Functions with parameters

Why is this important?

So far we've been writing functions that don't take any arguments.

Most times you're going to pass some information into a function and, let the function do something with that information.

This is important because this building block is essential to writing more complex programs. You can write some functions that do a variety of things based on the information that you pass into it.

What are we going to do?

We're going to modify our `rock_paper_scissors.py` program to make it more modular; so that we can reuse the function.

We're also going to store our newly created function in another file and import it into our program so that we can simplify our program even further.

Steps

1. Let's take a look at what the code in our original `rock_paper_scissors.py` program is doing.

```
$ python rock_paper_scissors.py
Scissor (0), rock (1), paper (2): 1
The computer is rock. You are rock. It is a draw
```

- We're getting the user's input, then we're doing some processing to determine the winner, and then we're printing out the results.
- We can already see a bit of a function here. We can see that we're doing some processing to determine the winner. Let's create a function for that.

2. Let's first create a function that will just return the translated value.

- under the import create a function name `translate_number_to_choice` with one parameter `number` and print out the number. Your code should look like below.

```python
import random

def translate_number_to_choice(number):
    print(number)

# rest of code removed for brevity
```

- So what this does is just print out the number. Once you can call this in our code like so: `translate_number_to_choice(1)`. This will print out `1`.
- but we see some parts of our code that is repeats themselves. Let's use our function to shorten our code and make it a bit more modular. The code that is repeating is:

```python
# other code removed for brevity
user_result = ""
if user_input == 0:
    user_result = "scissor"
elif user_input == 1:
    user_result = "rock"
elif user_input == 2:
    user_result = "paper"

if computer_input == 0:
    computer_result = "scissor"
elif computer_input == 1:
    computer_result = "rock"
elif computer_input == 2:
    computer_result = "paper"
# rest of code removed for brevity
```

- let's take this code and put it in our function and return a value from our function (the translated value.).

```python
import random

def translate_number_to_choice(number):
    choice = ""
    if number == 0:
        choice = "scissor"
    elif number == 1:
        choice = "rock"
    elif number == 2:
        choice = "paper"
    return choice

user_input =  int(input("Scissor (0), rock (1), paper (2): "))
computer_input = random.randint(0, 2)

user_result = translate_number_to_choice(user_input)

computer_result = translate_number_to_choice(computer_input)

# rest of code removed for brevity
```

- you can see here that we've removed lots of code a replaced with it a call to our function `translate_number_to_choice`. We're passing in the user's input and the computer's input and translating the value and using `return choice` to return the translated value from the function. This will be stored in the variables `user_result` and `computer_result`.

3. Let's create a function to return the result of the game.

- Let's create a function called `get_game_result` that takes two arguments, `user_input`, and `computer_input`. Both inputs are going to be numbers 0, 1, or 2.
- Let's use the existing code that we created when we were learning about if statements. Under the `translate_number_to_choice` function, create a function called `get_game_result` that takes two arguments, `user_input` and `computer_input`.
    - in this function, we're going to return "tie", "user wins", and "computer wins" based on the arguments that we pass to the function.

```python
import random

def translate_number_to_choice(number):
    choice = ""
    if number == 0:
        choice = "scissor"
    elif number == 1:
        choice = "rock"
    elif number == 2:
        choice = "paper"
    return choice

def get_game_result(user_input, computer_input):
    pass


user_input =  int(input("Scissor (0), rock (1), paper (2): "))
computer_input = random.randint(0, 2)

user_result = translate_number_to_choice(user_input)
computer_result = translate_number_to_choice(computer_input)

# tie
if user_result == computer_result:
    print(F"The computer is {computer_result}. You are {user_result}. It 
is a draw")
# user wins
elif ((user_result == "paper" and computer_result == "rock") or
      (user_result == "rock" and computer_result == "scissor") or
      (user_result == "scissor" and computer_result == "paper")):
    print(F"The computer is {computer_result}. You are {user_result}. You 
win")
# computer wins
elif ((computer_result == "paper" and user_result  == "rock") or
      (computer_result == "rock" and user_result  == "scissor") or
      (computer_result == "scissor" and user_result  == "paper")):
    print(F"The computer is {computer_result}. You are {user_result}. You 
lose")
```

- now let's take a look at the code that we created when we were learning about if statements.
  - we're going to move all of this code into our `get_game_result` function and return the proper values based on the condition.

```
# other code removed for brevity

def get_game_result(user_input, computer_input):
    user_result = translate_number_to_choice(user_input)
    computer_result = translate_number_to_choice(computer_input)

    # tie
    if user_result == computer_result:
        print(F"The computer is {computer_result}. You are {user_result}.
It is a draw")
        return "tie"
    # user wins
    elif ((user_result == "paper" and computer_result == "rock") or
        (user_result == "rock" and computer_result == "scissor") or
        (user_result == "scissor" and computer_result == "paper")):
        print(F"The computer is {computer_result}. You are {user_result}.
You win")
        return "user wins"
    # computer wins
    elif ((computer_result == "paper" and user_result  == "rock") or
        (computer_result == "rock" and user_result  == "scissor") or
        (computer_result == "scissor" and user_result  == "paper")):
        print(F"The computer is {computer_result}. You are {user_result}.
You lose")
        return "computer wins"
```

- Let's take a look at what this code is doing now.
  - in the first two lines of the function, we're calling our `translate_number_to_choice` function and passing in the `user_input` and `computer_input` arguments. This will return the translated value and store it in the `user_result` and `computer_result` variables.
    - You can see here that we can call functions in other functions here, this is super common!
  - Let's take a look at the first if statement. We're checking to see if the `user_result` is equal to the `computer_result`. If it is, we're printing out the results and returning the string "tie" from the function. Remember that if you hit a `return` statement in a function, the function will stop executing and return the value that you passed to the `return` statement.
  - Next, we are checking to see if the user wins. If the user wins, we're printing out the results and return the string "user wins" from the function.
    - We're checking to see if the user wins with all of the possible combinations of the user winning.
      - If it's true then it's

        - Note that the elif statements are checking if the user wins and the condition if the computer wins uses some of the same code. We can refactor this code to make it a bit more readable (let's create another function!).
    - The computer wins code is doing the same thing as the user wins code, but it's checking to see if the computer wins.
        - Again we're going to "refactor" the code to make it a bit more readable.
- use this function at the end of the file under the `user_input` and `computer_input`.

```
# other code removed for brevity
user_input =  int(input("Scissor (0), rock (1), paper (2): "))
computer_input = random.randint(0, 2)

get_game_result(user_input, computer_input)
```

- Note that if you

4. Let's make one more function to refactor or code to check if the user or computer is the winner.

Having code duplication is something you want to avoid. It makes your code harder to read and maintain. If you need to make a change to the code, you need to make the change in multiple places. This is a recipe for disaster.

- Let's create a function before `get_game_result` called `is_winner` that takes two arguments, `player_choice` and `opponent_choice`. This is going to return a boolean that is `True` if the `player_choice` wins and `False` if it doesn't.
    - This is a common technique to use when you have complicated conditions like we have here.
- Let's take a look at the code that we're going to use in our `is_winner` function.

```
# other code removed for brevity

# checks if the player choice wins
def is_winner(player_choice, opponent_choice):
    return ((player_choice == "paper" and opponent_choice == "rock") or
        (player_choice == "rock" and opponent_choice == "scissor") or
        (player_choice == "scissor" and opponent_choice == "paper"))

# rest code removed for brevity
```

- Note that you can see that it's returning a boolean, but it depends on the order that we pass our arguments to the function. If we pass in the user's choice first, then it's going to return `True` if the user wins. If we pass in the computer's choice first, then it's going to return `True` if the computer wins.
- Let's use this in our function to see how our code looks now.
    - we're also going to put the code that is executed under `if __name__ == "__main__"`

```python
import random

def translate_number_to_choice(number):
    choice = ""
    if number == 0:
        choice = "scissor"
    elif number == 1:
        choice = "rock"
    elif number == 2:
        choice = "paper"
    return choice

# checks if the player choice wins
def is_winner(player_choice, opponent_choice):
    return ((player_choice == "paper" and opponent_choice == "rock") or
        (player_choice == "rock" and opponent_choice == "scissor") or
        (player_choice == "scissor" and opponent_choice == "paper"))

def get_game_result(user_input, computer_input):
    user_result = translate_number_to_choice(user_input)
    computer_result = translate_number_to_choice(computer_input)

    # tie
    if user_result == computer_result:
        print(F"The computer is {computer_result}. You are {user_result}. It is a draw")
        return "tie"
    # user wins
    elif is_winner(user_result, computer_result):
        print(F"The computer is {computer_result}. You are {user_result}. You win")
        return "user wins"
    # computer wins
    elif is_winner(computer_result, user_result):
        print(F"The computer is {computer_result}. You are {user_result}. You lose")
        return "computer wins"

if __name__ == "__main__":
    user_input =  int(input("Scissor (0), rock (1), paper (2): "))
    computer_input = random.randint(0, 2)

    get_game_result(user_input, computer_input)
```

- You can see here that our code here is much easier to read and we could easily make changes to the code if we need to!

5. Let's modularize this even further by storing our functions in another file and importing them into our program.

- Let's create a file called `rps_functions.py` and move our functions into this file.

- Copy the functions `translate_number_to_choice`, `is_winner` and `get_game_result` into our new file. Our `rps_functions.py` file should look like below.

```python
def translate_number_to_choice(number):
    choice = ""
    if number == 0:
        choice = "scissor"
    elif number == 1:
        choice = "rock"
    elif number == 2:
        choice = "paper"
    return choice

# checks if the player choice wins
def is_winner(player_choice, opponent_choice):
    return ((player_choice == "paper" and opponent_choice == "rock") or
        (player_choice == "rock" and opponent_choice == "scissor") or
        (player_choice == "scissor" and opponent_choice == "paper"))

def get_game_result(user_input, computer_input):
    user_result = translate_number_to_choice(user_input)
    computer_result = translate_number_to_choice(computer_input)

    # tie
    if user_result == computer_result:
        print(F"The computer is {computer_result}. You are {user_result}.
It is a draw")
        return "tie"
    # user wins
    elif is_winner(user_result, computer_result):
        print(F"The computer is {computer_result}. You are {user_result}.
You win")
        return "user wins"
    # computer wins
    elif is_winner(computer_result, user_result):
        print(F"The computer is {computer_result}. You are {user_result}.
You lose")
        return "computer wins"
```

- to be able to import files from another folder we need to create a file called `__init__.py` in the folder that we want to import from. In this case, we need to create a file called `__init__.py` in the `functions_with_parameters_intro` folder (which is the folder we're in)
- Now in our `rock_paper_scissors.py` file, we can import our functions from our `rps_functions.py` file and delete our functions in this file.
  - The line `from rps_functions import get_game_result` is going to import the `get_game_result` function from the `rps_functions.py` file. Note that this `from ... import ...` works because we have our `__init__.py` file in the `functions_with_parameters_intro` folder.

```python
import random

from rps_functions import get_game_result

if __name__ == "__main__":
    user_input =  int(input("Scissor (0), rock (1), paper (2): "))
    computer_input = random.randint(0, 2)

    get_game_result(user_input, computer_input)
```

- Note you can see that we've now removed all of our functions from our `rock_paper_scissors.py` file and we're just importing the `get_game_result` function from our `rps_functions.py` file, and we essentially have a very small program that is just calling our function.
- Let's run our program to make sure that it works.

```
$ python rock_paper_scissors.py
Scissor (0), rock (1), paper (2): 1
The computer is rock. You are rock. It is a draw
```

- You can see here that our program is working as expected except it's a lot more organized and easier to read!