

Exercises – Week 3. Loops and Data Structures

Part 1. Loops

To complete this week's exercises you will also need to download the `Question2.py` and `HowManySquares.py` files from Blackboard.

Exercise 1 - For Loops (Essential)

“For” loops are typically used when you know how many times you need to repeat something, before ending the loop. You specify a limit to the number of loops you wish to run the code for, and then the loop will automatically stop.

1. A `for` loop can loop over anything that is *iterable* such as strings, tuples, lists and dictionaries. Try the following:

```
Words = "Hello World"
for Letter in Words:
    print(Letter)
```

Try with some different words.

2. Open and run the `Question2.py` program. What does it do? Research any new functions using `help()`.
3. In this program the loop is written in the following format:

```
for Value in range(1,11,1):
```

Why do we use the value 11 here? What are the values of the range?

4. In Q1 we used the string `Words` as the loop iterator. In the previous question we also saw that you can use the function `range()` as a loop iterator. This is a very useful and important function in Python. Call `help()` on `range()` to see how it works. What will be returned by calling `range(1, 11, 2)`?
5. Can you write a loop which now prints both each letter and its position in the string `Words`?
Hint: The `len()` function returns the length of a string.
6. Find the mistake(s) in the following program, which is meant to sum the first 10 multiples of 5:

```
sum = 0
for i in range(1,10)
sum = sum + 5 * i
```

Fix the program and show that `sum = 275`.

7. Compute the factorial of 10. Recall that the factorial of an integer n is defined as $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$.
8. Using a `for` loop and the `break` keyword, determine how many positive cubic numbers are less than 2,000. Recall that cubic number is a number of the form n^3 where n is an integer.

Exercise 2 - While Loops (Essential)

“While” loops are used when the number of times the program needs to loop is not known beforehand. The `while` loop checks a condition statement at the beginning of each loop and will carry out the loop as long as this condition is true.

1. Can you finish the `while` loop by replacing the `<?>` in the code below?

Note: `<?>` is used as a placeholder to represent something missing - this could be an operator, a variable name, function name, etc. Multiple `<?>` in the same question are not necessarily representing the same thing.

```
Word = "Hello World"
TargetLetter = <?>
i = 0
while <?>:
    i += <?>
print("Target letter is at position", i)
```

Try changing `Word` and `TargetLetter`. Why do we have to increment the `i`?

2. How would you change the code to print all the occurrences of the letter?

Hint: What type of loop would be best?

3. Finding characters or substrings in a string is very useful and so Python has a built-in function `str.find()`. Test your program is right by comparing with the `str.find()` function for some different target letters and words.

Hint: Remember the `help()` function.

4. Open and run the `HowManySquares.py` program. Can you change the code to use the `+=` operator?
5. In the `HowManySquares.py` example we do not know when the cumulative sum will exceed 1,000,000. Therefore, we use a `while` loop until the sum exceeds this limit, and therefore the condition no longer evaluates to `True`. Can you add a line that shows the results for every step of the `while` loop?
6. Can you rewrite Q1 using a `for` loop, an `if` clause and `enumerate()`?

Hint: research the `enumerate` function. To get a feel for how this works, run the following code

```
s = 'hello'
enum = enumerate(s)
print(list(enum))
```

Exercise 3 - More Loops (Advanced)

Use loops and conditionals to solve the following problems:

1. Use two `for` loops to compute the double sum

$$S = \sum_{i=1}^{10} \sum_{j=0}^5 j^2(i+j)$$

2. The value of π can be approximated using the Leibniz formula:

$$\pi_N = \sum_{n=0}^N \frac{8}{(4n+1)(4n+3)}$$

where N is a large number. Taking the limit as $N \rightarrow \infty$ produces the exact value of π , but this requires evaluating an infinite number of terms, which is impossible on a computer. Therefore, we can only approximate the value of π by using a finite number of terms in the sum. Use this formula to compute approximations to π by taking $N = 100$, $N = 1,000$, and $N = 10,000$. Given that $\pi = 3.141592653589793\dots$, what is the error in the approximation in each of these cases? Note that the error is defined as $|\pi - \pi_N|$. The function `abs` can be used to compute the absolute value in Python. What value of N is needed to ensure an error that is less than 10^{-6} ?

3. The Fibonacci sequence is a sequence of numbers where each number is the sum of the two preceding ones. The first two numbers in the Fibonacci sequence are 0 and 1. The sequence therefore starts as:

0, 1, 1, 2, 3, 5, 8, ...

Each number in this sequence is called a Fibonacci number. How many Fibonacci numbers are less than (i) 100, (ii) 1,000, (iii) 10,000?

4. Write a program that assigns two variables, say `A` and `B`, to the throws of two random dice. Your program should keep reassigning dice throws to `A` and `B` until `A == B`. Then, create another variable to count the number of times the program assigns `A` and `B` random values until both numbers are equal. When this happens, print out a success message and the number of assignments it took for `A == B`.

Hint: Recall you replicated dice throws for an exercise in Week 1.

5. Implement the “Number Guessing Game”, which works in the following way:

- Pick a random number between 1 and 100.
- Ask the user to guess the number using the `input()` function.
- Tell the user if they are correct and stop the program. If they are incorrect, tell them whether their guess is too low or too high.
- Repeat until the user has guessed correctly.
- Congratulate the user on guessing the number and tell them how many guesses it took them.

What about when the user guesses a number out of range? Add a check to your program to instruct the user to enter a guess within the accepted range.

Part 2. Data structures

Exercise 4 - Lists (Essential)

1. Make two lists containing the values `[1,2]` and `[3,4]`.
2. Change the value 1 to the value 5.
3. Sort the lists.
Hint: Have a look at `help(list)`.
4. Make a nested list that contains both lists.
5. Use two loops to print out all the values in the nested list (2x2 matrix) one by one.
6. Write a program that asks the user to input a list of 10 words (strings) and then creates a list containing the length of each word. Print out each word and word length, like so:

Word: Algorithm - Word length: 9

Hint: You can read all 10 words at a time, as one large string, and use the `.split()` function on the string. Read the documentation if you are unfamiliar with the `split` function. Then loop through the resulting list of words and print out the length of each word.

7. List comprehension is an elegant way to produce lists using loops and conditional statements. Read how to do this here: <https://www.pythonforbeginners.com/basics/list-comprehensions-in-python>

Using list comprehension, create lists of the following between 0 and 100:

- odd numbers
- multiples of 3
- prime numbers (NB: this is quite tricky and can be considered an advanced question)

Exercise 5 - Tuples (Essential)

1. Have a look at `help(tuple)`.
2. Make a tuple named `FondueIngredients` containing the values “gruyere” and “vacherin”.
3. Print all the items in the tuple.
4. Change the value “gruyere” to the value “cheddar”. Does it work? Why?
Note: Fondue recipes are sacred.
5. Is there a function to remove the last item of the tuple? How else could you do it?

Exercise 6 - Sets (Essential)

1. Have a look at `help(set)`.
2. Make two set `s1 = {1,2,5,5,8}` and `s2 = {1,2,4,9,2}`. Print out the sets, are there any duplicates?
3. Can you access an element of the set based on index e.g. `s1[2]`?
4. Use the keyword `in` to check if 4 is in both sets.
5. Use the operators `&`, `|`, `-`, `^`. What do they do?
6. Remove the value 1 from the first set, and add the value 6.

Exercise 7 - Dictionaries (Essential)

1. Have a look at `help(dict)`.
2. Make a dictionary that contains `{"Jill":21, "Sally":20, "Bob":20, "Harry":21}`. Remember to give it a sensible name
3. Print out all the keys in the dictionary. Use `help(dict)` to work out how to do this.
4. Add the item `"Rachel":19` to the dictionary.
5. Remove the item `"Bob"`.
6. Add the item `"Jill":22` to the dictionary. Are there two Jills now?
7. Check if `"Harry"` is in the dictionary.

Exercise 8 - FizzBuzz Game (Advanced)

In the game FizzBuzz, we count from 1 to n , replacing any multiple of 3 with the word “Fizz” and any multiple of 5 with the word “Buzz”. As follows:

“1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, ...”.

1. Create two variables `Mult3` and `Mult5`, setting their values to be the strings `"Fizz"` and `"Buzz"`, respectively.
2. Create an additional variable `Limit`, which will be the number we count up to.
3. At the beginning of your program, after you have assigned `Mult3` and `Mult5` their values, you will need to ask the user to **input** a value for `Limit`. This can be done using the `input()` function, which waits for the user to input some *string* when you run your program, before continuing.
Hint: you will need to **convert** the input string created by `input()` into an integer, using `int()`.
4. The computer should say each number from 1 to `Limit`, replacing each multiple of 3 with the word “Fizz” and each multiple of 5 with “Buzz”. What kind of **loop** will you need for this?
5. You will also need to use the `%` operator, which returns the remainder of a division e.g. $4 \bmod 3 = 1$ and $15 \bmod 3 = 0$, indicating that 15 is a multiple of 3. You will need to check whether each number is either a multiple of 3, a multiple of 5, or *both*.
Hint: Start with the basic loop, printing out each number, and then work on replacing it with “Fizz”, “Buzz”, or “FizzBuzz”, in stages.