Hi there! Welcome to the first Python exercises! Before we get started, I need to explain some things. First, every time you see a gray box with the words **In [ ]**, like this:

```
In [ ]:
```

It means that you can type things in it. Usually, there is a part for you to change shown with an *underbar* symbol: _

Each of the gray boxes end with a line that says: `check()`

That line tells me to correct your work and tell you if you did the assignment correctly. So you can just ignore it (but don't change it).

Finally, after you have made your changes press the button at the top that looks like triangle. That will *run* what you've typed.

Shall we begin?

Welcome

When we *program* a project, we often have to store things. These *things* are pieces of information we call "data", and today's assignment is all about making data and holding it in *variables*. Variables are like little boxes that can hold some data. Each variable has a name. This lets us get or change the data in a variable by using its name.

Of course this will make sense as we play with it. Let's begin.

Your first assignment is to put the number 42 into a variable box named "pizza". I told you that a variable could be named just about anything.

```
In [1]: pizza = _

        check(1, 1, pizza)   ## This is one of those lines to ignore right now.
```

```
You need to change the _ symbol to the number 42.
```

Python has two types of numbers:

- A regular, whole number like 1, 8, -142 or 275382876634695077374
- A fraction/decimal number like 5.5 (for $5\ \frac{1}{2}$)

In this assignment, you need to put $2\ \frac{1}{2}$ into the `apple` variable. In Python, we first right all fractions as decimals, so first convert $2\ \frac{1}{2}$ to a decimal, and the assignment will be almost done:

```
In [2]: apple = _

        check(1, 2, apple)
```

. . .

Python can also store words. If Python sees words like `orange`, it will think that is a variable, so if we want to use the words "coconut shells", you have to put them in quotes, like this:

```
carrying = "coconut shells"
```

The same words can be placed in more than one variable box. For instance, this is fine:

```
carrying = "coconut shells"
banging = "coconut shells"
```

In this assignment, I want you to introduce yourself by putting your name in the `name` variable. Remember the quotes!

In [3]:
```
name = _

check(1, 3, name)
```

```
You need to change the _ symbol to the number you name inside quotes, like "David"
```

Words in quotes are called *strings* ... think of it like *stringing words and letters together*. You can use either the double quotes (") or the single quotes ('). We usually choose double quotes if the string or words we are writing uses an apostrophe, like:

```
weapon = "King Arthur's sword is Excalibur"
```

If we want our string of words to be quoted, then we wrap up the string in single quotes, like this:

```
reading = 'I am reading, "The Holy Grail".'
```

Warning: The start and end of your string must use the same type of quote. You can mis-match them. For instance, the following is wrong:

```
bird = "Unladen swallow'   <-- Error!
```

Now it is time to practice. I want you to write a sentence about *Pat's ball* and assign it to the variable `sentence`:

In [4]:
```
sentence = _

check(1, 4, sentence)
```

```
The sentence should say something about Pat's ball.
```

What if you wanted to write a *paragraph* that needed more than one line? To do this, you use *three quotes*, like this:

```
paragraph = """Flora and Charlie are nice kids who like to play ball with the kids across the street.
One day, Flora and Charlie went across the street, but..."""
```

For this next assignment, write a paragraph of words that has at least two lines in it:

In [5]:
```
paragraph = _

check(1, 5, paragraph)
```

```
You need to change the _ symbol to a string with more than one line.
```

A variable can't be named anything. For instance, you can not use a space inside a variable name. You can guess why, right? Look at this example:

```
i like = "spam"
```

That's right, Python will think that "i" and "like" are two different variables. If you want a variable to look like two words, use an *underbar* (we sometimes call that symbol an *underscore*), like this:

```
i_like = "spam"
```

Also, each variable must start with a letter. After that, it can be letters or numbers or the underbar.

What about the spaces around the equals sign? Python doesn't care about those spaces. The following three lines are the same to Python:

```
breakfast="spam and eggs"
breakfast = "spam and eggs"
breakfast         =        "spam and eggs"
```

Why do we use spaces? I think Python code with spaces is more readable. But that is up to you.