Debugging strategies

In this notebook, we'll talk about what happens when you get an error message (it will happen often!) and some steps you can take to resolve them.

Run the code in the next cell.

In [1]:
```
x = 10

if x > 20
    print(f'{x} is greater than 20!')
```

```
  File "/var/folders/6p/3792ml551vv6d6c6yvwm5py00000gn/T/ipykernel_86133/90046043.py", lin
e 3
    if x > 20
             ^
SyntaxError: invalid syntax
```

The "traceback" message shows you a couple of useful things:

- What line the error is on: `line 3`
- The class of error: `SyntaxError` (v common)
- Exactly *where* the error occured -- see where the `^` symbol is pointing?

What's the problem?

Googling

If it's not immediately clear what's wrong -- if you're not even sure what a `SyntaxError` even is -- I might start by Googling the error messsage, the word "python" and maybe some keywords for what I was trying to do when I got the error. Something like `"SyntaxError: invalid syntax" python if statement`

Click through the first couple of links -- you'll become *very* familiar with StackOverflow -- and see if you spot the problem.

If you're still stuck, maybe it's time to ...

Read the docs

My next stop would be the Python documentation to find some examples of the thing I'm trying to do. Here's the page outlining how to write an `if` statement in Python. From there, I would copy the example code, run it, compare it line by line with my code and see what's different.

If I'm *still* stuck, I might see if there are other keywords to search on and take another run at Google.

Use `print()` liberally

The `print()` function can be a lifesaver -- it can show you *what* a value is before you try to do something to it, and whether it matches up with your expectations of what that value should be, and thereby give you a clue about why your script is failing. An example can help clarify this idea.

Scenario: Your newsroom is handing out longevity bonuses. (Congratulations!) Each employee's bonus will be the number of years they've been with the company, times 50.

So we're going to loop over our staff data, held in a list of dictionaries, and calculate each person's bonus.

In [2]:
```
staff = [
    {'name': 'Fran', 'years_of_service': 2, 'job': 'Reporter'},
    {'name': 'Graham', 'years_of_service': 7, 'job': 'Reporter'},
    {'name': 'Pat', 'years_of_service': 4, 'job': 'Web Producer'},
    {'name': 'John', 'years_of_service': '26', 'job': 'Managing Editor'},
    {'name': 'Sue', 'years_of_service': 33, 'job': 'Executive Editor'}
]

for person in staff:
    name = person['name']
    bonus = person['years_of_service'] * 50
    print(f'{name} is getting a bonus of {bonus}')
```

```
Fran is getting a bonus of 100
Graham is getting a bonus of 350
Pat is getting a bonus of 200
John is getting a bonus of 26262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626
Sue is getting a bonus of 1650
```

We didn't get an exception, but something is *clearly* wrong with John's bonus. What's going on?

Maybe you spot the error already. If not, we might Google something like "python multiply numbers repeating" -- which leads us to this StackOverflow answer. Is that what's going on here? Let's add a `print()` statement before we do the multiplication and use the `type()` function to check the value that we're pulling out of each dictionary.

In [3]:
```
for person in staff:
    name = person['name']
    bonus = person['years_of_service'] * 50
    print(name, type(person['years_of_service']))
    print(f'{name} is getting a bonus of {bonus}')
```

```
Fran <class 'int'>
Fran is getting a bonus of 100
Graham <class 'int'>
Graham is getting a bonus of 350
Pat <class 'int'>
Pat is getting a bonus of 200
John <class 'str'>
John is getting a bonus of 26262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626262626
Sue <class 'int'>
Sue is getting a bonus of 1650
```

Aha! John's value for `years_of_service` has been stored as a string, not an integer. Let's fix that by using the `int()` function to coerce the value to an integer.

In [4]:
```
for person in staff:
    name = person['name']
    bonus = int(person['years_of_service']) ** 2
    print(f'{name} is getting a bonus of {bonus}')
```

```
Fran is getting a bonus of 4
Graham is getting a bonus of 49
Pat is getting a bonus of 16
John is getting a bonus of 676
Sue is getting a bonus of 1089
```

Winner winner, chicken dinner.

Here are some more debugging exercises for you to work through. See if you can figure out what's wrong and fix them.

In [5]:
```
print(Hello, Sydney!)
```

```
  File "/var/folders/6p/3792ml551vv6d6c6yvwm5py00000gn/T/ipykernel_86133/2876108359.py", l
ine 1
    print(Hello, Sydney!)
                        ^
SyntaxError: invalid syntax
```

In [6]:
```
desk = {
    'wood': 'fir',
    'color': 'black',
    'height_in': 36,
    'width_in': 48,
    'length_in': 68
}

print(desk['drawer_count'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/var/folders/6p/3792ml551vv6d6c6yvwm5py00000gn/T/ipykernel_86133/1035002135.py in <module>
      7 }
      8
----> 9 print(desk['drawer_count'])

KeyError: 'drawer_count'
```

In [7]:
```
students = ['Kelly', 'Larry', 'José', 'Frank', 'Sarah', 'Sue']

for student in students:
    if student = 'Kelly':
    print('It's Kelly!')
    elif student == 'José':
        print("It's José!")
```

```
  File "/var/folders/6p/3792ml551vv6d6c6yvwm5py00000gn/T/ipykernel_86133/1961114220.py", l
ine 4
    if student = 'Kelly':
               ^
SyntaxError: invalid syntax
```

Further reading

- Python's tutorial on errors and exceptions
- Software Carpentry post on understanding Python errors
- How to read a traceback