

---

---

# Week 3

Intro Python

---

---

# Objectives

## Chapter 4

- Generate random numbers using the random functions
  - Boolean expressions using comparison operators
  - Logical operators (and, or, not)
  - Nested if statements
  - Multi-way if-elif-else statements
-

# Intro to Boolean Expressions

A program can decide which statements to execute based on a condition.

For example, if we have an input statement asking the user to enter the radius of a circle, we can give an error statement if they enter a negative value.

```
import math

radius = eval(input("Enter radius: "))

if radius < 0:
    print("Incorrect input")
else:
    area = math.pi*(radius**2)
    print("Area of radius:", area)
```

The selection statement uses a condition, which is a boolean expression

# if Statements

An *if* statement executes statements if the condition is true.

An *else* statement follows an *if* statement and executes if the condition is false. You do not always need an *else* statement.

```
if expression:  
    print(statement)
```

# If-else Statements

What if we want something to happen if the expression is false?

An if statement takes action only if the expression is true.

```
if expression:  
    # if True  
    print(statement)  
else:  
    # if expression is False  
    print(statement2)
```

# Nested if Statements

One *if* statement can be placed inside another *if* statement to form a nested *if* statement.

Nested *if* statements are used to implement multiple alternatives or check multiple conditions.

The following example compares the numbers *i* and *j* to *k*.

```
if i > k:
    if j > k:
        print("i and j are greater than k")
    else:
        if j < k:
            print("i and j are less than k")
        else:
            print("i is less than k", end=" ")
            print("but j is greater than k")
```

# if-elif-else Statements

*elif* allows for implementation of multiple alternatives without nesting.

```
if score >= 90.0:
    grade = 'A'
else:
    if score >= 80.0:
        grade = 'B'
    else:
        if score >= 70.0:
            grade = 'C'
        else:
            if score >= 60.0:
                grade = 'D'
            else:
                grade = 'F'
```

Equivalent

This is better

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

# Logical Operators

Logical operators create composite conditions based on boolean expressions and they use boolean values (True or False)

And - true only if both expressions are true

Or - true if at least one expression is true

Not - inverts boolean value (True goes to False, False goes to True)

```
a = True  
b = False
```

```
a and b  
>>> False
```

```
a or b  
>>> True
```

```
a and a  
>>> True
```

```
b or b  
>>> False
```

```
b or not b  
>>> True
```

```
b and not b  
>>> False
```



# Comparison Operators

```
print(radius<0)
>>> False
```

<i>Python Operator</i>	<i>Mathematics Symbol</i>	<i>Name</i>	<i>Example (radius is 5)</i>	<i>Result</i>
<	<	less than	<code>radius &lt; 0</code>	<b>False</b>
<=	≤	less than or equal to	<code>radius &lt;= 0</code>	<b>False</b>
>	>	greater than	<code>radius &gt; 0</code>	<b>True</b>
>=	≥	greater than or equal to	<code>radius &gt;= 0</code>	<b>True</b>
==	=	equal to	<code>radius == 0</code>	<b>False</b>
!=	≠	not equal to	<code>radius != 0</code>	<b>True</b>

# Common Errors

Most common errors are caused by incorrect indentation.

```
radius = -20

if radius >= 0:
    area = radius * radius * math.pi
print("The area is", area)
```

(a) Wrong

```
radius = -20

if radius >= 0:
    area = radius * radius * math.pi
    print("The area is", area)
```

(b) Correct

Are both of these correct? If so, which one is better?

```
if age < 16:  
    print("Cannot get a driver's license")  
if age >= 16:  
    print("Can get a driver's license")
```

(a)

```
if age < 16:  
    print("Cannot get a driver's license")  
else:  
    print("Can get a driver's license")
```

(b)

How would you rewrite  
this in one line?

```
if count % 10 == 0:  
    newLine = True  
else:  
    newLine = False
```

# Generate Random Numbers

```
import random
```

*randint(a, b)* generates a random integer between a and b inclusive

*random()* generates a random float  $r$ ,  $0 \leq r < 1$

# Random Numbers

To generate random numbers, we need the module *random*.

The *randint(a, b)* function returns an integer between a and b.

The *random()* function generates a random float between 0 and 1.

```
import random

random.randint(0, 1000)
>>> 212

random.random()
>>> 0.4592567969083148
```

[/github.com/cadtexas/fa17-intro-to-python/](https://github.com/cadtexas/fa17-intro-to-python/)