Automating The Boring Stuff with Python

(Compiled answers to end of chapter questions)

Contents

[Chapter 1 Practice Questions – Python Basics 3](#_Toc127792985)

[Chapter 2 Practice Questions Flow Control 5](#_Toc127792986)

# Chapter 1 Practice Questions – Python Basics

|  |  |
| --- | --- |
| **Q1** | **Which of the following are operators, and which are values?**  **\***  **‘hello’**  **-88.8**  **-**  **/**  **+**  **5** |
| A | \* (multiplication), / (division) and + (addition) are operators. The rest are values |
| **Q2** | **Which of the following is a variable, and which is a string?**  **spam**  **‘spam’** |
| A | spam is a variable and ‘spam’ is a string  Strings always start and end with quotes |
| **Q3** | **Name three data types** |
| A | 1. Integer 2. String 3. Floating-point numbers |
| **Q4** | **What is an expression made up of? What do all expressions do?** |
| A | An expression is made up of values and operators which can always evaluate to a single value.  A single value with no operators is also considered an expression, though it evaluates only to itself. |
| **Q5** | **This chapter introduced assignment statements, like spam = 10. What is the difference between an expression and a statement?** |
| A | ~~The difference between an assignment and an expression is that expression is never stored in memory whilst an assignment is stored in memory~~  An expression is a combination of values, variables and operators that evaluates to a single value. A statement does not. A statement is an instruction that the Python interpreter can execute which may or may not evaluate to a value e.g. print() |
| **Q6** | **What does the variable bacon contain after the following code runs?** |
| A | bacon = 20  bacon + 1  The variable bacon evaluates to 21 (only if reassigned i.e., bacon = bacon + 1.) |
| **Q7** | **What should the following two expressions evaluate to?**  **‘spam’ + ‘spamspam’**  **‘spam’ \* 3** |
| A | Both expressions evaluate to ‘spamspamspam’  The first expression being string concatenation and the second expression being string replication |
| **Q8** | **Why is eggs a valid variable name while 100 is invalid?** |
| A | Variables should obey the following rules:   1. Can only be one word 2. Can use only letters, numbers, and the underscore (\_) character. 3. Can’t begin with a number |
| **Q9** | **What three functions can be used to get the integer, floating-point number, or string version of a value?** |
| A | str(), int() and float() |
| **Q10** | **Why does this expression cause an error? How can you fix it?**  **‘I have eaten ‘ + 99 + ‘ burritos.’** |
| A | This causes an error because the expression is trying to concatenate a string with an integer. It can be fixed using the str() function to convert 99 into a string i.e. str(99) |

# Chapter 2 Practice Questions Flow Control

|  |  |
| --- | --- |
| **Q1** | **What are the two values of the Boolean data type? How do you write them?** |
| A | Boolean data type has only two values: True and False |
| **Q2** | **What are the three Boolean operators?** |
| A | The three Boolean operators (and, or, and not) are used to compare Boolean values |
| **Q3** | **Write out the truth tables of each Boolean operator (that is, every possible combination of Boolean values for the operator and what they evaluate to)** |
| A | The and operators’ truth table:  True and True = True  True and False = False  False and True = False  False and False = False  The or operator’s truth table:  True or True = True  True or False = True  False or True = True  False or False = False  The not operators’ truth table:  Not True = False  Not False = True |
| **Q4** | **What do the following expressions evaluate to?**   1. **(5 > 4) and (3 == 5)** 2. **Not (5 > 4)** 3. **(5 > 4) or (3 == 5)** 4. **Not ((5 > 4) or (3 ==5))** 5. **(True and True) and (True == False)** 6. **(not False) or (not True)** |
| A | 1. False 2. False (The not operator simply evaluates to the opposite Boolean value) 3. True 4. False 5. False 6. True |
| **Q5** | **What are the six comparison operators?** |
| A | == | Equal to  != | Not equal to  < | Less than  > | Greater than  <= | Less than or equal to  >= | Greater than or equal to |
| **Q6** | **What is the difference between the equal to operator and the assignment operator?** |
| A | The equal to operator asks whether two values are the same as each other and evaluate to a Boolean value of True or False  The assignment operator puts the value on the right into the variable on the left |
| **Q7** | **Explain what a condition is and where you would use one** |
| A | Conditions are expressions which always evaluate down to a Boolean value (True or False) |
| **Q8** | **Identify the three blocks in this code:**  **spam = 0**  **If spam == 10:**  **print(‘eggs’)**  **if spam > 5:**  **print(‘bacon’)**  **else:**  **print(‘ham’)**  **print(‘spam’)**  **print(‘spam’)** |
| A | 3; separated by the colon  The colon is there to declare the start of an indented block |
| **Q9** | **Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! If anything else is stored in spam.** |
| A | if spam == 1:  print(‘Hello’)  elif spam == 2:  print(‘Howdy’)  else:  print(‘Greetings!’) |
| **Q10** | **What can you press if your program is stuck in an infinite loop?** |
| A | If you ever run a program that has a bug causing it to get stuck in an infinite loop, press CTRL-C |
| **Q11** | **What is the difference between break and continue?** |
| A | The difference is that the break statement causes a while loop to terminate based on a set condition(s) being met whereas the continue statement also terminates a while loop until a set condition(s) is met but if the condition is not met it goes back to the beginning of the while loop to retest the condition(s) unlike the break statement which does not return the program to the beginning of the while loop if the condition(s) is not met. |
| **Q12** | **What is the difference between range(10), range(0, 10), and range(0, 10, 1) in a for loop?** |
| A | A block of code can be run a specified number of times using the for loop statement e.g.  For i in range(5):  print(‘Hello’)   1. range(10) will run a block of code 10 times with each sequence being 0-9 2. range(0, 10) will run a block of code a block of code 10 times (10 not inclusive). These two arguments allow for the range iterations to start from any number other than 0 if so wish 3. range(0, 10, 1) will run a block of code 10 times. The third argument is by how many numbers the loop should skip e.g. range(0, 10, 2) will skip each sequence by 1 and each sequence will thus be: 0, 2, 4, 6 and 8 |
| **Q13** | **Write a short program that prints the numbers 1 to 10 using a for loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.** |
| A | For loop:  for I in range(1, 11):  print(i)  While loop:  I = 1  while I <= 10:  print(i)  I = I + 1 |
| **Q14** | **If you had a function named bacon() inside a module named spam, how would you call it after importing spam?** |
| A | import spam  spam.bacon() |

# Chapter 3 Practice Questions – Functions

|  |  |
| --- | --- |
|  | Chapter 3 Practice Questions – Functions |
| **Q1** | **Why are functions advantageous to have in your programs?** |
| A | They help organise code into ‘black boxes’ that perform certain tasks by taking in parameters and giving out outputs which can be used in other parts of a program |
| **Q2** | **When does the code in a function execute: when the function is defined or when the function is called?** |
| A | When the function is called |
| **Q3** | **What statement creates a function?** |
| A | Def |
| **Q4** | **What is the difference between a function and a function call?** |
| A | A function call involves passing in parameters for the function to process and give an output whereas a function is the code of the function i.e. when it is defined |
| **Q5** | **How many global scopes are there in a Python program? How many local scopes?** |
| A | 1 global scope is created when the program begins |
| **Q6** | **What happens to variables in a local scope when the function call returns?** |
| A | Variables assigned within a function exist in the local scope and when the function returns, the local scope is destroyed along with any variables |
| **Q7** | **What is a return value? Can a return value be part of an expression?** |
| A | A return value is the value that a function call evaluates to. An expression consists of values and operators and can always evaluate down to a single value therefore return values are normally not part of an expression |
| **Q8** | **If a function does not have a return statement, what is the return value of a call to that function?** |
| A | Python adds return value = None to the end of any function definition with no return statement |
| **Q9** | **How can you force a variable in a function to refer to the global variable?** |
| A | If you need to modify a global variable from within a function, use the global statement |
| **Q10** | **What is the data type of None?** |
| A | None is the only value of the NonType data type |
| **Q11** | **What does the** *import areallyourpetsnamederic* **statement do?** |
| A | The import statement itself imports a module which contains many functions |
| **Q12** | **If you had a function named bacon() in a module named spam, how would you call it after importing spam?** |
| A | You call it by typing the module name followed by a period and then the function name as below: spam.bacon(*arguments*) |
| **Q13** | **How can you prevent a program from crashing when it gets an error?** |
| A | The code that could potentially have an error is put in a try clause. When code in a try clause causes an error, the program execution immediately moves to the code in the except clause |
| **Q14** | **What goes in the try clause? What goes in the except clause?** |
| A | The code that could potentially cause an error goes in the try clause while the code in the except clause is what is executed right after an error is encountered in the try clause |

# Chapter 4 Practice Questions – List

|  |  |
| --- | --- |
|  | Chapter 4 Practice Questions - Lists |
| **Q1** | **What is []** |
| A | Indicates the beginning and ending of a list; an empty list. A list value that contains no items |
| **Q2** | **How would you assign the value ‘hello’ as the third value in a list stored in a variable named spam? (Assume spam contains [2,4,6,8,10])**  **For the following three questions, let’s say spam contains the list [‘a’,’b’,’c’,’d’]** |
| A | Spam[2] = ‘hello’ |
| **Q3** | **What does spam[int(int(‘3’\*2)/11)] evaluate to?** |
| A | ‘d’  Because ‘3’\*2 = 33/11 = 3 = index of spam list |
| **Q4** | **What does spam[-1] evaluate to?** |
| A | ‘d’  Because the integer -1 refers to the last index in a list |
| **Q5** | **What does spam[:2] evaluate to?**  ***For the following three questions, let’s say bacon contains the list [3.14, ‘cat’, 11, ‘cat’, True]*** |
| A | ‘a’,’b’  Because the integer 2 is where the slice ends i.e. at ‘c’ |
| **Q6** | **What does bacon.index(‘cat’) evaluate to?** |
| A | 1 i.e. gives the index number or position of the item in the list |
| **Q7** | **What does bacon.append(99) make the list value in bacon look like?** |
| A | [3.14, ‘cat’, 11, ‘cat’, True, 99] The append method adds the new item to the end of the list |
| **Q8** | **What does bacon.remove(‘cat’) make the list value in bacon look like?** |
| A | [3.14, 11, ‘cat’, True] The remove method removes the first instance of a duplicate item |
| **Q9** | **What are the operators for the list concatenation and list replication?** |
| A | For list concatenation the operator is:  += e.g. names += ‘Steve’  For list replication the operator is:  \*= e.g. names \*= 2  += Addition Assignment; Adds the value and the variable and assigns the new result to the variable |
| **Q10** | **What is the difference between the append() and insert() list methods?** |
| A | Both add new values to a list but the difference is that append() adds the value to the end of the list while the insert() method can insert the value at any point in the list |
| **Q11** | **What are two ways to remove values from a list?** |
| A | Use the remove() method to remove a value passed to it as an argument. Use the del statement when you know the index of the value to be removed e.g. del spam[1] removes the value at index 1 |
| **Q12** | **Name a few ways that list values are similar to string values** |
| A | 1. They share some methods such as len() which do not change their values 2. They both can be copied |
| **Q13** | **What is the difference between lists and tuples?** |
| A | Tuples are typed with parentheses () instead of square brackets and more importantly tuples are immutable i.e. cannot have their values modified, appended or removed |
| **Q14** | **How do you type the tuple value that has just the integer value 42 in it?** |
| A | Add a trailing comma after the value inside the parentheses otherwise Python will think it’s just a value inside a parentheses e.g. (42,) |
| **Q15** | **How can you get the tuple form of a list value? How can you get the list form of a tuple value?** |
| A | Use the function tuple([]) to get the tuple form of a list value and the function list(()) to get the list form of a tuple |
| **Q16** | **Variables that “contain” list values don’t actually contain lists directly. What do they contain instead?** |
| A | They contain references to the list values so that for example if you have two variables (different names) reference to the same list and alter the list referring to the first variable; the second variable list is also altered. |
| **Q17** | **What is the difference between copy.copy() and copy.deepcopy()?** |
| A | They are methods from the copy module:   1. copy.copy(): can be used to copy a mutable list or dictionary so that a change to the copy does not change the original due to the referencing nature of lists and dictionaries 2. deepcopy(): copies inner list of a list |

Chapter 5 Practice Questions - Dictionaries and Structuring Data

|  |  |
| --- | --- |
|  | Chapter 5 Practice Questions |
| **Q1** | **What does the code for an empty dictionary look like?** |
| A | {} |
| **Q2** | **What does a dictionary value with a key ‘foo’ and a value 42 look like?** |
| A | {‘foo’: ‘42’} |
| **Q3** | **What is the main difference between a dictionary and a list?** |
| A | Dictionaries have indexes called keys and can be chose to be an data type Lists have indexes that are integers only and cannot be chosen |
| **Q4** | **What happens if you try to access spam[‘foo’] if spam is {‘bar’: 100}?** |
| A | Trying to access a key that does not exist in a dictionary will result in a KeyError error message |
| **Q5** | **If a dictionary is stored in spam, what is the difference between the expressions ‘cat’ in spam and ‘cat’ in spam.keys()?** |
| A | ‘cat’ in spam checks whether ‘cat’ exists as a key or a value in the dictionary ‘cat’ in spam.keys() checks whether ‘cat’ exists as a key only  The return value is True or False |
| **Q6** | **If a dictionary is stored in spam, what is the difference between the expressions ‘cat’ in spam and ‘cat’ in spam.values()?** |
| A | ‘cat’ in spam checks whether ‘cat’ exists as a key or a value in the dictionary ‘cat’ in spam.keys() checks whether ‘cat’ exists as a value only  The return value is True or False |
| **Q7** | **What is a shortcut for the following code? if ‘color’ not in spam:  spam[‘color’] = ‘black’** |
| A | spam.setdefault(‘color’,’black’)  This code means: check to see if the there is a key called (by the first argument); if the key does not exist then set the key and its value according to the arguments.  If the key does exist, then the key’s value is returned |
| **Q8** | **What module and function can be used to “pretty print” dictionary values?** |
| A | Module: pprint and Function: pprint  As in pprint.pprint(someDictionaryValues) |

Resume Chapter 6