IF,IF/ELSE,IF/ELIF/ELSE STATEMENTS

Python Programming — Auburn University

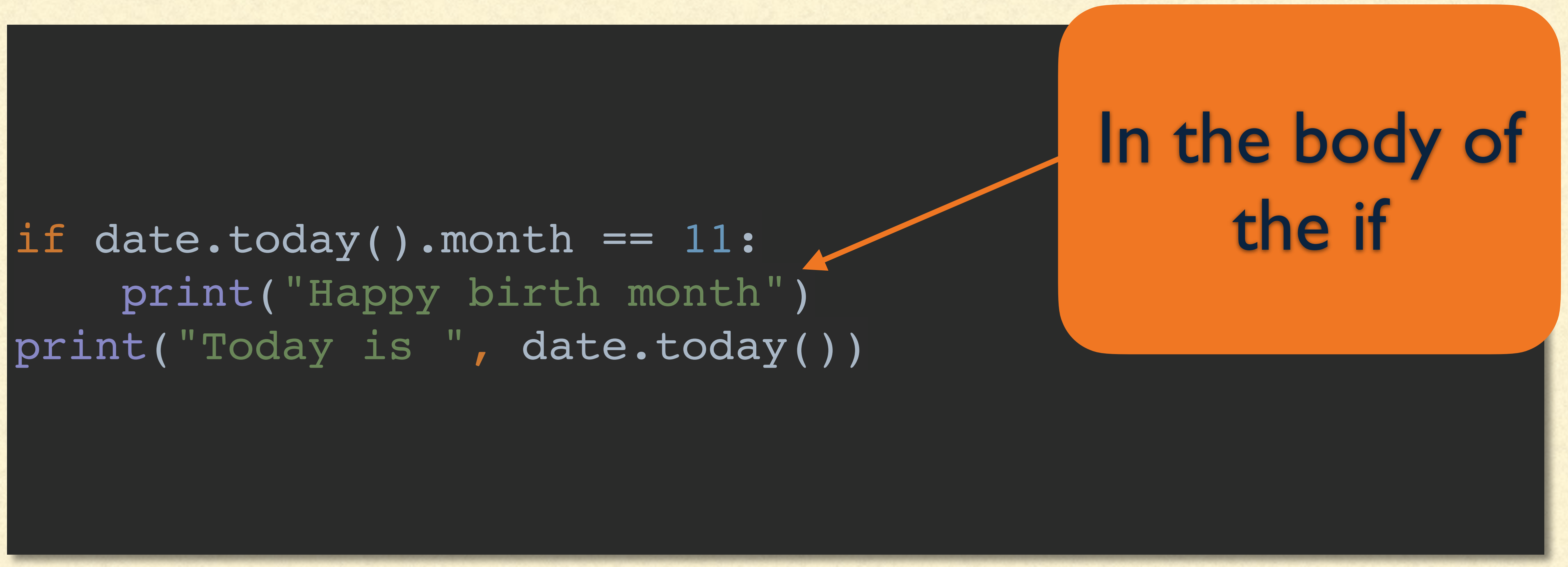
IF STATEMENT

- Level of indentation is used to determine which statements are in the body and which aren't.
- Can indent multiple times for nested structures.
- Notice no parentheses around conditional expression.
- Be careful of spaces vs tabs (not a problem in PyCharm).

```python
if date.today().month == 11:
    print("Happy birth month")
print("Today is ", date.today())
```

IF STATEMENT

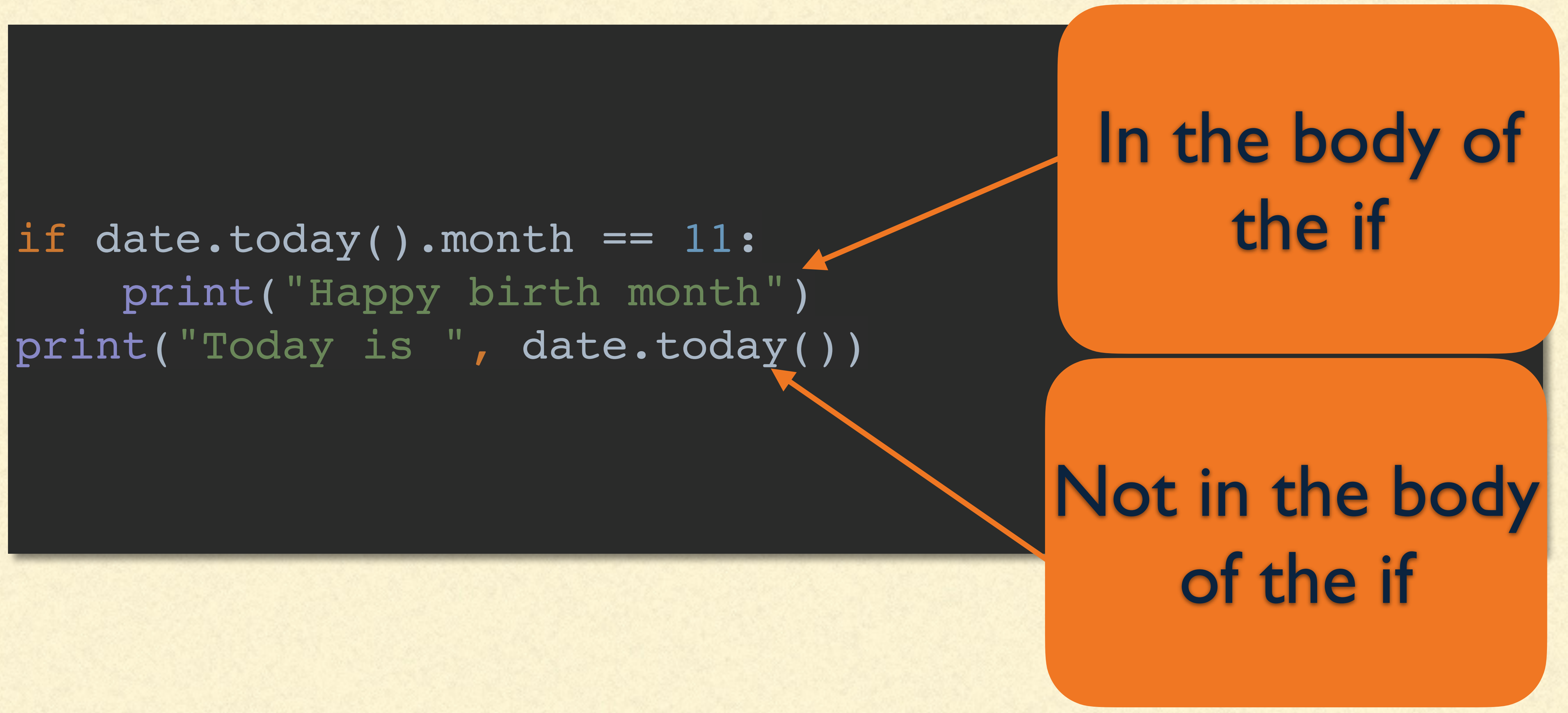
- Level of indentation is used to determine which statements are in the body and which aren’t.
- Can indent multiple times for nested structures.
- Notice no parentheses around conditional expression.
- Be careful of spaces vs tabs (not a problem in PyCharm).



```python
if date.today().month == 11:
    print("Happy birth month")
print("Today is ", date.today())
```

IF STATEMENT

- Level of indentation is used to determine which statements are in the body and which aren't.
- Can indent multiple times for nested structures.
- Notice no parentheses around conditional expression.
- Be careful of spaces vs tabs (not a problem in PyCharm).



```python
if date.today().month == 11:
    print("Happy birth month")
print("Today is ", date.today())
```

IF STATEMENT

- Level of indentation is used to determine which statements are in the body and which aren't.
- Can indent multiple times for nested structures.
- Notice no parentheses around conditional expression.
- Be careful of spaces vs tabs (not a problem in PyCharm).

```python
if date.today().month == 11:
    print("Happy birth month")
print("Today is ", date.today())
```

IF STATEMENT

- Level of indentation is used to determine which statements are in the body and which aren't.
- Can indent multiple times for nested structures.
- Notice no parentheses around conditional expression.
- Be careful of spaces vs tabs (not a problem in PyCharm).

```
if date.today().month == 11:
    print("Happy birth month")
print("Today is ", date.today())
```

```
if x < y and x // 2 == 0:
    print("Example with multiple")
    print("statements in the body of the if")
    if x > 99:
        print("and a nested statement")
```

TRUTH VALUES

- Conditions in if statements may be non-boolean.
- Any object whose __bool__() method return False or whose __len__() returns 0 is considered a “false value.”
- For example, the following values are considered False:
    - the constants `None` and `False`.
    - zero of any numeric type: `0, 0.0, 0j, Decimal(0), Fraction(0, 1)`
    - empty sequences and collections: `'', (), [], {}, set(), range(0)`
- Most other values of built-in types are considered true.
- In practice I make *very little* use of this.

INLINE BODY

- If the body of an if (or any other control structure) is a single **simple statement**, it may be placed inline.
  - Simple statements have no body. Examples include assignment, message sends etc.
  - Compound statements have a body. Examples include: if, while, for
- Can have multiple statements in body by separating them with semicolons (**discouraged).**
- Use inline body only when it increases readability.

```
if x == 3: print("yes!")     # VALID
if x == 3: if y == 0: print("Hello")   # INVALID

if x == 3: print("Hello"); print(“goodbye") # VALID
```

IF/ELSE STATEMENT

- No surprises here…

```
lst = ["monkey", "banana", "tree"]
if "monkey" in lst:
    print("monkey is in the list")
else:
    print("the monkey is missing!")
```

MULTI-WAY SELECTION

- Note: Python does not have a “switch” statement but you probably won’t miss it.

```
if x < 20:
    print("less than 20")
elif x < 30:
    print("less than 30")
elif x < 40:
    print ("less than 40")
else:
    print("greater than or equal to 40")
```

EXPRESSIONS VS STATEMENTS

- Sometimes it would be nice if "`if`" was an expression. Consider:

```
print(“Hello ”, end=“”)
if name is None:
   print(“no name”)
else:
   print(name)
```

  versus

```
print(“Hello”, (if name is None: “no name” else: name))
```

- Unfortunately if is a statement so it can't be used this way.
- Same issue in Java/C++. Enter the `?:` operator…

SELECTION EXPRESSIONS IN JAVA `(?:)`

- `?:` is an operator (of arity 3 so it is called a "ternary" operator). The value of:

  ```
      condition ? a : b
  ```

  is a if `condition` is true, otherwise its value is b.

- Example:

  ```
      3 == 2 ? “blah” : “blim”      is equal to “blim”
      3 == 3 ? “blah” : “blim”      is equal to “blah”
  ```

- Let’s us write things like

  ```
      System.out.println(“Hello “+ (name == null ? “no name” :
  name));
  ```

SELECTION EXPRESSIONS IN PYTHON

- Python also has two ternary selection expressions: if-else construction and the and-or construction.
- Choose whichever one you like :)
- `a if condition else c` – evaluates to a if `condition` is `Truthy`, otherwise b
- Examples:

```
    print(“blah” if 3 == 2 else “blim”)    # prints blim
    print(“blah” if 3 == 3 else “blim”)    # prints blah
    print(“Hello “, “no user” if name is None else name)
```

SELECTION EXPRESSIONS IN PYTHON

- and-or construction
- `condition and a or b` — evaluates to a if `condition` is `Truthy`, otherwise `b`
- Examples:

```
print(3 == 2 and “blah” or “blim”)     # prints blim
print(3 == 3 and “blah” or “blim”)     # prints blah
print(“Hello”, name is None and “no user” or name())
```