09-trace-debug

September 27, 2020

0.1 Code tracing and debugging

Once you come up with an idea for a code, it is almost never the case that your first implementation will work. When this happens, you need to take a breath and start debugging.

Debugging is basically tracing your program in your head (often with the help of some pen and paper) for some particular input, and finding out where and why it fails. Obviously you should choose an input that makes your program fail, but something small enough so that you don't need to run too many steps until failing. Your input should be **as simple as possible and as complicated as necessary**.

If you happen to have a computer around, you can use it to help you trace the code by adding `print` statements to your code.

[1]:
```python
print("Hello world!")
```

```
Hello world!
```

[ ]:
```python

```

`print` is a python command used to print things to the console.

Whatever is in the parentheses is going to be printed. If the thing in parentheses is in quotes, it is going to be printed as is. If the thing in parentheses is a variable, python prints the value that is inside that variable.

[2]:
```python
x = 10
print(x)
```

```
10
```

Quotes and variables can be combined inside prints to make the messages more understandable. A comma indicates there is a space between the parts.

[3]:
```python
x = 42
print("The answer to the life, universe, and everything is", x, ".")
```

```
The answer to the life, universe, and everything is 42 .
```

Note that this does not alter the flow of your program. Python does nothing to whatever is printed, and this is only for your scrutinity.

[4]:
```python
def mystery(n):
    x = 0
    while (x != 1 and x != 4):
        x = 0
        print("while 1")
        print("n =", n)
        print("x =", x)
        while (n != 0):
            x = x + (n % 10) ** 2
            n = n // 10
            print("   while 2")
            print("   n =", n)
            print("   x =", x)
        n = x
    if x == 1:
        return True
    else:
        return False

mystery(42)
```

```
while 1
n = 42
x = 0
   while 2
   n = 4
   x = 4
   while 2
   n = 0
   x = 20
while 1
n = 20
x = 0
   while 2
   n = 2
   x = 0
   while 2
   n = 0
   x = 4
```

[4]: False