Compyler Syntax

Simple example

```
import compyler as cp
import compyler.stat as cstat
import numpy as np

# Create a FHE program object
# If the parameter scheme or library is missing,
# compiler automatically determines proper scheme or target l

program = cp.FheProgram(scheme="CKKS")

# A function to be a FHE circuit
def inverse(a):
    b = -a + 2
    c = a + 0
    for i in range(5):
        c = c * (-c + 2)
        b = b * (-c + 2)

    # After 5 iterations b should be more or less
    # b ≈ 1 / a
    return b

# Add secret variables .add_secret("a", "b", ...)
a_sec = program.add_secret("a")

# Compile the given function into a circuit
# Internally, this call generate SIR (Scheme IR) and apply se
program.compile(inverse(a_sec), backend="liberate")

# Show parameters
print(program.get_params())
# (32768, [60, 40, 40, 40, 40, 40, 40], [60, 60])
```

```
# Create an example input data
plain_a = np.linspace(2 ** (-3), 1, program.slots)

# Execute the circuit
# Internally, this call maps SIR to specific library codes
# and execute the codes with the given input.
result_liberate = program.compute({"a" : plain_a})

# Execute the reference (including approximation)
plain_result = program.plain_compute({"a": plain_a})

# Calculate relative errors
error = np.abs((result - plain_result) / plain_result)
print(np.mean(error))
```

Interface update complete https://github.com/Desilo/compyler