CS 130R: Programming in Python

#3: Indentation, IF_ELSE

Reading: Chapter 2, Chapter3
pp.49-59
Contents

- String concatenation
- Line continuation character `\`
- User input
- Conversion Functions
- Indentation
- IF_ELSE statement
String Concatenation

**Concatenating strings** means joining them together to create a whole new string.
Operation is done with the ‘+’ operator.

```python
string1 = “String 1 ”
string2 = “String 2”
print(string1, string2)
String 1 String 2
```

```python
stringConcat = string1 + string2
print(stringConcat)
String 1 String 2
```
String Methods

upper() / lower() /capitalize() / title()

strg = “Python”

print(strg.upper())

strip() – all white spaces at the beginning and end are removed

replace(old, new [,max]) – the occurrences of string old are replaced with the string new; max marks the number of these replacements.
String Methods

strg = "abcdddddabcdxs abcdwllabc"
print(strg.replace("abcd","mmm"))
mmmdddmmmmxs mmmwllabc

print(strg.replace("abcd","mmm",2))
mmmdddmmmmxs abcdwllabc
Line continuation character ‘\’

‘\’ or the backslash character can be used to continue a statement on the next line.

```python
print("This string is a long string that continues on the next \ 
+ "line to demonstrate the usage of the continuation character.")
```

This string is a long string that continues on the next line to demonstrate the usage of the continuation character.

```python
mylongstring = "string 1, " + " string 2, " + \ 
    "string3, " + "string 4"

print(mylongstring)
```

string 1, string 2, string3, string 4
User input – the input() function

name = input('Hi. What is your name? ')  
print(“Hello “, name)

Hi. What is your name?
<program waits for user input>
Joe

Hello Joe

The input() function waits for input from user, given as string. Once Enter is pressed, the function returns the string that was given
User input

```python
input("\n\nPress the Enter key to exit program.\")
```
This line makes the program wait until user presses the “Enter” key.

**Input always returns a string!**

```python
age = input("How old are you? ")
age = int(age)  # to convert string into integer
```
Or in one line

```python
age = int(input("How old are you? "))
```
Conversion Functions

- **float(x)** – returns a floating point value by converting x

```python
floatFromString = float("12.3")
```
12.3 as float number

- **int(x)** – returns an integer value by converting x

```python
intFromString = int("10")
```
10 is an integer

- **str(x)** – returns a string value by converting x

```python
stringFromInteger = str(15)
```
‘15’
Indentation

Python uses indentation (i.e., the number of spaces left at the beginning of a statement – start counting spaces from the left border) to create blocks and interpret code.

A block is one or more consecutive lines indented by the same amount. (Hint: use 4 spaces for each block)
IF statement

IF statement allows for executing a block if a condition is true.

if condition:
    execute block

if strg == “python”:
    print(“IF statement is executed...”)

IF statement

if strg == “python”:
    print(“IF statement is executed...”)
IF statement is executed...

Careful:
if strg = “python”:
    print(“IF statement is executed...”)
This strg = “python” is not a condition. It assigns value “python” to string str
Types and Variables

• **Comparison Operators**: There are 6 comparison operators. The result of applying the comparison operators is a Boolean - *True* or *False*.
  
  – Equal to: `==`
  – Not equal to: `!=`
  – Greater than: `>`
  – Greater than or equal to: `>=`
  – Less than: `<`
  – Less than or equal to: `<=`
Types and Variables

• **Boolean Operators**: There are 3 Boolean operators:
  
  – *not*: unary operator that returns *True* if the operand is *False* and vice versa.
  
  – *x and y*: if *x* is false, then that value is returned; otherwise *y* is evaluated and the resulting value is returned.
  
  – *x or y*: if *x* is true, then that value is returned; otherwise *y* is evaluated and the resulting value is returned.
IF ELSE statement

IF ELSE statement allows for executing a block if a condition is true and another block if the condition is false.

```python
if condition:
    execute block1
else:
    execute block2
```

```python
if strg == "python":
    print("IF condition is true...")
else:
    print("IF condition is false...")
```
IF ELIF ELSE statement

IF ELIF ELSE statement allows for executing a block based on a decision of which condition is true from several conditions.

    if condition1:
            execute block1
    elif condition2:
            execute block2
    elif condition3:
            execute block3
    else:    # execute only when no condition is true
            execute block
IF  ELIF ELSE statement

if strg == "python":
    print("PYTHON condition is true...")
elif strg == "java":
    print("JAVA condition is true...")
elif strg == "C++":
    print("C++ condition is true...")
else:
    print("NO condition is true...")
print("Done choosing language!")

If my string is "java" program outputs:
JAVA condition is true...
Done choosing language!

If my string is strg = "abc" program outputs:
NO condition is true...
Done choosing language!
Next lecture ...

• While Loops
• For loops