Program Execution

• In what order, will the following lines of code be executed?

```python
num1 = 10
double = num1 * 2
print("double refers to value", double)
```

• But what if you want to execute some code, if and only if, some specific conditions exist?
  • Solution: if-statements

  Execute this first.
  Then, execute this.
  And finally execute this call to function print.
If Statements

<table>
<thead>
<tr>
<th>English</th>
<th>Python</th>
</tr>
</thead>
</table>
| Check The Temperature:                                                 | def check_temp(temperature):
| If the temperature > 0 then                                            |    if temperature > 0:
| it is above the freezing point                                        |        return "above the freezing point."
| Otherwise, if the temperature == 0 then                               |    elif temperature == 0:
| it is at the freezing point                                           |        return "at the freezing point."
| Otherwise                                                              |    else:
| it is below the freezing point                                         |        return "below the freezing point."
| **Condition to check**                                                 | **If, elif and else are reserved keywords in Python**
If Statement Format

• Both **if** and **elif** should be followed by a condition.

• **else** does not have a condition. It’s the equivalent of “in all other cases not covered by the previous if/elif conditions”.

• The order in which you write the if/elif conditions matters.

\[
\text{if condition}_1:\n\text{statements}_1
\]
\[
\text{elif condition}_2:\n\text{statements}_2
\]
\[
\text{elif} \ldots
\]
\[
\text{else}:
\text{statements}_N
\]
If Statement – Execution Order

Form for writing an if statement:

```python
if condition_1:
    statements_1
elif condition_2:
    statements_2
elif ...
...
else:
    statements_N
```

The execution:

- evaluate `condition_1`
- if the result is `True`, execute `statements_1` and skip the rest of the if statement
- otherwise, evaluate `condition_2`
- if the result is `True`, execute `statements_2` and skip the rest of the if statement
- ...
- otherwise, execute `statements_N`
Lecture Takeaways (added after lecture)

• Type `bool` only has two possible values: True, False

• Relational (or comparison) operators are: >, >=, <=, <, ==, !=
  • Note: when comparing an int with a float, Python compares values and not types.
    • Thus, 2 == 2.0 evaluates to True, even though type(2) == type(2.0) evaluates to False.

• Logical operators are (in order of precedence): not, and, or
  • We can use multiple operators to write a more complex Boolean expression.
Lecture Takeaways cont’d (added after lecture)

• \( \text{var1} = \text{condition}_a \ \text{and} \ \text{condition}_b \)
  • \( \text{var1} \) will refer to True if and only if both \( \text{condition}_a \) and \( \text{condition}_b \) evaluate to True

• \( \text{var2} = \text{condition}_a \ \text{or} \ \text{condition}_b \)
  • \( \text{var2} \) will refer to True if at least one of the two conditions (\( \text{condition}_a \), \( \text{condition}_b \)) evaluates to True. Note, “at least one” means it’s OK if both \( \text{condition}_a \) and \( \text{condition}_b \) evaluate to True.
Administrative Notes

• Exercise E2 will be posted later this week; it will likely be due this coming Sunday (and not Wednesday as was the case in the past).

• Please remember to bring today’s worksheet to Wednesday’s lecture.