PACKAGE 1:

Exercise 1

Function 1: A useless function

Create a function called **useless**, that takes 3 parameters of any type, does nothing with them, and returns the string **That was a waste of time**. Running your code in the shell should look like this:

```
>>> useless(1, 2, 3)
 'That was a waste of time'
>>> useless("Hello", "Goodbye", 999)
 'That was a waste of time'
```

Function 2: Squaring a number

Create a function called **square_me**, that takes 1 number (int or float) parameter and returns the square of that number as a result (you don't have to worry about what it does if something other than a float/int is passed in). Your code should behave as follows:

```
>>> square_me(3)
 9
>>> square_me(9.5)
 90.25
```

Function 3: Biographical Data

Create a function called **student_data**, that takes 4 parameters, a name (a string), age (an integer), student number (a string) and whether they are enrolled in ETY (a boolean), and returns a string containing that information in the following format: **<student number, name, age, enrolled>**. It may be helpful to remember that you can cast a number or a boolean to a string using the str function. Your code should work as follows:

```
>>> student_data("Brian", 35, "1234567", False)
 '<1234567, Brian, 35, False>'
 >>> student_data("Nick", 97, "0000001", True)
 '<0000001, Nick, 97, True>'
```

Exercise 2

1: Percentage

Complete a function called **percentage**, which takes as its parameters two floats, and returns a number representing the first parameter as a percentage of the second parameter. We've provided you with the first few steps of the Design Recipe to get you started.

2: Term Mark

Write a function called **term_work_mark**, which calculates your term mark (as a contribution to your final grade) given your marks on all of the coursework components). Using the default weights this will return a maximum result of 60 (since the exam is worth 40% of your final grade):

- your mark on assignment 0 (by default this is out of 25)
- your mark on assignment 1 (by default this is out of 50)
- your mark on assignment 2 (by default this is out of 100)
- your mark on the exercises (by default this is out of 10)
- your mark on the quizzes (by default this it out of 5)
- your term test marks (by default this is out of 50)

In the starter code provided in class, there are already several global variables that you will need to compute this score. The contribution of each piece of coursework is computed as:

((mark-you-received) / (maximum-mark)) * (weight)

Your coursework mark is the sum of these component contributions. An example output is given below (note that this may vary depending on the state of the global variables):

```
>>> term_work_mark(25, 50, 100, 10, 5, 50)
 60.0
>>> term_work_mark(20, 45, 70, 8, 4, 40)
 47.9
```

Hint: Rather than writing the same computation over and over, you can make your life easier by using a helper function... perhaps some nice person may have left a useful function in the file before uploading it...

3: Final Mark

Write a function called **final_mark**, which performs a similar calculation to **term_work_mark**, but takes an additional parameter, your final exam mark (be default, out of 100), and returns your final mark for the term out of 100. An example output is given below:

```
>>> final_mark(25, 50, 100, 10, 5, 50,100)
 100.0
>>> final_mark(20, 45, 70, 8, 4, 40, 73)
 77.1
```

Hint: Be lazy! If you use the other functions you've written, this shouldn't take more than 3-4 lines of code.

4: See If You Pass

Write a function called **is_pass** that takes the same parameters as **final_mark**, and returns a boolean representing whether you passed the course. Remember that in order to pass the course you must get a final overall mark at or above a certain threshold (by default 50) as well as an exam mark at or above a certain threshold (by default 40). An example output is given below:

```
>>> is_pass(20, 45, 70, 8, 4, 40, 41)
 True
```

```
>>> is_pass(20, 45, 70, 8, 4, 40, 39)
 False
>>> is_pass(10, 21, 12, 2, 1, 15, 23)
 False
```

Hint: See hint for the previous section. 3-4 lines of code should be plenty. Don't do more work than you need to.

Exercise 3

GPA Calculator

Create a function called **percent_to_gpa** that takes a percentage mark (int) as input, and returns the corresponding Grade Point Average (float). GPA is calculated as follows:

Percentage Mark	GPA
90-100	4.0
85-89	4.0
80-84	3.7
77-79	3.3
73-76	3.0
70-72	2.7
67-69	2.3
63-66	2.0
60-62	1.7
57-59	1.3
53-56	1.0
50-52	0.7
0-49	0.0

You can safely assume that we will not give you values outside of the given ranges for testing, but your docstring should make it clear what values are acceptable

Card Naming

Write a function called **card_namer** that takes two single character strings, representing the value and suit of a card following the shorthand below, and returns the full name of the card. Some sneaky cheaters have been trying to slip in fake cards, like the 9 of triangles. So, if the **suit** input isn't one of the recognized inputs the function should return **'CHEATER**!'. You may assume that **value** will be always be a valid input.

Input	Value
A	Ace
2 ... 9	2 ... 9
T	10
J	Jack
Q	Queen
K	King

Input	Suit
D	Diamonds
C	Clubs
H	Hearts
S	Spades

An example output of the function would be:

```
>>> card_namer('Q','D')
 'Queen of Diamonds'
>>> card_namer('9','S')
 '9 of Spades'
>>> card_namer('8','T')
 'CHEATER!'
```

Our Own str() Function

We've been using the **str()** function in class quite frequently to turn other data types into strings. However, I'd now like to build our own version, so that we can control exactly how the conversion happens. Create a function called **my_str** that takes an **object** as input, and returns a string representation of that object. However, we don't want to use the boring old built-in representations, we're going to make our own.

- If the input is a string, just return it as it is
- If the object is a boolean, return "**YES**" (for True) or "**NO**" (for False)
- If the object is an integer:
  - If it's less than or equal to 10, return "**Small Number**", for 11 - 99 return "**Medium Number**", for anything larger return "**Large Number**"
- If the object is a float, return a standard string representation, but rounded to at most 2 decimal places
  - Try to do this with just a single call to **round()**
- If the object is any other data type, simply return the phrase "I dunno"

Some sample output from the function:

```
>>> isinstance("Hello",str)
 True
>>> isinstance(True,bool)
 True
>>> my_str("Hello")
 'Hello'
>>> my_str(False)
 'NO'
>>> my_str(42)
 'Medium Number'
>>> my_str(42.0)
 '42.0'
>>> my_str(3.1415926)
 '3.14'
>>> my_str([1, 2, 3])
 'I dunno'
```