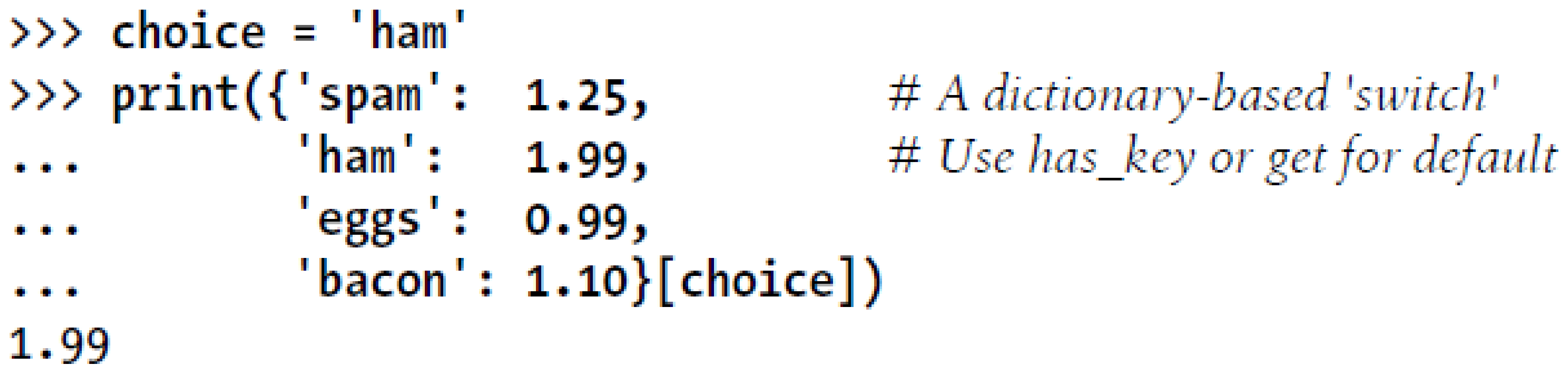
PYTHON – IF TESTS AND SYNTAX RULES

Use “else” for a catch all

```
if choice == 'spam':
 print(1.25)
elif choice == 'ham':
 print(1.99)
elif choice == 'eggs':
 print(0.99)
elif choice == 'bacon':
 print(1.10)
else:
 print('Bad choice')
```

SWITCH

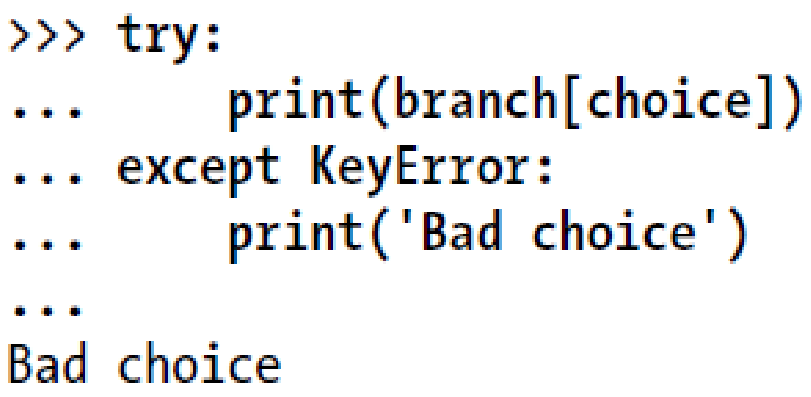


```
>>> choice = 'ham'
>>> print({'spam':  1.25,            # A dictionary-based 'switch'
...        'ham':   1.99,            # Use has_key or get for default
...        'eggs':  0.99,
...        'bacon': 1.10}[choice])
1.99
```

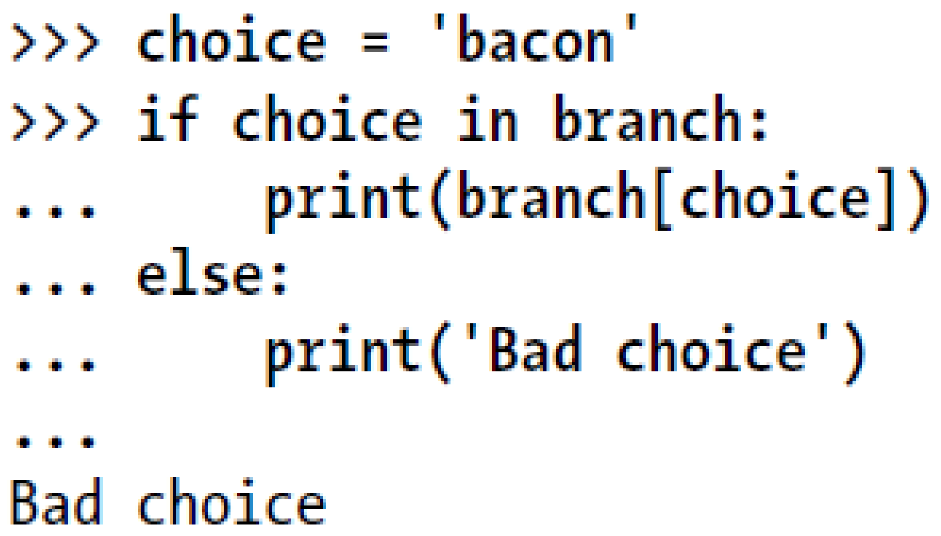
SWITCH DEFAULT USING IF

```
>>> branch = {'spam': 1.25,
...           'ham':  1.99,
...           'eggs': 0.99}

>>> print(branch.get('spam', 'Bad choice'))
1.25

>>> print(branch.get('bacon', 'Bad choice'))
Bad choice
```

TRY AND CATCH



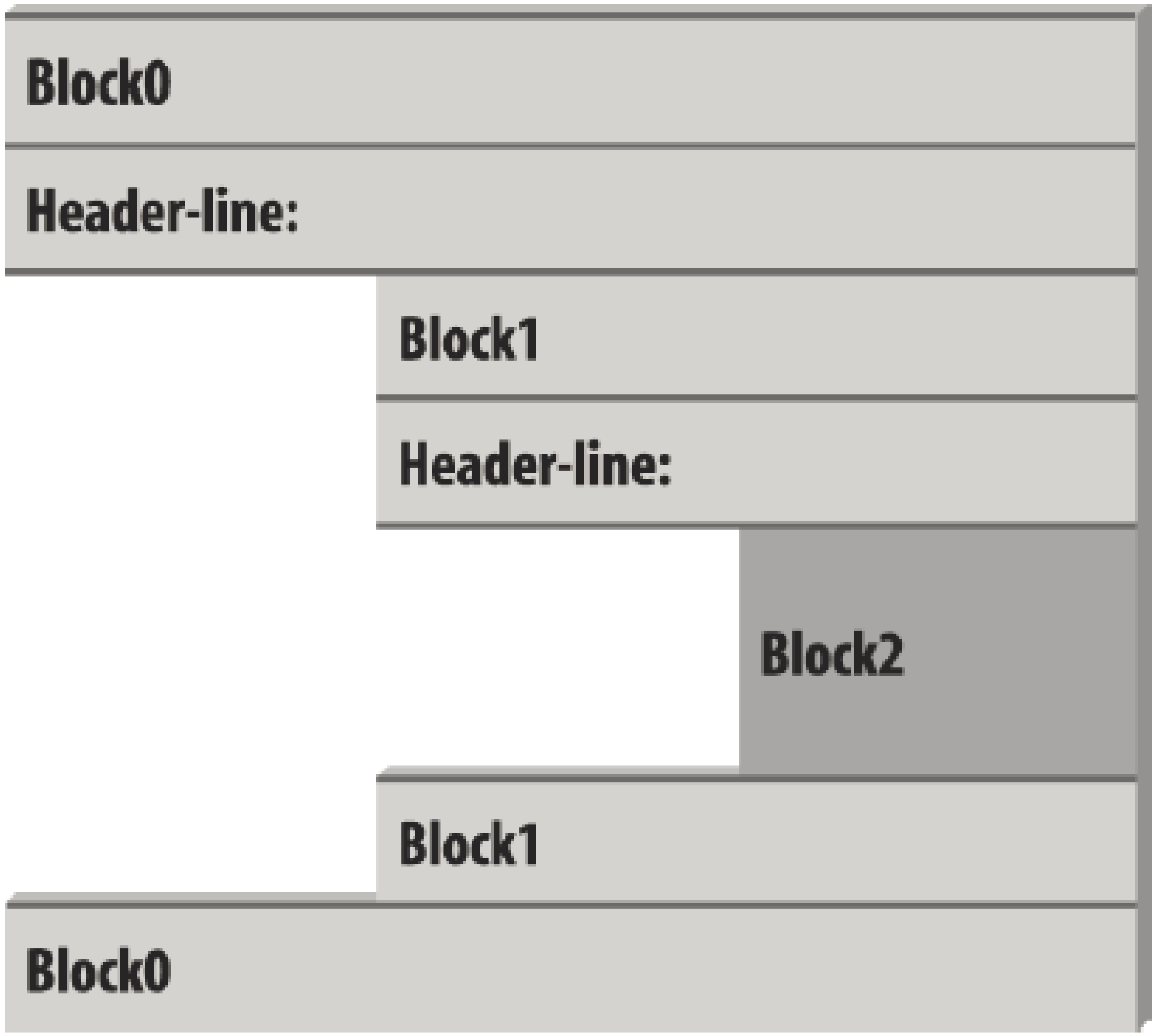
```
>>> try:
...     print(branch[choice])
... except KeyError:
...     print('Bad choice')
...
Bad choice
```



```
>>> choice = 'bacon'
>>> if choice in branch:
...     print(branch[choice])
... else:
...     print('Bad choice')
...
Bad choice
```

Block Delimiters: Indentation Rules

- Python detects block boundaries “automatically” by line indentation
- All statements indented the same distance belong to the same block of code.
  - blocks line up vertically, as in a column.
  - The block ends when a lesser indented line is encountered.
- More deeply nested blocks are simply indented further to the right.


Block0
Header-line:
Block1
Header-line:
Block2
Block1
Block0

Does this work?

```
x = 1
if x:
 y = 2
 if y:
     print(‘deepest block')
 print(‘middle block')
print(‘outside block')
```

Does this work?

```
  x = 'SPAM'
if 'rubbery' in 'shrubbery':
    print(x * 8)
        x += 'NI'
        if x.endswith('NI'):
                x *= 2
            print(x)
```

Tabs and Spaces

Avoid mixing tabs and spaces:

- Use spaces or tabs to indent, don’t mix
  - Py3 issues errors!

- Technically, tabs count for enough spaces to move the current column number up to a multiple of 8

Python’s Boolean operators

- Any nonzero number or nonempty object is true.
- Zero numbers, empty objects, and None are considered false.
- Comparisons and equality tests are applied recursively to data structures.
- Comparisons and equality tests return True or False
- Boolean “and” and “or” operators return a true or false operand object

Boolean operators are used to combine the results of other tests.

The three Boolean expression operators in Python:

1. X and Y
2. X or Y
3. not X

Ex: if (x==y) or (y==z and y==q): print('ok')