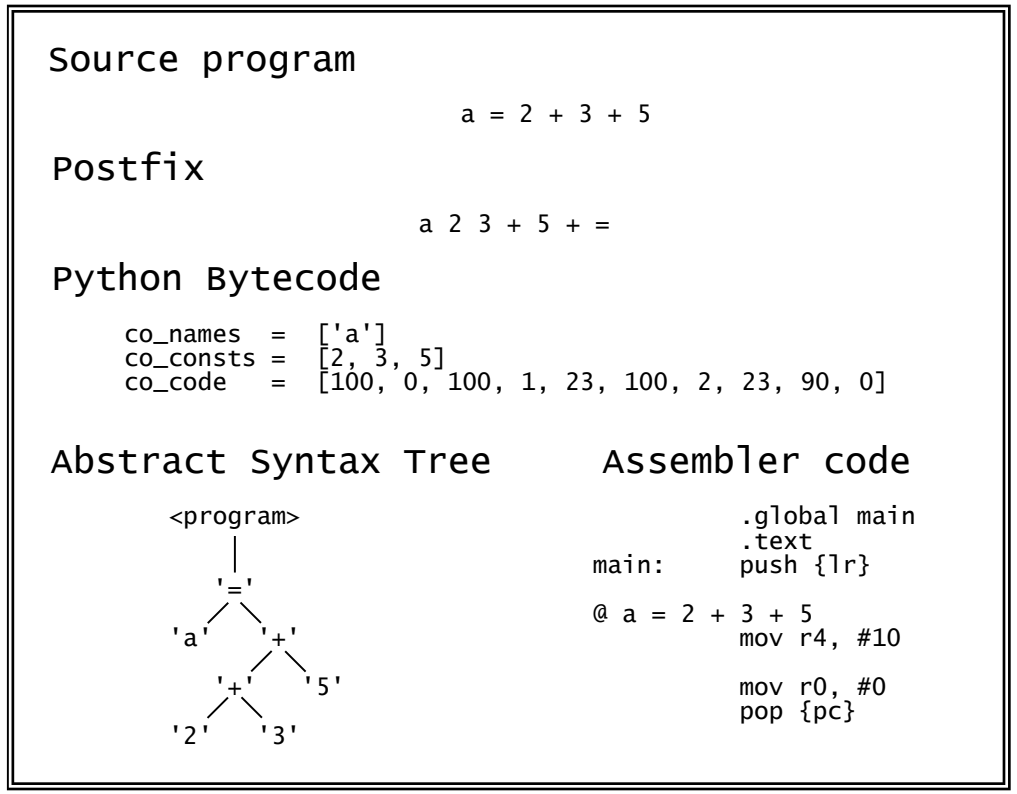
Slides
Writing Interpreters and Compilers for the Raspberry Pi Using Python

Second Edition



Anthony J. Dos Reis

Slides Writing Interpreters and Compilers for the Raspberry Pi Using Python Second Edition

1 Basic Concepts

sample.py

```
print('hello')
```

sample.c

```
#include <stdio.h>
int main(void)
{
   printf("hello\n");
   return 0;
}
```

Figure 1.1

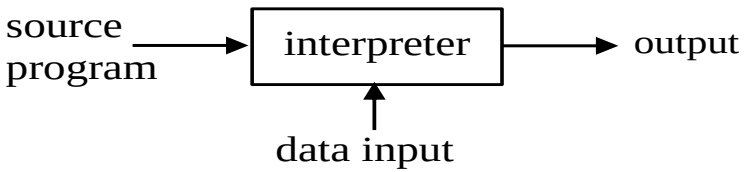


Figure 1.2

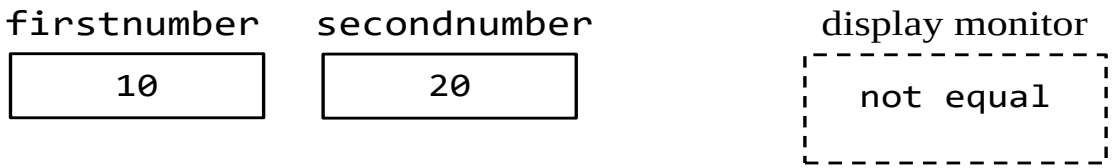
What happens when a pure interpreter interprets

1. Tokenize: break up into meaningful units
2. Parse: determine structure and identify components
3. Execute

Let's interpret:

```
1 firstnumber = 10
2 secondnumber = 20
3 if firstnumber == secondnumber:
4    print('equal')
5    print('good')
6 else:
7    print('not equal')
```

Figure 1.3



Newline is a token in Python

```
x = 1 + 2 + 3
```


```
x = 1          // C++ statement
    + 2
    + 3;
```

Indents and dedents are tokens in Python

```
if x == y: ←——— indent marks the start of the code block
   print('1')
   print('2') ←———  dedent marks the end of the code block
print('done')

if x == y: ←——— indent marks the start of the code block
   print('1') ←———  dedent marks the end of the code block
print('2')
print('done')
```

```
1 firstnumber = 10
2 secondnumber = 20
3 if firstnumber == secondnumber:
4    print('equal')
5    print('good')
6 else:
7    print('not equal')
```

Figure 1.3

```
1) firstnumber
2) =
3) 10
4) NEWLINE

5) secondnumber
6) =
7) 20
8) NEWLINE

9)  if
10) firstnumber
11) ==
12) secondnumber
13) :

14) NEWLINE
15) INDENT
16) print
17) (
18) equal
19) )
20) NEWLINE

21) print
22) (
23) good
24) )
25) NEWLINE
25) DEDENT

27) else
28) :

29) NEWLINE
30) INDENT
31) print
32) (
33) not equal
34) )
35) NEWLINE
36) DEDENT

37) EOF
```

Code block for `if` part

Code block for `else` part

Figure 1.4

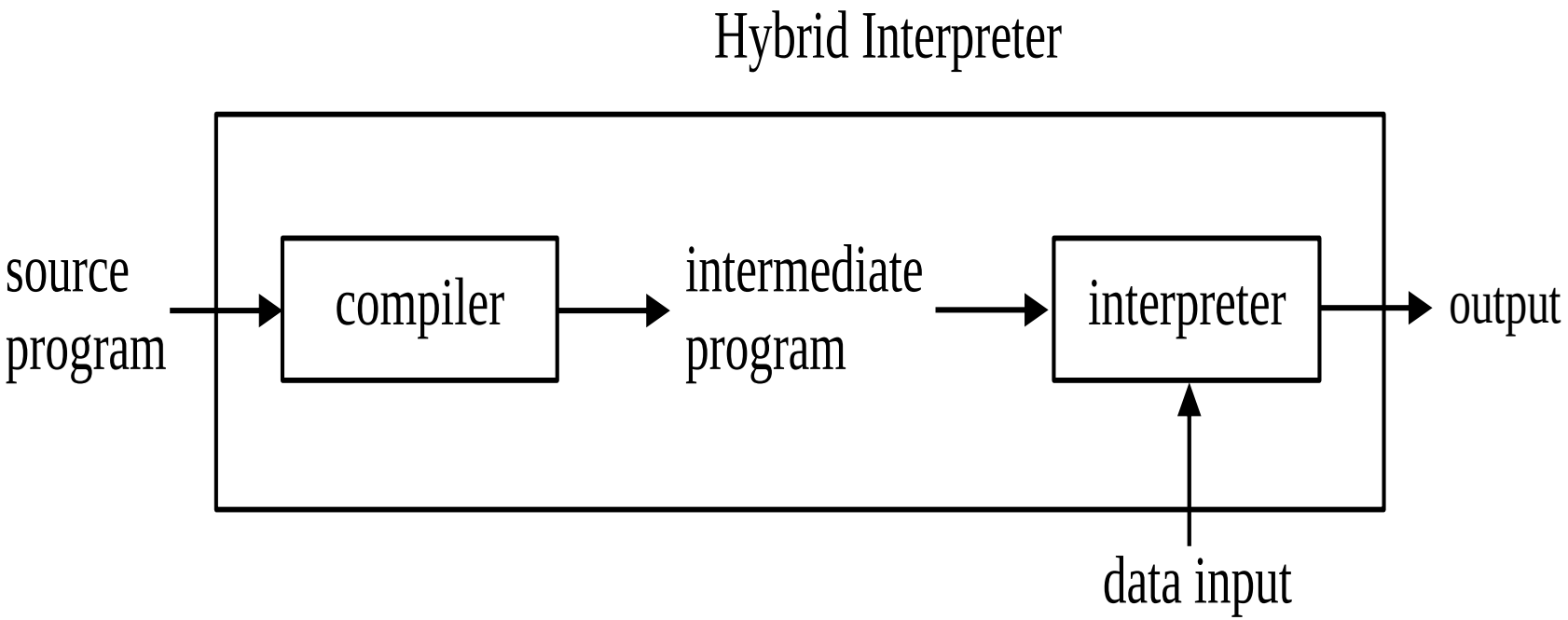

Hybrid Interpreter
source
program
compiler
intermediate
program
interpreter
output
data input

Figure 1.5

Advantages of Interpreters over Compilers

- Immediate execution—no user-observable compile or link steps
- Portability
- Easier to implement—do not need to know machine architecture to write an interpreter
- Easy to produce error messages meaningful to end user

But slow!