1 **Probe**

Add `print` statements. Use to:

- Check if a function is being called or not:
  ```python
def f(x, y):
    return x + 3*y
  ```
  ```python
  → def f(x, y):
    print("HELLO FROM f")
    return x + 3*y
  ```

- Check the value of a variable:
  ```python
  y = 15 / x
  ```
  ```python
  → print("x ":, x)
   y = 15 / x
  ```

- Check what happens at a conditional:
  ```python
  if x > 5:
    y = 10
  else:
    y = 3
  ```
  ```python
  → if x > 5:
    print("x > 5")
    y = 10
  else:
    print("x <= 5")
    y = 3
  ```

2 **Trace**

Use multiple probes to understand code. Use to:

- Figure out where a value comes from:
  ```python
def f(a):
  g(a * 3)
def g(b):
  for i in range(b):
    h(9-i)
def h(c):
  print(10/c)
  ```
  ```python
  (error if c is 0 in function h)
  → def f(a):
    print("a ":, a)
    g(a * 3)
def g(b):
  print("b ":, b)
  for i in range(b):
    print("i ":, i)
    h(9-i)
def h(c):
  print("c ":, c)
  print(10/c)
  ```

3 **Unpack**

Split up a complicated expression into multiple statements. Use this to:

- Isolate an error in a complex expression:
  ```python
  x = function( 
  (a + 3*b)/(c * d),
  b / a
  )
  ```
  ```python
  → top = a + 3*b
  bot = c * d
  fst = top / bot
  sec = b / a
  x = function(fst, sec)
  ```
  (ZeroDivisionError on line 1)
  (ZeroDivisionError on line 4, so a must be the problem)

4 **Toggle**

Turn a line of code into a comment. Use to:

- Disable (can later re-enable) optional code:
  ```python
def f(a, b):
  print("R : ", a/b)
  return a + b + a
  ```
  ```python
  ↔ def f(a, b):
    # print("R : ", a/b)
    return a + b + a
  ```

- Temporarily replace broken code with a dummy value:
  ```python
  x = (3*y + 4*z)/w
  ```
  ```python
  → #x = (3*y + 4*z)/w
   x = 9
  ```

5 **Bisect**

Comment out half of your code to find the half that works, and then half of the broken part, etc., until you isolate an error. Use this to:

- Find missing brackets or commas:
  ```python
  pairs = [
  [0, 1],
  [10, 11],
  [20, 21],
  [30, 31],
  ]
  ```
  ```python
  (syntax error at end of file)
  → pairs = [
    # [0, 1],
    # [10, 11],
    [20, 21],
    [30, 31],
    ]
  ```
  (works now, so error must be in the commented zone)

Note: To fit examples on this page, short and meaningless variable names have been used. DO NOT do this in your own code.