# CS 1340:Fall 2020:Lecture 04

Intro to Python for CS and Data Science

Mark Fontenot, PhD

Southern Methodist University



## Highlights from Remainder of Ch 1

## Whitespace

- Programming requires precision in the way you "say" things to Python
- Code Formatting can be:
  - required by the Python language
  - suggested by convention of the Python language
- For some activities, ZyBooks will be strict about output formatting as well
  - yes, including whitespace
- Whitespace comes in two forms:
  - vertical (mentioned this last class)
  - horizontal



## Example

```
# Vertical WhiteSpace
hourly_wage = 22
hours_week_01 = 12
hours_week_02 = 15
hours_week_03 = 11

pre_tax = (hours_week_01 + hours_week_02 + hours_week_03) * hourly_wage
print('Before taxes, you earned', pre_tax)
```

```
Before taxes, you earned 836
```

## Precision precision precision!

- You've got to pay attention to detail
  - *yes... even if you aren't a **detail** person*
- If things look very similar to you,
  - then they are still VERY different to the computer
  - Examples:
    - = vs ==
    - counting from 1 to 10 and counting from 0 to 10 are very different to a computer



## The Python Documentation Tour

Python 3 Official Documentation

## New Stuff

## Data Types

- Numeric
  - integer
  - floating point number (number with a fractional component)
- String
- List
- Dictionary
- Set
- Tuple

## 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'>
```



## 3 Fundamental Constructs in all Programming

1. Sequential Execution (What you've been doing so far)
2. Conditional Execution
3. Repetitive Execution

## Conditional Execution

- only execute a block of code **if some condition is true**.
- Conditional Execution is sometimes called **branching**

```
if some_condition:
    statement1
    statement2
    ...
elif some_other_condition:
    statement3
    statement4
    ...
else:
    statement5
    statement6
```



## Conditional Example (a.k.a. if statement)

```
final_grade = 93

if final_grade >= 97:
    print('You earned an A+!')
elif final_grade >= 93:
    print('You earned an A!')
elif final_grade >= 90:
    print('You earned an A-!')
else:
    print('Better luck next time!')
```

```
You earned an A!
```

- **condition** - a **test** that is either true or false
- **relational operators** -
  - >, <, >=, <=, == <– work with numerical data