Interpreter - evaluating Numbers

General idea: process the stuff in the "data" (or "left" and "right", if a binary expression like x + y or x * y) attributes of the AST node, and return the result.

If unable to process the attached node data, then raise an error.

Number is the base case for all our recursive calls, so it doesn't call any other interpreter functions.

Uses type hinting, introduced in Python 3.5

```
def process_number(ast: ast_types.Number):
    try:
        return int(ast.data)
    except:
        raise SyntaxError('Unable to process "{}" as number'.format(ast.data))
```

Interpreter - evaluating binary (and unary) expressions

```
def process_mul_exp(ast):
    a = process_ast_exp(ast.left)
    b = process_ast_exp(ast.right)
    return a * b
```

```
def process_add_exp(ast):
    a = process_ast_exp(ast.left)
    b = process_ast_exp(ast.right)
    return a + b
```

```
def process_neg_exp(ast):
    result = process_ast_exp(ast.data)
    return -result
```

Unary expression

Binary expressions

```
def process_sub_exp(ast):
    a = process_ast_exp(ast.left)
    b = process_ast_exp(ast.right)
    return a - b
```

```
def process_div_exp(ast):
    a = process_ast_exp(ast.left)
    b = process_ast_exp(ast.right)
    if b == 0:
        raise ZeroDivisionError(f"Zero division error: cannot divide {a} by 0")
    return int(a / b)
```