# Conditionals

### should_work

Create a function called "should_work" it should take one parameter: "tired" which is a boolean. if tired is True, should_work should return False. If tired is False, should_work should return True. (i.e. it returns the opposite)

### Version 1

In [31]:
```
def should_work(tired: bool) -> bool:
    if tired: # if tired is not in (False, 0, None)
        return False
    else:
        return True
```

### Version 2

In [5]:
```
def should_work(tired: bool) -> bool:
    return not tired

# return not tired: The not operator is used to negate the value of tired. If tired is True,
# not tired evaluates to False, and if tired is False, not tired evaluates to True.
# The return statement returns the opposite boolean value as the result of the function.
```

### is_even

Create a function called "is_even" that takes one parameter: "num" which is a number. It returns True if the number is even and False if the number is odd. It should raise an error if not given a numer

In [7]:
```
def is_even(num: (int, float)) -> bool:
    if isinstance(num,(int, float)):
        return num % 2 == 0
    else:
        raise Exception('No number given')

print(is_even(2))
```

```
True
```

### car_at_light

Create a function called "car_at_light" It should take one parameter: "light" which gives the color of a traffic light. If the color is "red", the function should return "stop". If the color is "green", the function should return "go". If the color is "yellow" the function should return "wait". If the color is anything else, the function should raise an exception.

In [9]:
```
def car_at_light(light:str):
    colors = ['red', 'green', 'yellow']
    if light in colors:
        if light == 'red':
            return 'STOP!'
        if light == 'green':
            return 'GO!'
        if light == 'yellow':
            return 'WAIT!'

    raise Exception('No valid light color given')
```

### is_cold

Create a function called "is_cold" which takes one parameter: "temperature" which is a number. If the temperature is above 15 degrees, it should return False, if it is equal or below 15 degrees, it should return True, if the temperature is not a number, it should raise an exception that says "temperature must be a number" (hint: you need to raise an exception with EXACTLY that message)

In [14]:
```
def is_cold(temperature: (int, float)):

    if not isinstance(temperature,(int, float)):
        raise Exception('temperature must be a number')

    return temperature <= 15
```

### wear_sweater

Create a function called "wear_sweater" which takes two parameter: "temperature", "friolera" The first is a number, the second a boolean (which might sometimes be a None). If the temperature is above 15, it should return False, if it is 15 or below, it should return True if friolera is True, False if friolera is False, and True if friolera is None. (In other words, if we don't know whether she is a friolera or not, we should act cautiously and tell her to wear a sweater) (hint: you should use is_cold from above!)

*The following set of cases are what we want to conditionalize*

- Temperature>15, doesn't matter what friolera is -> don't wear sweater, it's hot
- Temperature<15,friolera=True -> wear sweater
- Temperature<15,friolera=False -> don't need to wear sweater
- Temperature<15,friolera=None -> let's be safe and wear sweater

In [21]:
```
def wear_sweater(temperature: (int, float), friolera:bool):

    if not isinstance(temperature,(int, float)):
        raise Exception('temperature must be a number')

    if not is_cold(temperature):
        return "don't wear sweater, it's hot"
    elif is_cold(temperature) is True:
            if friolera is True:
                return "wear sweater"
            if friolera is False:
                return "don't need to wear sweater"
            if friolera is None:
                return "let's be safe and wear sweater"
```

### equality

Create a function called "equality" which takes three parameters: "x", "y", and "z" it should return the following: if x, y, and z are all equal --> "all are equal" if only x and y are equal --> "x and y are equal" if only z and x are equal --> "x and z are equal" if only y and z are equal --> "y and z are equal" if nothing is equal --> "nothing is equal"

In [27]:
```
def equality(x,y,z):
    if x == y == z:
        return 'All are equal'
    if x == y:
        return 'x and y are equal'
    if x == z:
        return 'x and z are equal'
    if y == z:
        return 'y and z are equal'
    #if x != y != z:
    return 'nothing is equal'

equality(3,3,5)
```

Out[27]: 'x and y are equal'

### driver_seat

Write a function for the following situation: Sofia can drive manual and automatic cars. Diego only knows how to drive automatic. Sofia prefers to drive long distances (> 10 km). Diego prefers to drive short distances.

The function should be called "driver_seat" and should take two parameters: "distance", "is_manual". The first should be a number, the second should be a boolean. The function should return a string, which is the name of the person who should drive, "Diego" or "Sofia".

*The following set of cases are what we want to conditionalize*

- if is_manual is True -> Sofia
- if is_manual is False, distance > 10 -> Sofia
- if is_manual is False, distance <= 10 -> Diego

In [29]:
```
def driver_seat(distance: (int, float), is_manual: bool):
    if not isinstance(is_manual, bool):
        raise ValueError("Invalid is_manual type")

    if not isinstance(distance, (int, float)):
        raise ValueError("Invalid distance type")
```

```
    if is_manual or distance > 10:
        return "Sofia"
    return "Diego"

print(driver_seat(12, False))
```

```
Sofia
```