# Conditional logic and boolean

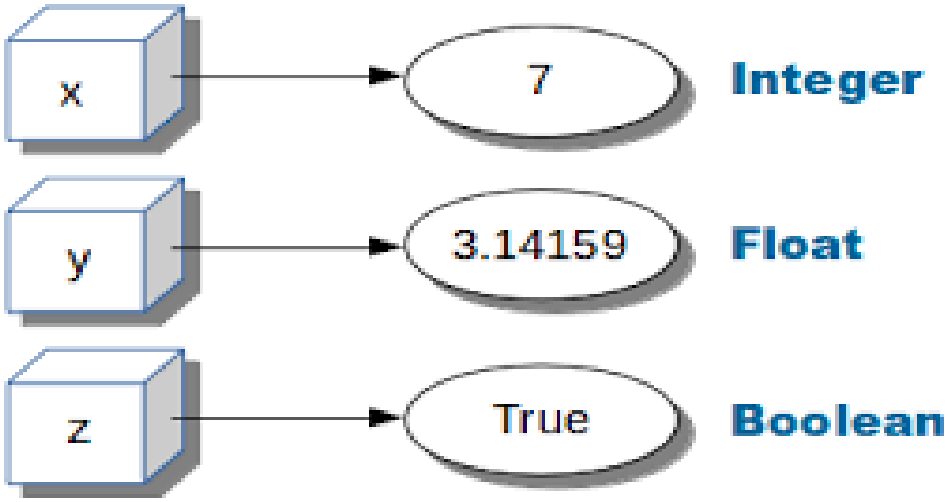
# Objective

- Learn about Boolean
- Learn how to use comparison operators
- Learn how to use Conditional statments

# Types

Variables and expression

- ints
- floats
- bool
- string
- Others we will see later



# Strings

- Letters, special characters, spaces, digits
- enclose in quotation marks or single quotes

  e.g greeting = “hello456%&123,.:”
- concatenate strings

  e.g name = "farhan"

  greet = hi + name

  greeting = hi + " " + name

# Operations on Strings

- ‘ab’ + ‘cd’ → Concatenation
- ‘farhan’[0] → ‘f’ → Indexing
- ‘hello’[1:3] → slicing

# Input/Output:

- print() → used to output stuff to console
- e.g

```
x = 1
print(x)
x_str = str(x)
print("value of x is: " + x_str)
```

# Input/Output:

- input(“”) → used to take input from user
- prints whatever is within the quotes
- user types in something and hits enter
- returns entered sequence
- can bind that value to a variable so can reference

  e.g
```
text = input("Type anything... ")
print(text)
```
- input returns a string so must cast/convert if working with numbers

  e.g
```
num = int(input("Type a number... "))
print(num)
```

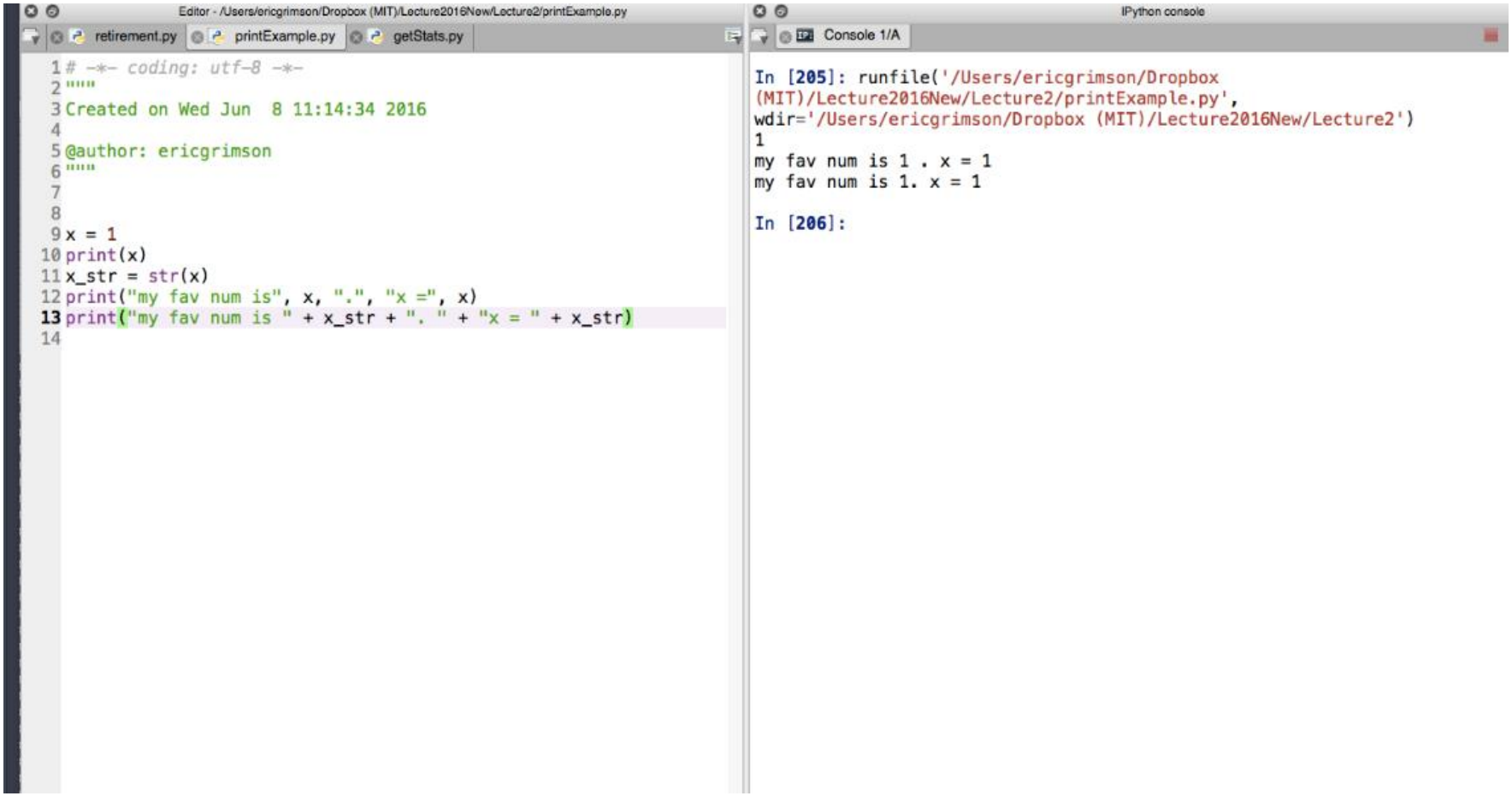
# IDE's

- **better to have a text editor – integrated development environment (IDE)**

  e.g IDLE or Anaconda are examples

- **comes with**
  - **Text editor – use to enter, edit and save your programs**
  - **Shell – place in which to interact with and run your programs; standard methods to evaluate your programs from the editor or from stored files**
  - **Integrated debugger (we'll use later)**

# IDE's


Editor - /Users/ericgrimson/Dropbox (MIT)/Lecture2016New/Lecture2/printExample.py
retirement.py
printExample.py
getStats.py
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Jun 8 11:14:34 2016
4
5 @author: ericgrimson
6 """
7
8
9 x = 1
10 print(x)
11 x_str = str(x)
12 print("my fav num is", x, ".", "x =", x)
13 print("my fav num is " + x_str + ". " + "x = " + x_str)
14
IPython console
Console 1/A
In [205]: runfile('/Users/ericgrimson/Dropbox
(MIT)/Lecture2016New/Lecture2/printExample.py',
wdir='/Users/ericgrimson/Dropbox (MIT)/Lecture2016New/Lecture2')
1
my fav num is 1 . x = 1
my fav num is 1. x = 1
In [206]:

# Comparsion Operators

- i and j are any variable names

  i>j

  i>=j

  i<j

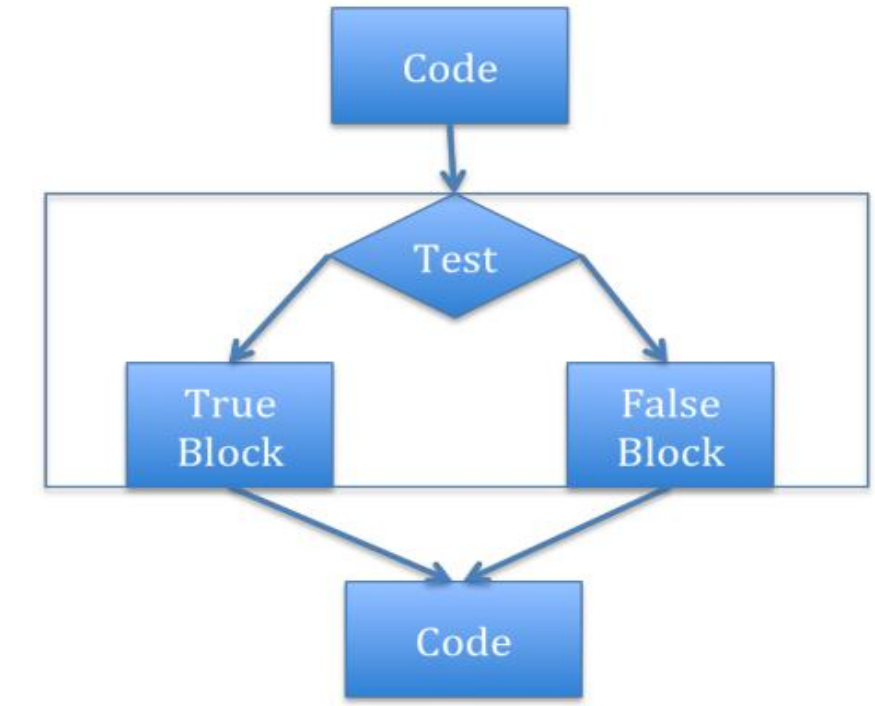
  i<=j

  i == j equality test, True if i equals j

  i != j inequality test, True if i not equal to j

# BRANCHING PROGRAMS

- Decisions bases of something begin True/ False
- The simplest branching statement is a conditional
  - A test (expression that evaluates to True or False)
  - A block of code to execute if the test is True
  - An optional block of code to execute if the test is False


Code
Test
True
Block
False
Block
Code

# A Simple Example

```
number = int(input("Give Some Number "))

if number == 2:
    print("")
    print("Equal to 2")

print("")
print("starts from here")
```

# A Simple Example

```
number = int(input("Give Some Number"))

if number % 2 == 0:
    print('')
    print('Even')
else:
    print('')
    print('Odd')

print('Done with Conditional Logic')
```

# Some obervations

- The expression x%2 == 0 evaluates to True when the remainder of x divided by 2 is 0
- Note that == is used for comparison, since = is reserved for assignment
- The indentation is important – each indented set of
- expressions denotes a block of instructions
  - For example, if the last statement were indented, it would be executed as part of the else block of code
- Note how this indentation provides a visual structure that reflects the semantic structure of the program

# Nested If Condition

```
number = int(input("Give Some Number "))

if number % 2 == 0:
    if number % 3 == 0:
        print("Number is divisible by 2 and 3")
    else:
        print("Divisible by 2 and not by 3")
elif number % 3 == 0:
    print("Divisible by 3 and not by 2")

print("")
print('Done with Conditional Logic')
```

# LOGIC OPERATORS ON bools

- a and b are any variable names

  not a → True if a is False

  False if a is True

  a and b → True if both are True

  a or b → True if either or both are True

# Table

| A | B | A AND B | A OR B | NOT A |
|---|---|---|---|---|
| False | False | False | False | True |
| False | True | False | True | True |
| True | False | False | True | False |
| True | True | True | True | False |

# Compound Boolean

```
x = int(input("Give value of x: "))
y = int(input("Give value of y: "))
z = int(input("Give value of z: "))

if x < y and y < z:
    print("x is least")
elif y < z:
    print("y is least")
else:
    print("z is least")
```

# Branching programs

```
if <condition>:
    <expression>
    <expression>
    ...
```

```
if <condition>:
    <expression>
    <expression>
    ...
else:
    <expression>
    <expression>
    ...
```

```
if <condition>:
    <expression>
    <expression>
    ...
elif <condition>:
    <expression>
    <expression>
    ...
else:
    <expression>
    <expression>
    ...
```

- <condition> has a value True or False
- evaluate expressions in that block if <condition> is True

# Indentation

- Matters in python
- how you denote blocks of code