Introduction to Programming - Part 2 Exercises

Loops

Exercises

1. Print the numbers 0 to 10 using a for loop.
2. Print the numbers 0 to 10 using a while loop.
3. Print the list of numbers below using a for loop.

```
nums = [0, 2, 8, 20, 43, 82, 195, 204, 367]
```

4. Print the message 'Done!' after printing the numbers 0 to 10 with a for loop. Hint: use the `for-else` construct.
5. Take the below two lists and use a nested for loop to determine if any elements from the first list are in the second. If it finds a match, print out the name of the element.

```
list1 = ["apple", "banana", "cherry", "durian", "elderberry", "fig"]
list2 = ["avocado", "banana", "coconut", "date", "elderberry", "fig"]
```

6. Using a while loop, on every iteration generate a random number. If it's a multiple of 5, **break** from the loop. If it's a multiple of 3, end the current iteration with **continue** and print a message to show you are skipping. If it's neither, print the number and continue the loop.

   When the loop has been broken, print a message indicating that this has happened before the program exits.

   Hint:

```
import random
x = random.randint(1,100)
```

Dictionaries

1. Add a new key-value pair to the below car dictionary for the colour. Print out the colour to verify it worked.
2. Update the value of the model in the car dictionary. Print out the new value to verify it worked.
3. Delete the model key-pair from the car dictionary. Print out the entire dictionary to verify it worked.
4. Use the `items()` function to loop through the dictionary and print each key-value pair like so:

```
key: x, value: y
```

   Hint: `for key, value in x.items():` Hint: You will need to cast the values to a string

   Car dictionary:

```
car = {
'brand': 'Ford',
'model': 'Mustang',
'year' : 1964,
'isNew': False
}
```

Functions

1. Create a function that takes two numbers as arguments and return the sum. Print the result.
2. Extend the above by passing an arbitrary amount of integers to a function and return the result. Print the result. Hint: use `*args`.
3. Pass an arbitrary amount of named arguments to a function and create a dictionary. The keys will be the names of the arguments and the values are assigned values of the named arguments. Hint: use `**kwargs`.
4. Create a scientific/basic calculator that makes use of separate functions to perform calculations, such as: `add`, `divide`, `area_of_a_circle` etc...
5. Add some form of user interface to allow the user to perform calculations
6. Print out a nice result / log to the screen

Bonus

Create a fibonacci function that returns `fib(n)`. So if i request `fib(100)` it returns the 100th position in the sequence. `n` is undetermined so working out a finite sequence beforehand is not acceptable ;)

```py
```

The fibonacci sequence is the sum of the previous two values

1, 1, 2, 3, 5, 8, 13...

fib(7) # 13 fib(50) # ?