User input

Input from the keyboard

Taking input from the user through the keyboard is accomplished using the `input` command. This takes as input a string which is the prompt to the user. Whatever the user inputs can be stored in a variable as a string.

```
name = input("How would you like to be addressed? ")
print(f"Nice to meet you, {name}.")
```

Notice the space after the question '`How would you like to be addressed?`'? What happens if you remove this space? Try it and see.

User error

The prompt above is a little crude, because it just prints whatever the user inputs. Try running the code and inputting some nonsense, and see what your program does with it.

You should be careful not to trust the input provided by the user, because you do not know who might be entering what into your program. There are ways to run user input as Python code, but these are best avoided.

Even putting aside malicious users, it is worth remembering that no matter how clearly you think you have specified what you want the user to do, the user will always find ways to surprise you. Have a look at this picture that I took while walking on the street near campus. I find it hard to believe that someone who is employed to lift up this inspection hatch could possibly misunderstand how to put it back, but here we go.

Checking user input

Let's start with some code that asks the user to input a string and prints it back to them.

```
your_input = input("Please enter some text: ")
print(your_input)
```

Say we only want to accept a string that starts with the letter 'p'. We can access the first character of `your_input` using `your_input[0]`, so we can test and print a message if this condition has not been met.

```
your_input = input("Please enter some text beginning with the letter 'p': ")
if your_input[0] != "p":
    print("Your text should begin with the letter 'p'")
print(your_input)
```

Try running this code and putting in different input values. If you put "hello", you should get the error message, and if you put "penguin" you should not see the error text. What happens if you put "Peter"?

If we want to accept strings that start with both 'p' and 'P', we can do this. We can convert a string `s` to lower case using `s.lower()`, so if we test `your_input[0].lower() == "p"` that will match either 'p' or 'P'.

```
your_input = input("Please enter some text beginning with the letter 'p': ")
if your_input[0].lower() != "p": # checks for 'p' or 'P'
    print("Your text should begin with the letter 'p'")
print(your_input)
```

Since we are testing `your_input`, a problem will occur if the test cannot run on whatever value `your_input` has taken. For example, if we were testing a number is in a certain range, we would have problems if what was entered was not a number. Here a problem happens if the user presses enter without typing any text, since if `your_input=""` then it does not have a first character to test using `your_input[0]`. If you run the code above and press enter without typing anything, you should see an `IndexError`, which means you have tried to access the character at position `0` and there is nothing there.

We can deal with this error by putting our code in a `try`/`except` block.

```
try:
    your_input = input("Please enter some text beginning with the letter 'p':
    if your_input[0].lower() != "p": # checks for 'p' or 'P'
        print("Your text should begin with the letter 'p'")
except IndexError:
    print("No text entered.")
print(your_input)
```

Finally, we might consider what happens if the user has done this wrong. They would have to run the program again to have another go. Instead, we can put our code in a loop that keeps going until a valid input is provided. We do this with a `while` loop. First we need a variable to act as a flag to tell us whether a string beginning with `p` has been provided, here we use `ask_for_input`. We initially set this to `"yes"`, only changing it to `"no"` (and so ending the loop) if a string beginning with 'p' is provided.

```
ask_for_input = "yes"
while ask_for_input == "yes":
    try:
        your_input = input("Please enter some text beginning with the letter '
        if your_input[0].lower() == "p": # checks for 'p' or 'P'
            ask_for_input = "no" # stop looping
        else:
            print("Your text should begin with the letter 'p'")
    except IndexError:
        print("No text entered.")
print(your_input)
```

A note on inputting numbers

What gets stored from `input()` is a string. If you would like to use this as a number, you must convert it.

Here we convert the input into an integer using `int()` and test whether it is greater than 5.

```
a = input("Think of a number: ")
if int(a)>5:
    print("It's a big 'un.")
```

Similarly, if your input is a decimal number, you convert a string `b` to a float (decimal) number using `float(b)`.