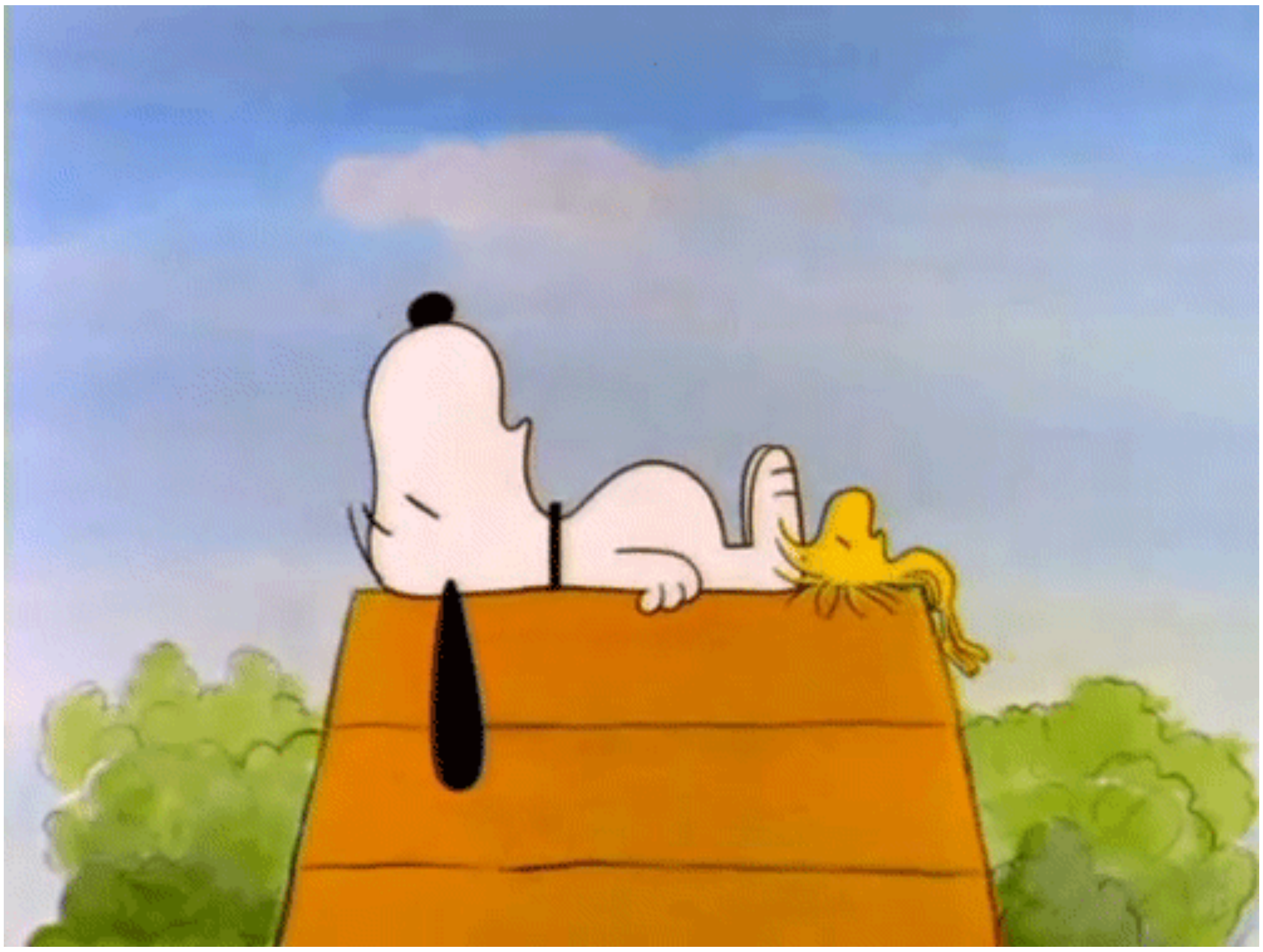
Good morning



How are you?

How did the sample test work out during the weekend?

Read the tasks

It is generally a good idea to read the tasks carefully.

Usually we would like you to do quite much *exactly* what is specified there.

Hint, verbs like `return` indicate what you have to do.

Stacks and queues? What was that?

Unrelated - But computational responsibility

Think about your use of your computers:

> In 2018, online video viewing generated more than 300 MtCO2, i.e. as much greenhouse gas as Spain emits: 1% of global emissions.
>
> Pornographic videos make up 27% of all online video traffic in the world. Taken alone, in 2018 they generated more than 80 MtCO2, i.e. as much as all France’s households: close to 0.2% of global emissions.
>
> The greenhouse gas emissions of VoD (video on demand) ser- vices (e.g. Netflix and Amazon Prime) are equivalent to those of a country like Chile (more than 100 MtCO2eq/year, i.e. close to 0.3% of global emissions), the country hosting the COP25 in 2019.
>
> https://theshiftproject.org/wp-content/uploads/2019/07/Excutive-Summary_EN_The-unsustainable-use-of-online-video.pdf (https://theshiftproject.org/wp-content/uploads/2019/07/Excutive-Summary_EN_The-unsustainable-use-of-online-video.pdf)

Functions, parameters and arguments

- Parameters are variables that are a part of a functions description.

In [26]:

```
# Here are x and y the parameters of the function multiply_numbers.
def multiply_numbers(x, y):
    return x * y
```

- Arguments are the values, which are passed into a function.

In [ ]:

```
# Here are 4 and 5 the arguments which are passed into the function multiply_
multiply_numbers(4, 5)
```

Required arguments

- The arguments which are required by the functions description.
- These arguments has to be in positional order (when we don't use keywords).

In [38]:

```python
def parrot(some_string, number_repetitions):
    """This function will repeat some_string, number_repetitions times

    :param some_string: string
        The string to repeat
    :param number_repetitions: int
        The number of times the string should be repeated
    :returns: string
    """
    return str(some_string) * int(number_repetitions)
```

In [42]:

```python
# What will happen?
# parrot("blah ", 16)

# parrot("blah ")

# parrot(16, "blah ")
```

Keyword arguments

- When the function is called, one can use the parameters names to identify the arguments.
- The positional order does not matter when one uses keyword arguments.

In [ ]:

```python
multiply_numbers(x=10, y=5)
```

In [ ]:

```python
parrot(some_string="Blah ", number_repetitions=20)
```

In [ ]:

```python
parrot(some_string="Blah ", number_repetitions=20) == parrot(number_repetitic
```

Default arguments

- Arguments which will take a default value, if no value were provided in the function call
- The default value must be specied in the functions definition.

In [ ]:

```python
print(help(print))
```

In [50]:

```
def parrot(some_string, number_repetitions = 1):
    """This function will repeat some_string, number_repetitions times

    :param some_string: string
        The string to repeat
    :param number_repetitions: int
        The number of times the string should be repeated, with 1 as default
    :returns: string
    """
    return str(some_string) * int(number_repetitions)
```

In [ ]:

```
parrot("blah")

# parrot("blah", number_repetitions=10)
```

Variable-length arguments

- These arguments can be used when you don't know have many arguments there will be used in a function call
- These variable-lenght arguments are shown by the a *
- We can see this in print's documentation https://docs.python.org/3/library/functions.html (https://docs.python.org/3/library/functions.html)

In [53]:

```
def print_knock_off(*objects):
    for obj in objects:
        print(obj, end="  ")
```

In [ ]:

```
print_knock_off("This", "is", "a", "test")
```

In [ ]:

```
print("This", "is", "a", "test")
```