CS 1340:Fall 2020:Lecture 03

Intro to Python for CS and Data Science

Mark Fontenot, PhD

Southern Methodist University

Finishing up Slides from Thursday

Basic Output

```
print(...)
```

- Notice:
  - printed in mono-spaced font
  - ... means other stuff will be put there
  - () indicate a method or function call

```
print('Hello')
print('World')
print('Hello World')
```

`'Hello'`, `'World'` are called **string literals**.

`print(...)`

```
name = 'Mark'
print(name)
```

- You can print **string literals** OR values contained in variables.
- What is the variable in this example?

```
print('Hello', end='')
print('World', end='')
```

> HelloWorld

- Note there is nothing between the open and close single quote

Can you do it?

Use print statements to draw a diamond shape.

Use print statements to draw a heart shape.

New Stuff

Printing On Different Lines

- Options:
  - use separate `print()` statements
  - embed `\n` in string literal

```
print('Hello')
print('World')
print('Hello\nWorld')
```


Printing Variable vs String Literal

```
roomNumber = 123
print('Our room number is', end=' ')
print(roomNumber, end=' ')
print('in Fondren.')
```

> Our room number is 123 in Fondren.

Printing Multiple things with `print(...)`

```
age = 19
print('I\'m currently', age, 'years old.')
age = age + 1
print('On my next birthday, I will be', age, 'years old.')
```

> I'm currently 19 years old.
> On my next birthday, I will be 20 years old.

Input

Basic Keyboard Input

- use the `input()` function to read from the keyboard (sometimes called `standard input`)
- `input()` returns a string (sequence of characters) of whatever the user types

```
print('What\'s your favorite class?')
className = input()
print(className, 'sounds like a fun class to me!')
```


Is everything a string?

- What if the user is entering a number that you want to add 1 to?

```
print('How old are you?')
age = input()
age = age + 1
print('Next year, you\'ll be ', age)
```

... results in ...

```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    age = age + 1
TypeError: can only concatenate str (not "int") to str
```

Data Types

- Every 'thing' in a Python program has a **Data Type**
- You can think of it as **metadata** describing what operations I can perform on it

```
someVar1 = '123'
someVar2 = 123
```

- `'123'` is a string
  - You can't perform mathematical ops on a string... doesn't make any sense.
- `123` is an integer

`input()` and Type Conversion

- You can use use the `int(...)` function to convert from string to integer.
  - Technical term: **casting**

```
age = input('How old are you? ')
print(type(age))
```

- *Note the alternative way of calling the* `input()` *function*
- `type(...)` will tell you the data type of the thing in parens.
- If you run this, even if you enter an integer for age, the type will be `'str'`

More on Type

```
var01 = 123
var02 = 'Mark'
var03 = 3.1415
print(type(var01))
print(type(var02))
print(type(var03))
```

Output:

```
<class 'int'>
<class 'str'>
<class 'float'>
```


Errors

Types of Errors

1. Syntax Errors
2. Runtime Errors
3. Logic Errors

Syntax Errors

- A **syntax error** is a violation of rules of how one can use the symbols of the language to construct a program.
- Examples:
  - improperly nested parentheses
    - `print(type('abc)`
  - putting two lines of code on one typed line
    - `print('Hello') print('World')`
- Syntax Errors of a program are caught before any lines of code are actually executed

Runtime Errors

- A **runtime error** is one in which the programmer attempts an impossible operation
- Examples:
  - mathematical operations on a string
    - `'123' + 1`
  - division by zero
    - `25 / 0`
- Runtime Errors are caught when the particular illegal instruction is executed
- You'll get an error message from Python usually with information about where the error is.
  - source file name
  - line number

Logic Errors

- A logic error is an error in the instructions in the program even though the instructions are used correctly according to the rules of Python.
- Examples:
  - You multiply two numbers when you really need to add two numbers

```
test1 = 90
test2 = 90
avg = test1 + test2 / 2
print (avg)
```

```
135.0
```