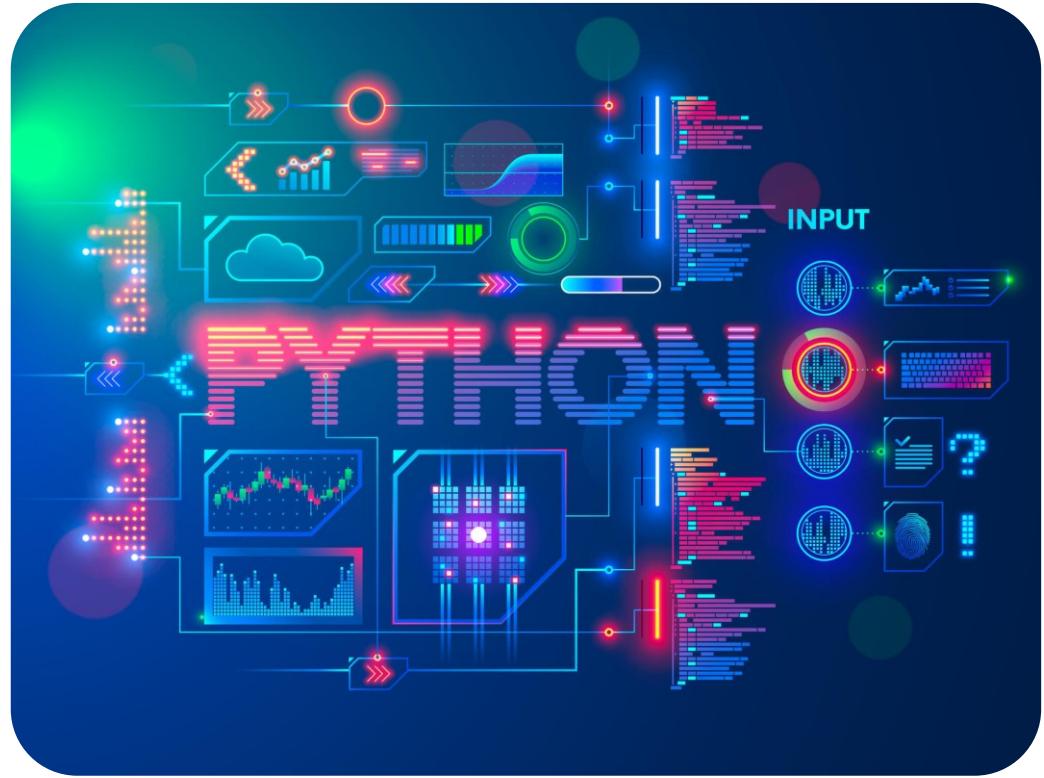


WIIT 7740: Scripting with Python

Week 2: Input, Variables, and Logical Control



Boolean

- Can only be two possible values. Variable type is used for logic operations.

Boolean values:

True

False

Variables

- Containers used for storing data values to be used later in the program; the data can be objects.
 - Example: `my_variable = 100`
- For Python variables, type does not need to be declared like other languages. Variables can change type by assigning a new value of a different type.
- Variables can also be "cast" to other type. Example:

```
my_variable = 100
print(type(my_variable))
my_variable = str(my_variable)
print(type(my_variable))
print(my_variable)
```

Output:
<class 'int'>
<class 'str'>
100

input()

- Python's built in method for reading input from the user
 - `user_input = input('Enter your age')`
- Input will always return a string, but this value can be cast if needed:
 - `user_input = int(input('Enter your age'))`

Logical Control: Outline

- **if**
- **else**
- **elif**
 - "else if"
- Boolean literals
 - True
 - False
- Comparison operators
 - <, >, ==, >=, <=, !=
- Boolean operators
 - **and**, **or**, **not**
- Nested conditionals
- Formatting

Logical Control Capabilities

- *Compare* one value with another
 - Are they equal?
 - Are they unequal?
 - Is one less/more than the other?
- Make *decisions* based on conditions
 - **If** some condition holds true, **then** do one thing, **else** do another thing.
 - If condition does not hold true...
 - If both of two conditions hold true...
 - If one of two conditions holds true...

Conditional Patterns

```
if condition:  
    do_something()
```

```
if condition:  
    do_something()  
else:  
    do_something_else()
```

```
if condition_1:  
    do_something()  
elif condition_2:  
    do_something_different()
```

```
if condition_1:  
    do_something()  
elif condition_2:  
    do_something_different()  
else:  
    do_something_else()
```

Logical Control Pattern: Inclusive Conditions

```
if "Albus" in meeting:  
    print("Albus is here.")  
  
if "Bernadette" in meeting:  
    print("Bernadette is here.")  
  
if "Cassandra" in meeting:  
    print("Cassandra is here.")
```

Note:

The `in` keyword returns True if the object `meeting` contains said value

Logical Control Pattern: Alternative

```
if "Albus" in meeting:  
    print("Albus is here.")  
else:  
    print("Albus is not here.")
```

Logical Control Pattern: Exclusive Conditions

```
if speaker == "Albus":  
    print("Albus may speak.")  
elif speaker == "Bernadette":  
    print("Bernadette may speak.")  
elif speaker == "Cassandra":  
    print("Cassandra may speak.")  
else:  
    print("No one may speak.")
```

Common Mistake: **if** instead of **elif**

```
if can_move:  
    print("animal")  
elif can_breathe:  
    print("vegetable")  
else:  
    print("mineral")
```

Nesting Conditionals

```
if chairperson in meeting:  
    if chairperson == speaker:  
        read_old_business()  
    else:  
        hear_new_business()
```

} Nested Conditional 1

```
else:  
    if vice_chairperson in meeting:  
        schedule_next_meeting()  
    else:  
        adjourn()
```

} Nested Conditional 2

Boolean Operators: Precedence

not ... and ... or

x and y or z differs from **x and (y or z)**

not x or y differs from **not (x or y)**

Formatting Data

```
print(x, y, z, sep=" ", end="\n")
```

```
format(1 / 7, ".3f") → ".143"
```

```
format(25 ** 3, ",.0f") → "15,625"
```

```
'My name is {}, and I'm {}'.format('Bob', 42)
```

Rounding Errors

Comparing float types can be unreliable due to rounding errors. Be careful comparing float values directly.

```
.1 + .2 == .3
```

```
(1 / 3) * 5 == 1.6666666666666665
```

```
(5 / 3)      == 1.6666666666666667
```

```
1.666666666666665 != 1.6666666666666667
```

Coding Practices

Bad	Good
<code>if condition == True:</code>	<code>if condition:</code>
<code>if condition == False:</code>	<code>if not condition:</code>

Glossary: Type, Functions/Method

Type

bool

Data type that can be one of two values, True or False

Functions/Method

print(object(s), sep, end, file, flush)

prints object(s) to the standard output

input(prompt)

reads input from the user from via the standard input (normally the command line)

string.format(value(s))

used for complex formatting of strings

Example: `'This is {} {}st formatted {}!'.format('my', 1, 'string')`

Glossary: Keywords and Operators

Keywords

if

starting condition for a condition tree

elif

sub-sequent condition of a condition tree; can't be the first condition, but can be the last

else

default condition of a condition tree; must be the last condition and will run if all other conditions fail

and

logical AND (^) operator

or

logical OR (v) operator

not

logical NOT (¬) operator

Operators

< less than

> greater than

== equal

!= not equal

<= less than or equal

>= greater than or equal

Practice Coding: Class Activity

