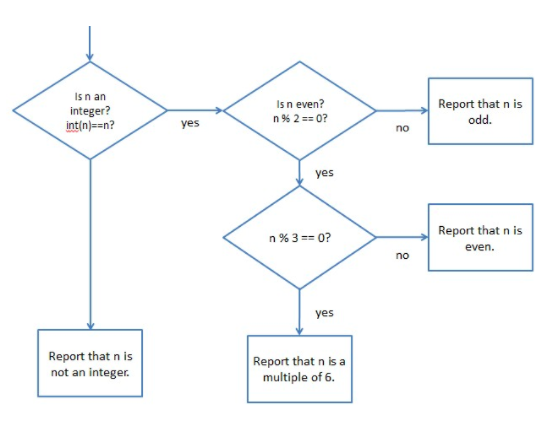
**APCSP Activity 1.3.4: Nested Branching and Input**

Learning Target: Define the problem and analyze research to create a solution to a problem.

**Step 3**

Complete the food\_id\_test() function that calls food\_id() several times: once for each of the return statements at lines 15, 17, 20, and 22.

|  |
| --- |
| def food\_id\_test():  ''' Unit test for food\_id  returns True if good, returns False and prints error if not  good  '''  works = True  if food\_id('orange') != 'Citrus, Fruit':  works = False  print('orange bug in food\_id()')  if food\_id('banana') != 'NOT Citrus, Fruit':  works = False  print('banana bug in food\_id()')  *# Add tests so that all likes of code are visited during test*  if works:  print('food\_id() passed all test')  return works |



**Step 4**

Design a function f(x) that implements the flowchart below. A flow chart is another way to represent an algorithm; input and output are in rectangles, and branching decisions are in diamonds. The exercise illustrates the % operator, called the modulo operator, which identifies the remainder after division. As an example, 13 % 4 =1 , since 13 divided by 4 is 3 with a remainder of 1.

|  |
| --- |
| def f(x):  if (type(x) is int):  if (x % 2 == 0):  if (x % 3 == 0):  return str(x) + ' is a multiple of 6'  else:  return str(x) + ' is an even number'  else:  return str(x) + ' is an odd number'  else:  return 'please specify an integer' |

**Step 5**

Write a set of test cases that can be used to visit all of the code.

|  |
| --- |
| if (type(x) is int):  if (x % 2 == 0):  If (x % 3 == 0):  return str(x) + ' is a multiple of 3'  elif (x % 4 == 0):  return str(x) + ' is a multiple of 4'  elif (x % 5 == 0):  return str(x) + ' is a multiple of 5'  elif (x % 6 == 0):  return str(x) + ' is a multiple of 6'  else:  return str(x) + ' is an even number'  else:  return str(x) + ' is an odd number' else:  return 'please specify an integer' |

**Step 6**

Explain the difference between + as concatenation and + as numeric addition.

String concatenation is just the addition of two or more strings to form a larger string like so:

|  |
| --- |
| var string1 = 'I am' var string2 = 'a string' var output = string1 + ' ' + string2 return output |

this segment of code would return the string “I am a string” because string1 and string2 were concatenated together with a space in the middle. If you want to concatenate an integer, float, or double into the string you would have to convert it into a string first, although some programming languages do that for you and allow you to concatenate the raw number with the string. When you’re adding multiple numbers together it would actually add them together like so:

|  |
| --- |
| x = 5 y = 6 z = x + y return z |

This block of code would return 11 because 5 + 6 is equal to 11. If you were to do this:

|  |
| --- |
| x = '5' y = '6' z = x + y return z |

it would return “56” because it’s treating it like a string concatenation and not the addition of two numbers.

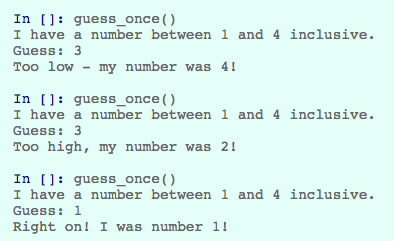
**Step 7**

1. Explain how line 11 works, using the explanation of line 9 as a model.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11: | from \_\_future\_\_ import print\_function *# must be first in file* import random   def guess\_once():  secret = random.randint(1, 4)  print('I have a number between 1 and 4.')  guess = int(raw\_input('Guess: '))  if guess != secret:  print('Wrong, my number is ', secret, '.', sep='')  else:  print('Right, my number is', guess, end='!\n') |

|  |
| --- |
| Line 9 is just a simple print statement where the first 3 parameters are being concatenated together and the “sep” parameter determines the separator between concatenations. The same concatenation method is used on line 11 except an “end” parameter is being used which is inserted at the end of the concatenated string. In this case it puts “!\n” and the end where “\n” is a line break. Due to the lack of a space at the end of the “Right, my number is” string it’s safe to assume that using this method of concatenation the separator value is equal to “ ”. |

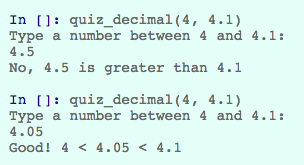
b. Modify the program to provide output as shown below.



|  |
| --- |
| def guess\_once():  secret = random.randint(1, 4)  print('I have a number between 1 and 4.')  guess = int(raw\_input('Guess: '))  if guess != secret:  if (guess > secret):  print('Too high, my number was ', str(secret), '.', sep='')  elif (guess < secret):  print('Too low - my number was ', str(secret), '!', sep='')  else:  print('Right on! I was number ', str(guess), end='!\n') |

**Step 8**

Create a function quiz\_decimal(low, high) that asks the user for a number between low and high and tells them whether they succeeded.



|  |
| --- |
| def quiz\_decimal(low, high):  response = raw\_input('Type a number between ' + str(low) + ' and ' + str(high) + ': ')  if (float(response) > high or float(response < low)):  if (float(response) > high):  print('No, ' + response + ' is greater than ' + str(high))  if (float(response) < low):  print('No, ' + response + ' is less than ' + str(low))  else:  print('Good! ' + str(low) + ' < ' + response + ' < ' + str(high)) |