In [2]:

```
### STUDENT ANSWER
# import relevant packages
import numpy as np
import nibabel
```

2. [3pts] Basic Data Types

- **(a)** Divide 6 by 2 using floating point division. Store this value in the name `a` . Then divide 6 by 2 using integer division. Store this value in the name `b` . Print both values.

In [3]:

```
a = 6 / 2
b = 6 // 2

print(a)
print(b)
```

```
3.0
3
```

In [4]:

```
# This checks whether you have created the names a and b, it does not check the result
ok.grade("q1_2a")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[4]:

```
{'passed': 2, 'failed': 0, 'locked': 0}
```

In [5]:

```
# This is for later, when the homework is graded. Leave it commented out until then.
# ok.grade("q1_2a_full")
```

- **(b)** Store the floating point value `5.0` into a name called `f` . Store the integer value `5` into a name called `i` . Divide `f` by `i` , store that in `f_divide_i` and print the type

In [6]:

```
f = 5.0
i = 5
f_divide_i = f/i
print(type(f_divide_i))
```

```
<class 'float'>
```

In [7]:

```
# To check whether your answer contains the right names
ok.grade("q1_2b")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
```

```
[ooooooooook] 100.0% passed
```

Out[7]:

```
{'passed': 3, 'failed': 0, 'locked': 0}
```

In [8]:

```
# For after grading
# ok.grade("q1_2b_full")
```

- **(c)** Create the string `"5"` and store it in a name called `s`. Then multiply `s` by `i`, store the result in `si` and print out the result.

In [9]:

```
s = "5"
si = s * i
print(si)
```

```
55555
```

In [10]:

```
# To check whether your answer contains the right names
ok.grade("q1_2c")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[10]:

```
{'passed': 2, 'failed': 0, 'locked': 0}
```

In [11]:

```
# For after grading
# ok.grade("q1_2c_full")
```

3- [2pts] Lists and Tuples

- **(a)** Create a tuple that contains each words of the following sentence as a separate object: `I love data science`. Store this tuple in a name called `tup`.

In [12]:

```
tup = ('I', 'love', 'data', 'science')
print(tup)
```

```
('I', 'love', 'data', 'science')
```

In [13]:

```
# To check whether your answer contains the right names
ok.grade("q1_3a")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[13]:

```
{'passed': 1, 'failed': 0, 'locked': 0}
```

In [14]:

```
# For after grading
# ok.grade("q1_3a_full")
```

- **(b)** Create an empty list and store it in a name called `l` . Then append `tup` , append another empty list, and append the number `5` . Finally print out this list.

In [15]:

```
l = []
l.append(tup)
l.append([])
l.append(5)
print(l)
```

```
[('I', 'love', 'data', 'science'), [], 5]
```

In [16]:

```
# To check whether your answer contains the right names
ok.grade("q1_3b")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[16]:

```
{'passed': 1, 'failed': 0, 'locked': 0}
```

In [17]:

```
# For after grading
# ok.grade("q1_3b_full")
```

4- [3pts] Creating Arrays

- **(a)** Create a 1-d array that is a sequence of even numbers between 50 and 100, inclusive. Store this in a name called `seq_50_100` and print it out.

In [18]:

```
seq_50_100 = np.arange(50, 101, 2)
print(seq_50_100)
```

```
[ 50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80  82  84
  86  88  90  92  94  96  98 100]
```

In [19]:

```
# To check whether your answer contains the right names
ok.grade("q1_4a")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[19]:

```
{'passed': 1, 'failed': 0, 'locked': 0}
```

In [20]:

```
# For after grading
# ok.grade("q1_4a_full")
```

- **(b)** Create another sequence that goes from `0.50` to `1.00` inclusive in increments of `0.02` . Store this in a name called `seq_half_one` and print it. **HINT** There is a quick way of doing this using a name you've already created.

In [21]:

```
seq_half_one = seq_50_100 / 100
seq_half_one = np.arange(0.5, 1.01, .02)
```

In [22]:

```
# To check whether your answer contains the right names
ok.grade("q1_4b")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Out[22]:

```
{'passed': 1, 'failed': 0, 'locked': 0}
```

In [23]:

```
# For after grading
# ok.grade("q1_4b_full")
```

In [24]:

```
seq_half_one[-1]
```

Out[24]:

```
1.0000000000000004
```

- **(c)** Create a 3-D array of integers that all have the value `1` . Make it size `4 x 5 x 3` and store it in a name called `array_3d` , then print out its shape

In [25]:

```
array_3d = np.ones((4,5,3), dtype=np.int)
print(array_3d.shape)
```

```
/tmp/ipykernel_123/1174820440.py:1: DeprecationWarning: `np.int` is a deprecated alias for
the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modif
y any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` o
r `np.int32` to specify the precision. If you wish to review your current use, check the r
elease note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/releas
e/1.20.0-notes.html#deprecations
  array_3d = np.ones((4,5,3), dtype=np.int)
(4, 5, 3)
```

In [26]:

```
# To check whether your answer contains the right names
ok.grade("q1_4c")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
```

```
        Passed: 1
        Failed: 0
    [ooooooooook] 100.0% passed

Out[26]: {'passed': 1, 'failed': 0, 'locked': 0}
```