Section 2: Welcome to Python

Welcome to Section 2!

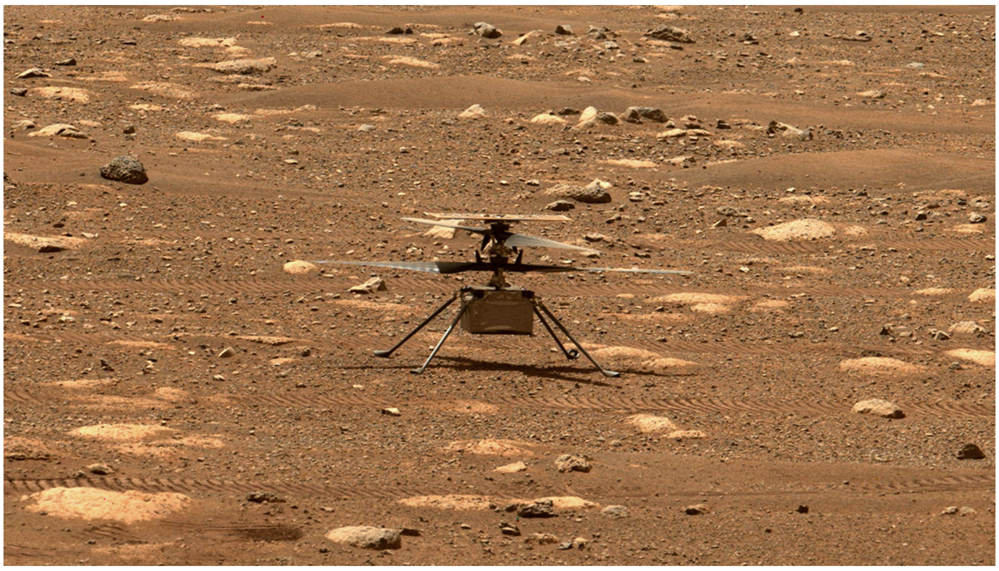
Heads up! You don't need to work through these problems till your section this week.

This week, section will focus on getting you comfortable with writing programs in Python. This week, we've covered using Variables and Expression, and the two problems in this section will help you apply those in different and exciting ways. By the end of section, you should be feel more comfortable with these ideas:

1. How to define a variable, how to use its value, and how to modify it
2. How to combine expressions with control flow tools like `if` statements and `while` loops.

Mars Weight

Last Monday, NASA made history with the first controlled flight on another planet. Its latest Mars Rover, *Perseverance,* has onboard a 50cm high helicopter called *Ingenuity.* On Sunday, *Ingenuity* made its third flight, during which it flew faster and further than it had on any of its test flights on Earth. Interestingly, *Ingenuity* uses Python for some of its flight modeling software!



Ingenuity on the surface of Mars (source: NASA)

When programming Ingenuity, one of the things that NASA engineers need to account for is the fact that due to the weaker gravity on Mars, an Earthling's weight on Mars is 37.8% of their weight on Earth. Write a Python program that prompts an Earthling to enter their weight on Earth and prints their calculated weight on Mars.

Sample Run

```
$ python marsweight.py
Enter a weight on Earth: 120
The equivalent on Mars: 45.36
```

8-ball

The idea behind an 8-ball is very simple. You ask it a yes or no question, and it tells you the answer. The catch is that the answer it chooses is randomly selected from a set of prefabricated responses. Here's what a physical 8-ball looks like:

Your job is to write a program that continuously prompts the user for a yes or no question, and then randomly selects from a set of canned answers. The classic 8-ball responses were:

1. As I see it, yes.
2. Ask again later.
3. Better not to tell you now.
4. Cannot predict now.
5. Concentrate and ask again.
6. Don't count on it.
7. It is certain.
8. It is decidedly so.
9. Most likely.
10. My reply is no.
11. My sources say no.
12. Outlook not so good.
13. Outlook good.

14. Reply hazy, try again.
15. Signs point to yes.
16. Very doubtful.
17. Without a doubt.
18. Yes.
19. Yes - definitely.
20. You may rely on it.

You **do not** need to have so many responses. Instead, choose at least 5 (or make up your own!).

Sample Run

Here's a demo of our program running, with some added Code in Place flair (use input is in `italics`):

```
$ python 8ball.py
Ask a yes or no question: Is Karel married?
Not a chance.

Ask a yes or no question: Is my real name Chris?
Only Karel knows.

Ask a yes or no question: 8-ball, are you using random numbers?
Without a doubt.
```

Milestones

Milestone 1: For the basic version of a program, have it answer *only one question.*

Milestone 2: If you have extra time, see if you can repeatedly answer questions until the user enters no question (i.e. they press enter without pressing anything)

Useful Syntax

Run PYTHON

```python
import random

random_number = random.randint(1, 42)
print(random_number)
```

`random.randint(A, B)` is a function that gives you back a random number in the range `A` to `B`, inclusive of `A` and `B`. `A` and `B` could be literal numbers, variables, or constants. You can choose any name for the variable that stores the result (in this case, `random_number`)

```python
# Try changing x and y
x = 42
y = 42

if x == y:
    print("x and y are equal!")
```

Python has an `if` statement, just like in Karel! This `if` statement passes if the value of `x` is the same as the value of `y`. `x` and `y` can be literal numbers, strings, variables, or constants.

Collaborative Workspace

This workspace slide does not have a description.