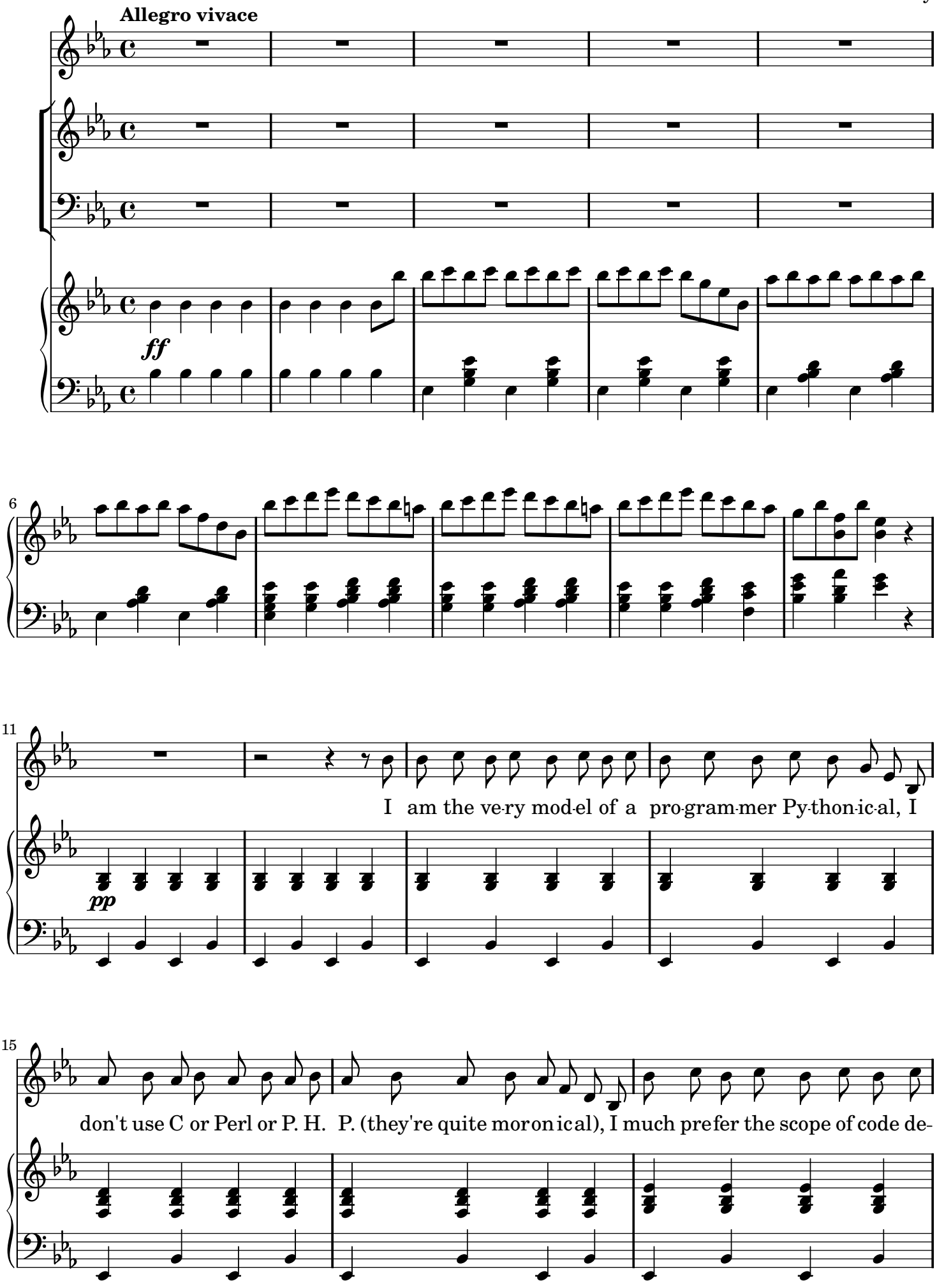
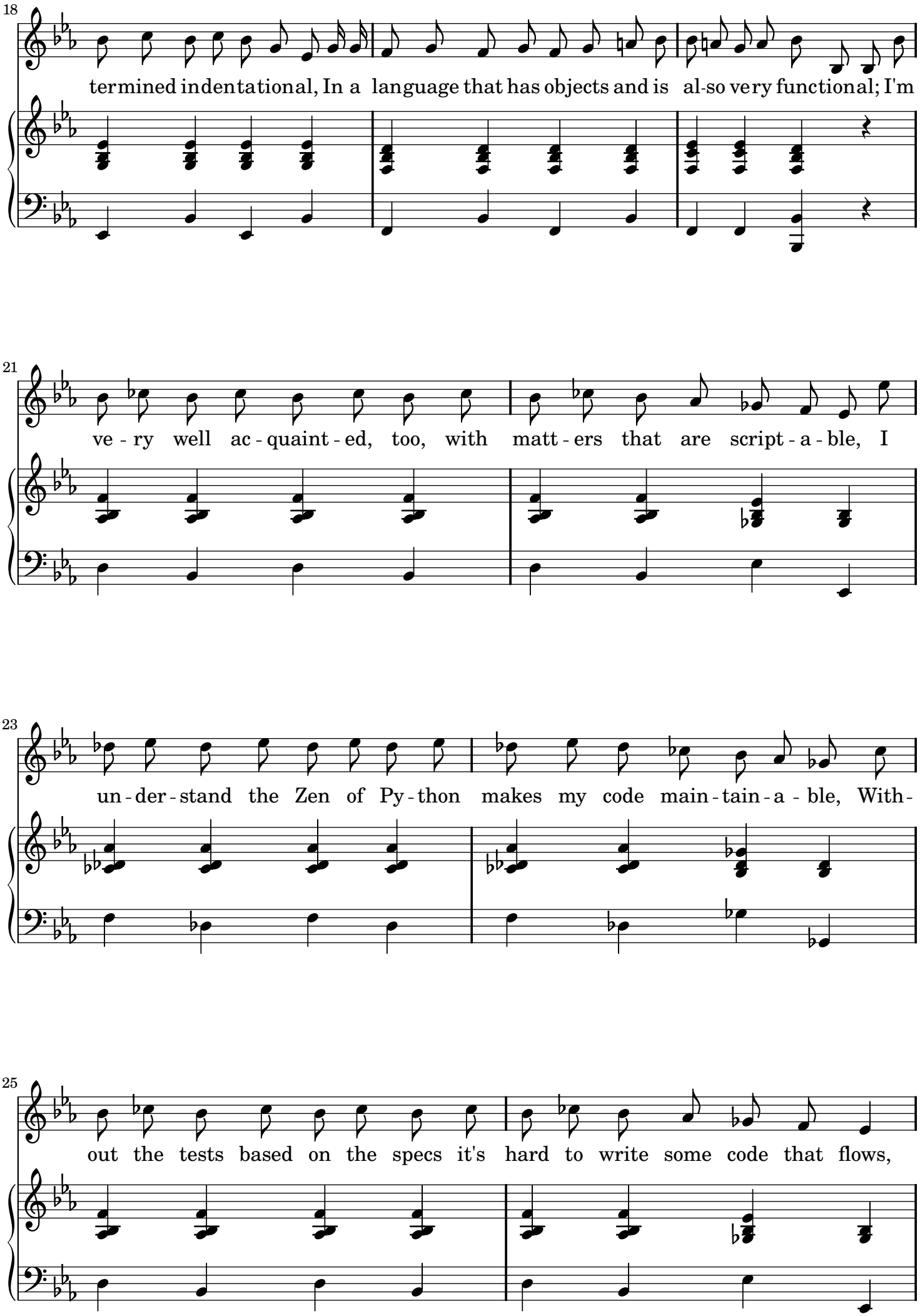
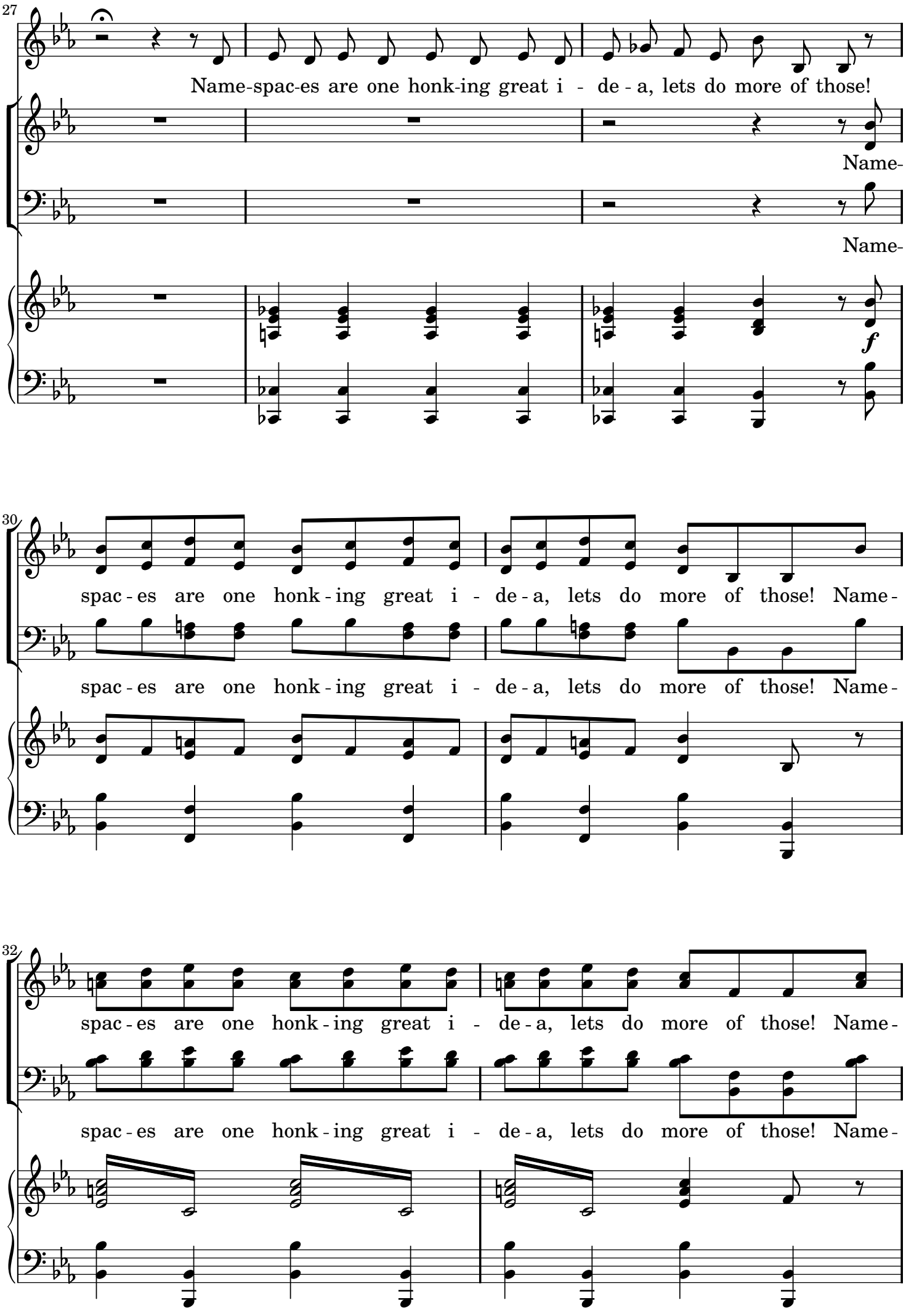
The Model Programmer Pythonical

With apologies to Gilbert and Sullivan

S.Hawkes & N.H.Tollervey




18
termined indentational, In a language that has objects and is al-so very functional; I'm
21
ve - ry well ac - quaint - ed, too, with matt - ers that are script - a - ble, I
23
un - der - stand the Zen of Py - thon makes my code main - tain - a - ble, With -
25
out the tests based on the specs it's hard to write some code that flows,


27
Name-spac-es are one honk-ing great i - de - a, lets do more of those!
Name-
Name-
f
30
spac - es are one honk - ing great i - de - a, lets do more of those! Name-
spac - es are one honk - ing great i - de - a, lets do more of those! Name-
32
spac - es are one honk - ing great i - de - a, lets do more of those! Name-
spac - es are one honk - ing great i - de - a, lets do more of those! Name-

34

spac-es are one honk-ing great i - de-a, lets do more and more of those!

spac-es are one honk-ing great i - de-a, lets do more and more of those!

fz

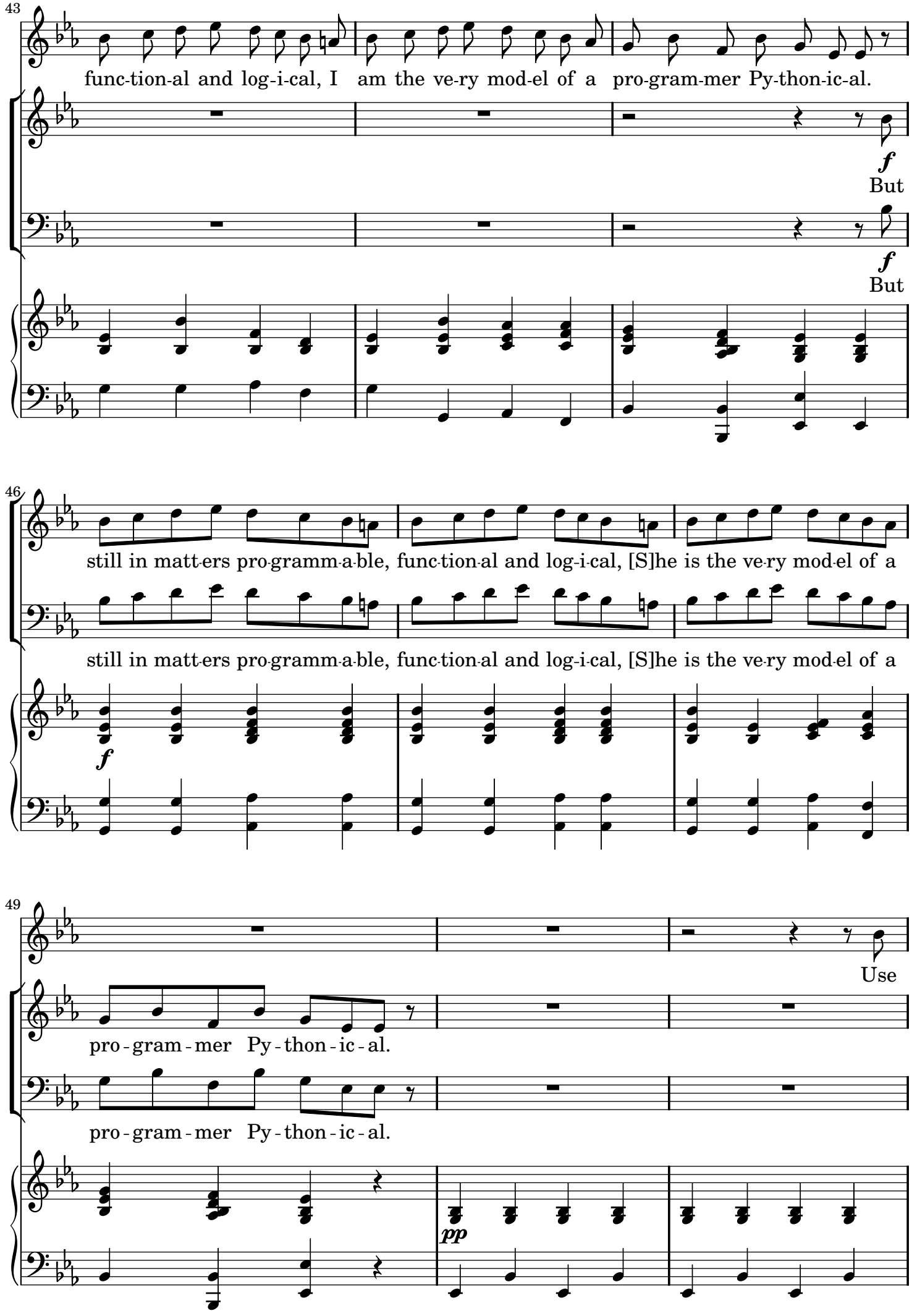
37

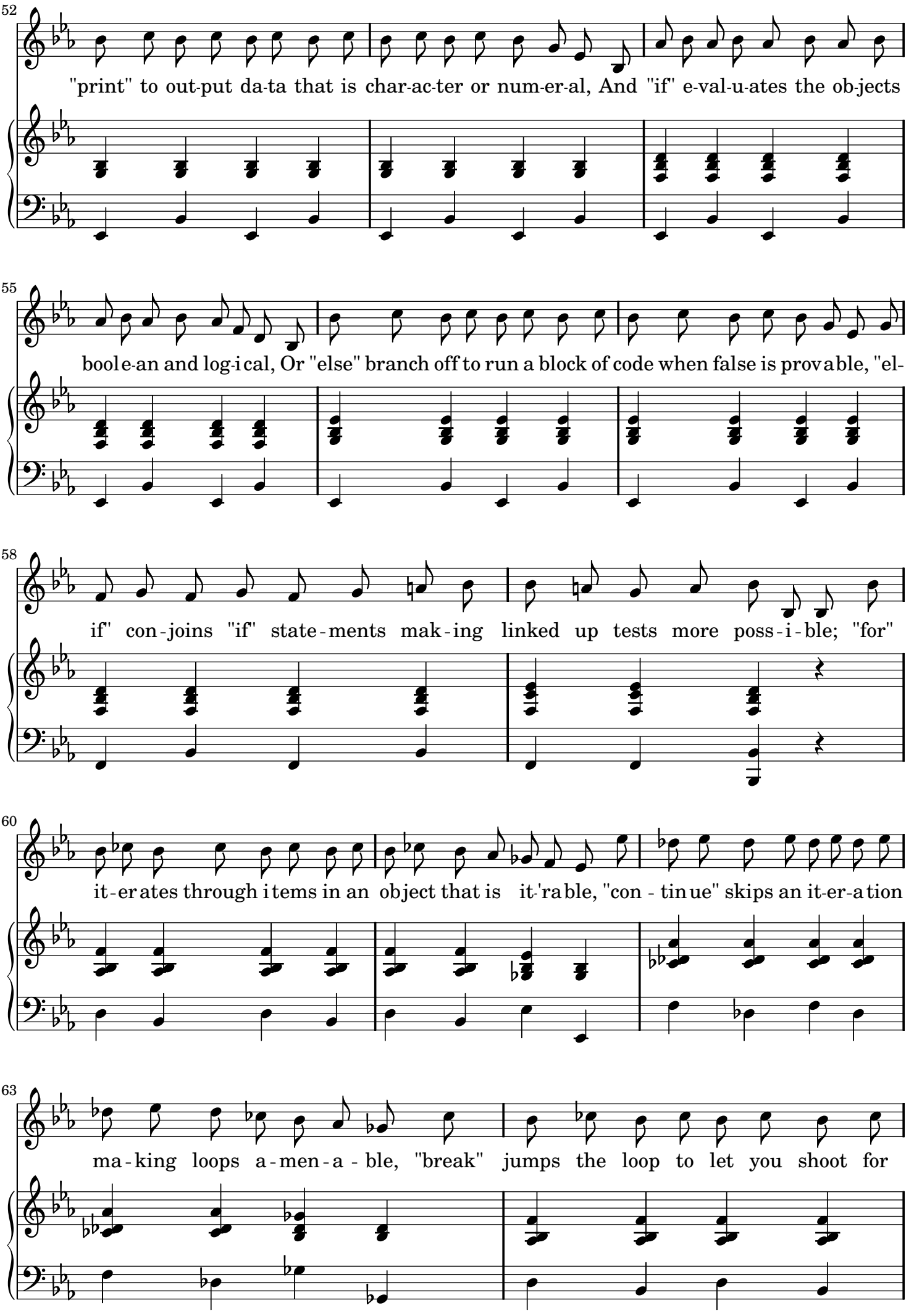
I'm good at yield-ing stuff that makes my func-tion gen-er-at-ab-le, I

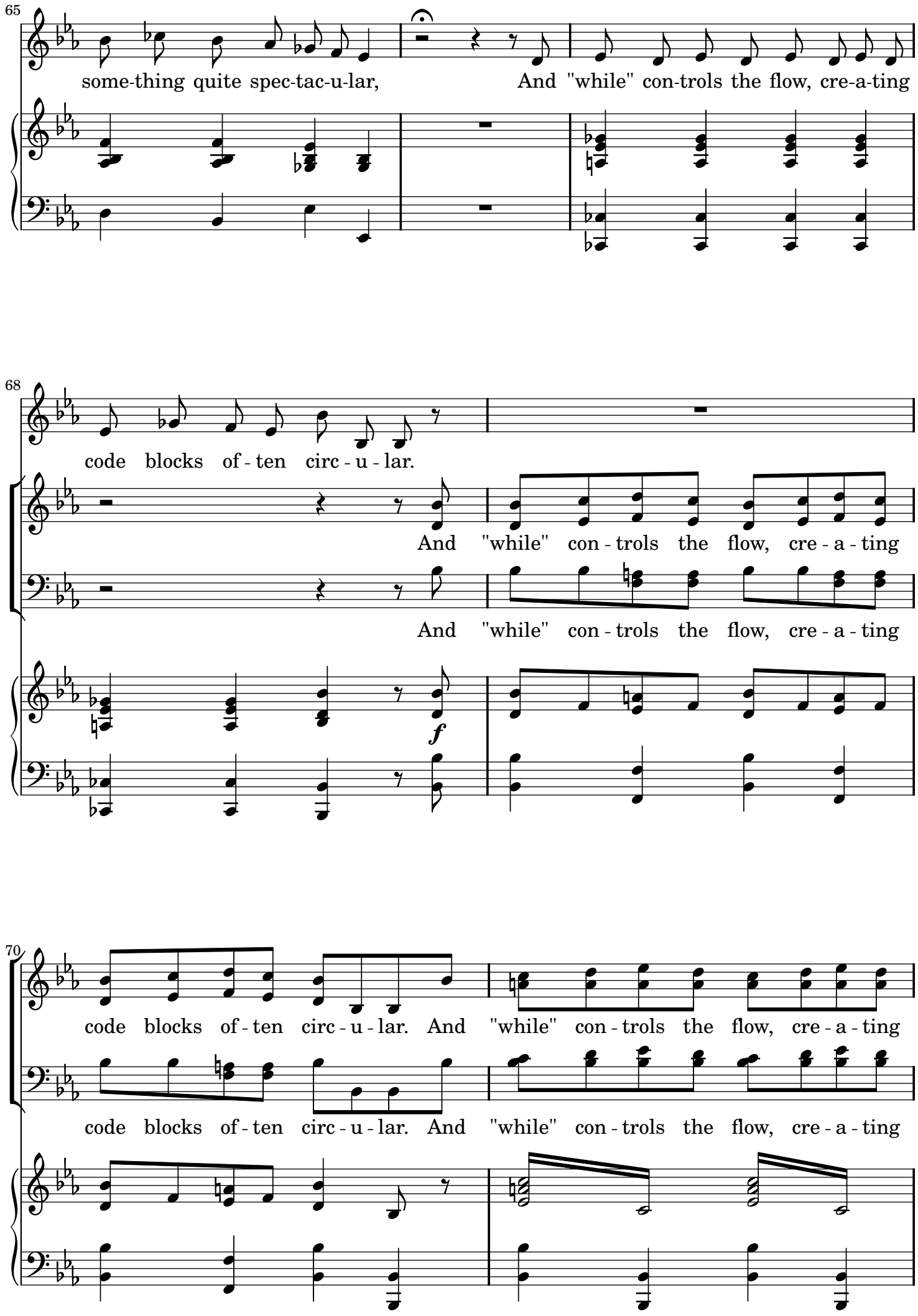
pp

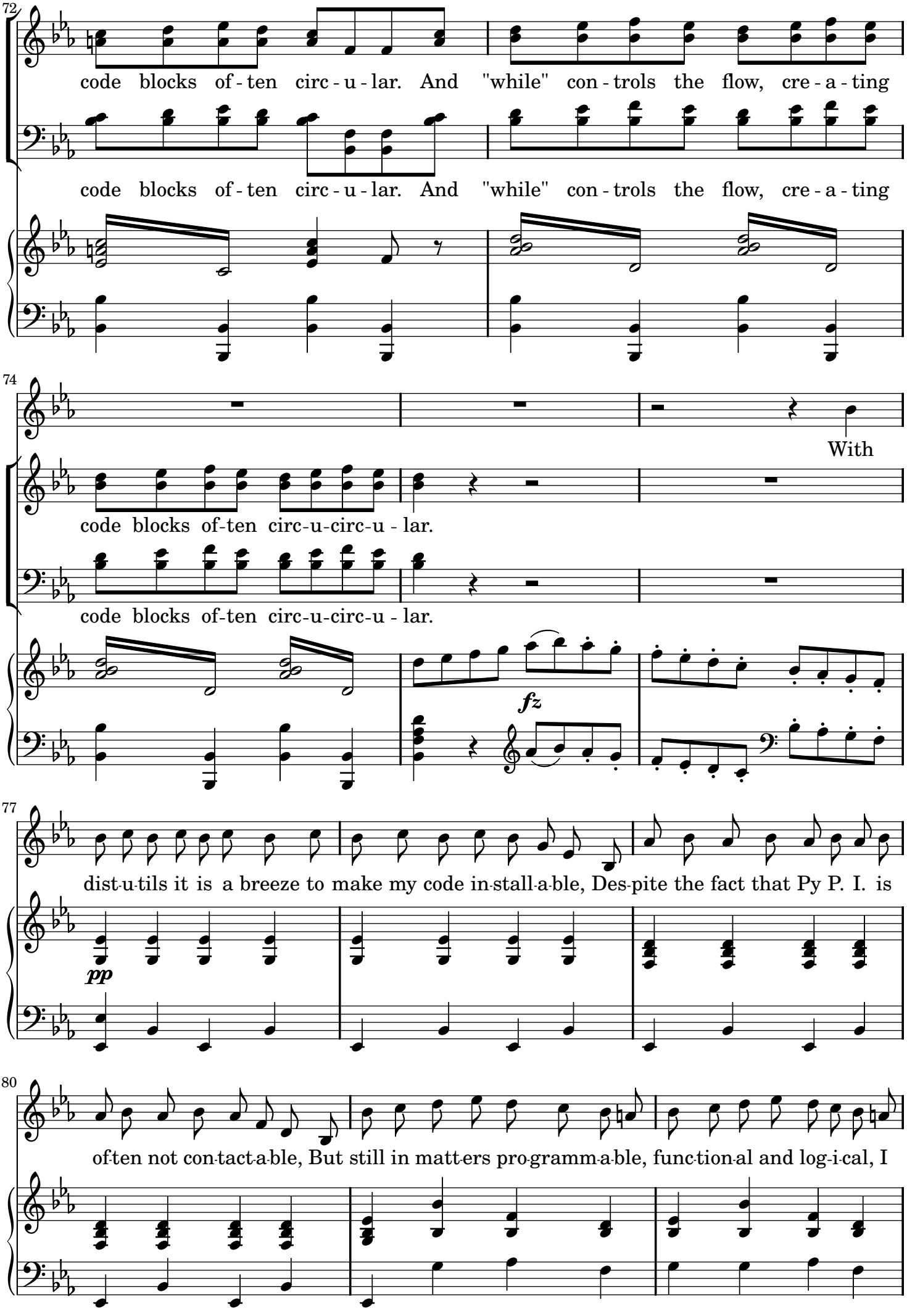
40

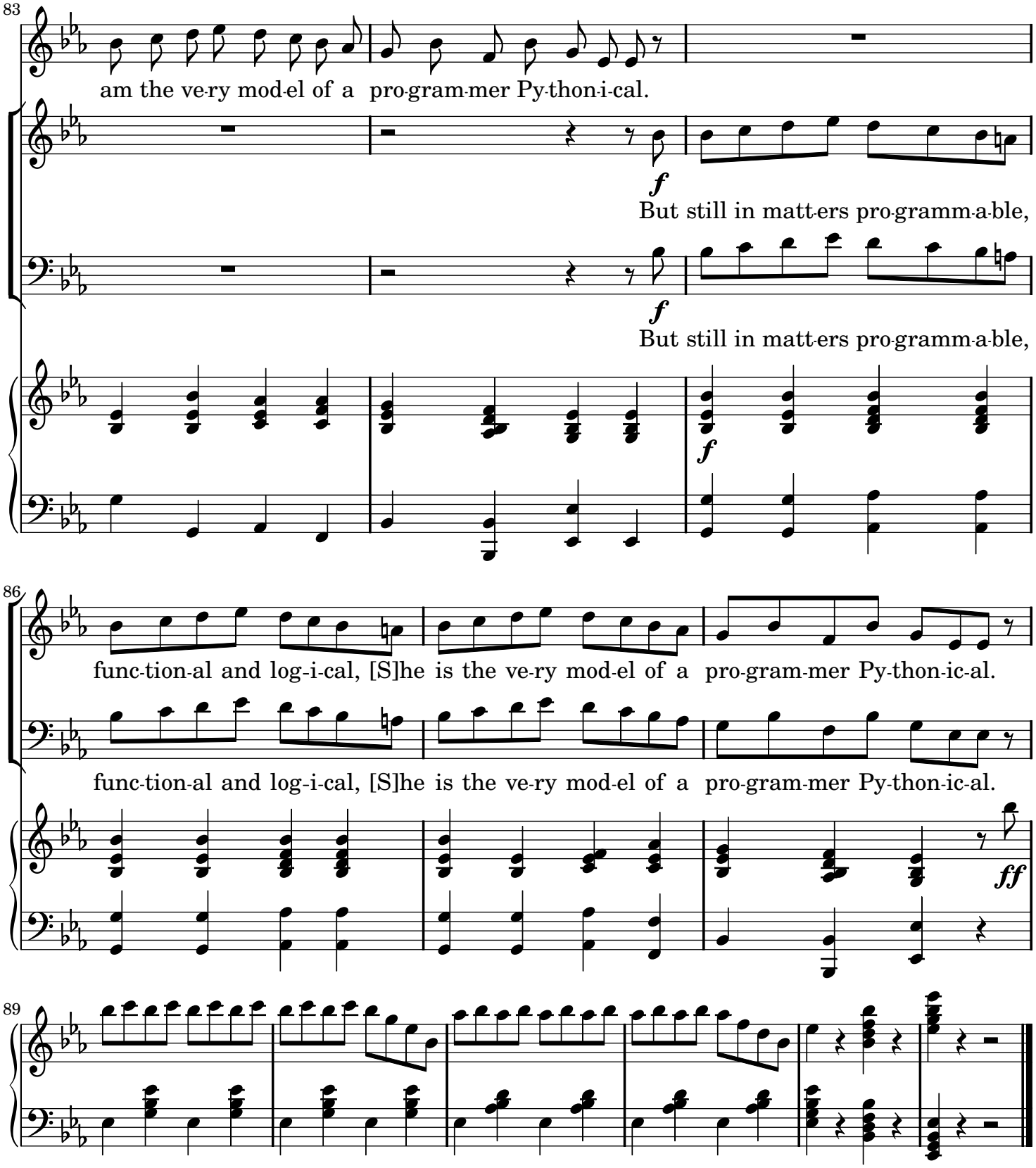
cre-ate class-es that are nice and eas-i-ly in-her't-ab-le, But still in mat-ters pro-gramm-a-ble,


43
func-tion-al and log-i-cal, I am the ve-ry mod-el of a pro-gram-mer Py-thon-ic-al.
f
But
f
But
46
still in matt-ers pro-gramm-a-ble, func-tion-al and log-i-cal, [S]he is the ve-ry mod-el of a
still in matt-ers pro-gramm-a-ble, func-tion-al and log-i-cal, [S]he is the ve-ry mod-el of a
f
49
Use
pro - gram - mer Py - thon - ic - al.
pro - gram - mer Py - thon - ic - al.
pp


52
"print" to out-put da-ta that is char-ac-ter or num-er-al, And "if" e-val-u-ates the ob-jects
55
boole-an and log-i cal, Or "else" branch off to run a block of code when false is prov able, "el-
58
if" con-joins "if" state-ments mak-ing linked up tests more poss-i-ble; "for"
60
it-er ates through i tems in an ob ject that is it'ra ble, "con - tin ue" skips an it-er-a tion
63
ma-king loops a-men-a - ble, "break" jumps the loop to let you shoot for


65
some-thing quite spec-tac-u-lar,
And "while" con-trols the flow, cre-a-ting
68
code blocks of - ten circ - u - lar.
And "while" con - trols the flow, cre - a - ting
And "while" con - trols the flow, cre - a - ting
f
70
code blocks of - ten circ - u - lar. And "while" con - trols the flow, cre - a - ting
code blocks of - ten circ - u - lar. And "while" con - trols the flow, cre - a - ting


72
code blocks of - ten circ - u - lar. And "while" con - trols the flow, cre - a - ting
code blocks of - ten circ - u - lar. And "while" con - trols the flow, cre - a - ting
74
With
code blocks of-ten circ-u-circ-u - lar.
code blocks of-ten circ-u-circ-u - lar.
fz
77
dist-u-tils it is a breeze to make my code in-stall-a-ble, Des-pite the fact that Py P. I. is
pp
80
often not con-tact-a-ble, But still in matt-ers pro-gramm-a-ble, func-tion-al and log-i-cal, I


83
am the ve-ry mod-el of a pro-gram-mer Py-thon-i-cal.
f
But still in matt-ers pro-gramm-a-ble,
f
But still in matt-ers pro-gramm-a-ble,
f
86
func-tion-al and log-i-cal, [S]he is the ve-ry mod-el of a pro-gram-mer Py-thon-ic-al.
func-tion-al and log-i-cal, [S]he is the ve-ry mod-el of a pro-gram-mer Py-thon-ic-al.
ff
89