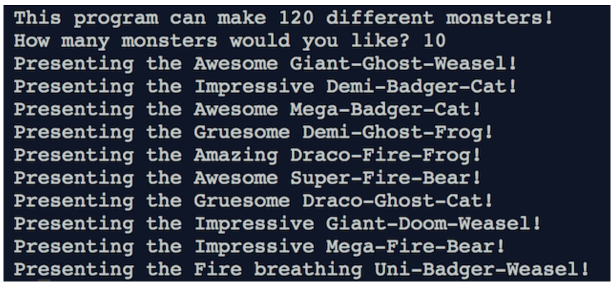
Monster mashup

Introduction

In this project — which is for people who've already completed the Beginner Python Sushi Cards (http://kata.coderdojo.com/wiki/Beginner_Python), or otherwise learned the basics of Python — you'll be making a random monster generator. It'll create a list of original monster names from parts you put into it! Who knows, maybe one will inspire your next Hallowe'en costume!

What you will make

A monster generator that will create such original creatures as the fire breathing mega-doom-cat, or the fearsome greater-draco-weasel.



```
This program can make 120 different monsters!
How many monsters would you like? 10
Presenting the Awesome Giant-Ghost-Weasel!
Presenting the Impressive Demi-Badger-Cat!
Presenting the Awesome Mega-Badger-Cat!
Presenting the Gruesome Demi-Ghost-Frog!
Presenting the Amazing Draco-Fire-Frog!
Presenting the Awesome Super-Fire-Bear!
Presenting the Gruesome Draco-Ghost-Cat!
Presenting the Impressive Giant-Doom-Weasel!
Presenting the Impressive Mega-Fire-Bear!
Presenting the Fire breathing Uni-Badger-Weasel!
```

What you need to remember

Here's a quick reminder of what you'll need to know before starting this project:

Sequence of code

Python reads your code from top to bottom. You can't use anything earlier in the program than you created it. So, for example, you can't use the value of a **variable** before you've actually created that variable and put the value in it.

Variables

Variables are the labels used to store values, such as text or numbers, so you can use them later in the program.

```
my_variable = 'Some text I want to remember'
another_variable = 42
```

Comments

Anything that follows a hash (#) character is a **comment** and will be ignored by Python until the end of the line. You can use comments to leave yourself, or other programmers, notes. This can be really helpful when you open up a program after a few weeks, months, or even years of not looking at it!

```
# This is a note on its own line
my_variable = 'Some text I want to remember' # This note could
remind me why the text is important
```

Indents are important

Python uses indenting (spaces in from the side) of your code to understand when things are inside other things, for example the code inside a loop or a conditional statement.

```
# no indent
  # one indent
    # two indents
```

Printing

You can use the `print()` function to display messages to your user.

```
print('My very important message')
```

Getting user input

You get user input using the `input()` function. Don't forget to store it in a variable!

```
user_name = input('What is your name?')
```

While loops

You can use a `while` **loop** to do the same thing over and over, as long as its condition is **true**.

```
my_number = 10

while my_number < 20:
  print(my_number)
  my_number = my_number + 1
```

What you will need

Software

- Python 3, either on your computer or through a site like repl.it (http://repl.it)

Monster parts

The monsters you're going to generate have three parts to their name: a beginning, a middle, and an end. Here's a few suggestions for each, but make up some of your own too!

Beginnings

- Giant
- Mega
- Lesser

Middles

- Draco
- Ice
- Uni

Ends

- Cat
- Ghost

- Dragon

- To start your program, you'll need three **lists** in Python, one for beginnings, one for middles, and one for ends.

Python lists

Python has a type of variable called a list. It's used when you have a group of items that you want to keep together in one variable. They're usually all related somehow, like the songs released by a band, or the players on a sports team.

Empty lists

This list has nothing in it yet — you can add items to it later:

```
my_list = []
```

Lists with things in them

This list has some numbers in it. Notice the commas (`,`) between them. These tell Python where each item in the list ends.

```
my_list = [1, 2, 3]
```

This list has some text in it. Again, notice the commas.

```
my_list = ['cat',
           'dog',
           'rabbit']
```

I need a hint

You'll need to have three lists in your code, like this:

```
name_start = ['Giant',
              'Mega',
              'Lesser'
              ]

name_middle = ['Draco',
               'Ice',
               'Uni'
               ]

name_end = ['Cat',
            'Ghost',
            'Dragon'
            ]
```

Ordering monsters

How many monsters can you make?

A quick bit of maths can tell you how many monsters you can make with the parts of names you've got.

- Have Python `print` out the number of monsters your program can create! All you need to do is multiply (`*`) the number of parts in each list by one another.

Getting the length of a list

To get a count of how many items are in a list, you can use `len()` like this:

```
my_list = [1, 2, 3]
length = len(my_list)
```

I need a hint

This code will work, but you may need to change the variable names to match your lists!

```
print('This program can make
'+str(len(name_start)*len(name_middle)*len(name_end))+'
different monsters!')
```

Placing an order

Next, ask the user for the number of monsters they'd like, and store that number in a variable.

I need a hint

This code will do it.

```
requested_monsters = input('How many monsters would you
like?')
```

Making monsters — picking pieces

You will need to make a few monsters to fill your user's order. The easiest way to do that is with a function, because it will let you re-use the same piece of code over and over again.

- Try creating a function that joins together the first items in each list.

Functions

Python functions are like pre-written sets of instructions which you can use by just **calling** the function's name. A similar example for a human would be "Make a cup of tea". You don't need to be told to put a teabag into a pot, then boil and add water, etc. You already know you'll need to do that to make tea. You can teach Python in the same way. Functions are created using `def`, which is short for 'define'. They usually **return** something, like that cup of tea!

```
def my_function():
  first_number = 4
  second_number = 6
  return first_number + second_number
```

You call a function — which tells it to run the code inside it — by using its name, like this:

```
my_function()
```

You can put the result of the function into a variable too:

```
my_number = my_function()
```

Getting things from lists

You can get the item at any position on the list by using its index — the number that marks its position. Note that the count of the position starts from 0 and not 1.

```
  my_list = ['cat', 'dog', 'rabbit']
  print(my_list[0]) # this will print 'cat'
```

I need a hint

A function that joins together the first items on each list would look like this, though you might need to change some variable names to match your lists.

```
def make_monster():
    start = name_start[0]
    middle = name_middle[0]
    end = name_end[0]

    name = start+'-'+middle+'-'+end

    return name
```

And you can test it like this:

```
print(make_monster())
```

Making monsters — at random!

Ok, your program makes the one monster. It's always the same. Time to make more of them! You're going to need to pick bits from each list at random. Luckily, this is pretty easy in Python: you just need to add `random`!

- Update your monster-making function to choose the pieces of the name at random.

Choosing from a list at random

You'll need to `import` some extra code so you can use the `random` functions. `random` is built into Python, so it'll always be available to import.

```
import random  # this line goes at the top of your file

my_list = ['cat', 'dog', 'rabbit']
random_animal = random.choice(my_list)
```

I need a hint

Add this at the top of the file:

```
import random  # this line goes at the top of your file
```

Update the function so it looks like this:

```
def make_monster():
    start = random.choice(name_start)
    middle = random.choice(name_middle)
    end = random.choice(name_end)

    name = start+'-'+middle+'-'+end

    return name
```

And you can test it like this:

```
print(make_monster())
print(make_monster())
print(make_monster())
```

The names you get should be different!

Making enough monsters

Your user ordered a certain number of monsters from you earlier. You need to deliver!

- Create a list of monsters, and then use a `while` loop to make them until you have enough.

Adding things to lists

You can add an item to a list using the `append` function.

```
my_list = ['cat', 'dog']
my_list.append('rabbit')
```

I need a hint

The code could look something like this:

```
while len(monsters) < int(requested_monsters):
  monsters.append(make_monster())
```

Note that you need to convert `requested_monsters` using `int()` before you can compare it to the result of `len()`.

Delivering your monsters

Now that you have enough monsters, it's time to introduce them to the user by printing them out! You can use a `for` loop to do something once for each item in a list, for example your list of monsters.

- Using a `for` loop and the list of monsters you created, `print` out the name of every monster.

For loops

`for` loops let you repeat a piece of code for every item in a list, like this:

```
my_list = ['cat', 'dog', 'rabbit']

for animal in my_list:
  print('I like to pet my '+animal)
```

I need a hint

The code could look something like this:

```
for monster in monsters:
    print('Presenting the '+monster+'!')
```

Deliver with style!

Give your monsters an impressive introduction!

- Create a list of introductions, and pick one at `random` for each monster as you're presenting them.

I need a hint

The code could look something like this:

```
introductions = ['Awesome',
                 'Amazing',
                 'Terrifying',
                 'Fire breathing'
                 ]

for monster in monsters:
    print('Presenting the '+random.choice(introductions)+'
'+monster+'!')
```