# Introduction to Python

**Conditionals**

## Topics

1) Conditionals
   a) if, if-if, if-elif, if-elif-else
   b) Ternary operator

## Conditionals

The reserved word **if** begins an conditional block.

```
if condition:
    block
```

The condition determines if the block is to be executed.

A block contains one or more statements.

The statements inside of a block must be indented the same number of spaces from the left. The standard is 4 spaces.

## If block

```
In[1]: x = -5
       if x > 0:
          print(x)
          print(“x is positive”)
       print(“outside of block”)
```

```
outside of block
```

## If block

```
In[2]: x = 5
       if x > 0:
           print(x)
           print(“x is positive”)
       print(“outside of block”)
```

```
5
x is positive
outside of block
```

## Sequence of Ifs

A sequence of consecutive if statements are independent. None, some or all of them can be executed.

```
In[3]: x = 4
       if x % 2 == 0:
          print(“x is even”)
       if x > 0:
          print(“x is positive”)
```

```
x is even
x is positive
```

## if-elif

An if block followed by a sequence of elif blocks will execute the first block whose condition evaluates to True. No block is executed if all conditions evaluate to False.

```
In[3]: x = 25
       if x < 0:
           print(“x is less than 5”)
       elif x < 10:
           print(“x is less than 10”)
       elif x < 15:
           print(“x is less than 15”)
```

Note that all of the above conditions are false and thus no block is executed.

## if-elif

```
In[3]: x = 1
       if x < 5:
           print(“x is less than 5”)
       elif x < 10:
           print(“x is less than 10”)
       elif x < 15:
           print(“x is less than 15”)
```

```
x is less than 5
x is less than 10
x is less than 15
```

## if-elif-else

An `if` statement followed by a sequence of `elif` statements and ending in an `else` statement will execute the first block whose condition evaluates to `True`. If all conditions evaluate to `False`, it will execute the default `else` block.

```
In[3]:  x = 0
       if x < 0:
           print(“x is negative”)
       elif x > 0:
           print(“x is positive”)
       else:
           print(“x is zero”)
   x is zero
```

## and, or, not

Use *and*, *or*, and *not* Boolean operators to simplify conditionals.

The following

```
if x > 0:
    if x < 10:
        print(x)
```

is equivalent to

```
if (x > 0) and (x < 10):
    print(x)
```

## Ternary Operators

A *ternary operator* evaluates an expression based on the value of a boolean condition. This is sometimes called *conditional expression* or an *inline if-else* statement.

The following

```
x = 50
grade = “pass” if x >= 60 else “fail”
print(grade)
```

```
fail
```

## References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.
2) Richard Halterman, Fundamental of Python Programming.