Advanced Python usage

Unpacking iterables and dictionaries

Variable number of arguments

- There are situations where a function can take a variable number of arguments.
- Or situations where all arguments to a specific function call are already available in a dictionary or a list
- Python has a mechanism for "unpacking" these arguments
- Syntax is `*args` (for an iterable) and `**kwargs` (for a dictionary).

Example : iterable

```
def func(*args):
    print(args)

func("yes", 2)    # prints ["yes", 2]
```

Example : dictionary

```
def func(**kwargs):
    print(kwargs)

func(example="yes", value=2)    # prints {"example": "yes", "value": 2}
```

Works both ways

```
my_list = ["yes", 2]
my_dict = {"example": "yes", "value": 2}

func(*my_list)      # equivalent to func("yes", 2)
func(**my_dict)     # equivalent to func(example="yes", value=2)
```

Use it wisely

- Extremely powerful way of dealing with variable / multiple arguments

```python
kwargs = {}
if difficulty in ("easy", "medium", "hard"):
    kwargs["difficulty"] = difficulty

if category in get_categories():
    kwargs["category"] = category

if number.isdigit():
    kwargs["number"] = int(number)

get_questions(**kwargs)
```