


Interpretation overview

- As discussed, we will be using a tree-walk interpreter
- This means that we will start at the top-level AST node returned by the parser, and recursively process each of its children
- This could be done with a single function that conditions on the type of the AST node, but is better served by having a function for each type of AST node instead
- In addition, we will have a general "process AST node" function that calls the relevant function for that type
- Note: it may be better to do this with class methods, but we haven't gone too in-depth on how classes work, so we will only be using functions

Interpreter - call the relevant function

```
def process_ast_exp(ast: ast_types.ASTNode):  
    result = None  
    match type(ast):  Uses pattern matching, introduced in Python 3.10  
        case ast_types.Number:  
            result = process_number(ast)  
        case ast_types.NegExp:  
            result = process_neg_exp(ast)  
        case ast_types.AddExp:  
            result = process_add_exp(ast)  
        case ast_types.SubExp:  
            result = process_sub_exp(ast)  
        case ast_types.MulExp:  
            result = process_mul_exp(ast)  
        case ast_types.DivExp:  
            result = process_div_exp(ast)  
    if result is None:  
        raise SyntaxError("Unable to process AST of type {}".format(type(ast)))  
    return result
```