

Introduction to Python

Feature Summary, Session 2

CONDITIONALS / PROGRAM FLOW

- **if** statement

- if the test in this statement is True, enter the block
- if the test in this statement is False, skip to below the block
- when the end of the block is reached, continue to the next line

```
var = 3
if var > 0:
    print('higher')
print('continuing...')
```

```
initialize int
if int is greater than 0:
    print str
```

- **elif** statement

- if all **if** and **elif** tests in the chain above this statement are False, and the test in this statement is True, enter the block
 - if all **if** and **elif** tests in the chain above this statement are True, skip to below the block
 - if the test in this statement is False, skip to below the block
- when the end of the block is reached, continue to the next line

```
var = 3
if var > 0:
    print('higher')
elif var < 0:
    print('lower')
print('continuing...')
```

```
initialize int
if int is greater than 0:
    print str
otherwise if int is less than 0:
    print str
```

- **else** statement

- if all **if** and **elif** tests in the chain above this statement are False, enter the block
- when the end of the block is reached, continue to the next line

```
var = 3
if var > 0:
    print('higher')
elif var < 0:
    print('lower')
```

```

else:
    print('equal')
print('continuing...')

initialize int
if int is greater than 0:
    print str
otherwise if int is less than 0:
    print str
otherwise:
    print

```

- **while** statement

- if the test in this statement is True, enter the block (**while True** is True)
- when the end of the block is reached, return to the top of the block and evaluate the test in this statement again (see above -- **while True** is always True)
- if a **break** statement is reached, drop to the first line after the block
- if a **continue** statement is reached, return to the top of the block and evaluate the test in this statement again (see above -- **while True** is always True)

```

var = 0
while var < 3:
    var = var + 1
    print(var)

initialize int
while int is less than 3:
    increment int
    print int

```

STRING METHODS

(this is a sampling of some of the many string methods available)

this string refers to the string upon which the method was called

substring refers to a string argument used to analyze this string

- **.count(): return int number of occurrences of a substring in this string**

argument: string

return value: integer

```

x = 'hello'
y = x.count('l')
# str -> str count in str -> int
print(y)                                # 2

```

- **.endswith(): return True if this string ends with a substring**

argument: string

return value: True or False (boolean)

```

x = 'hello.'
if x.endswith('.'):
# if str -> str check if endswith -> bool
    print('found a period')

```

- **.find(): return int index of substring within this string**

argument: string

return value: integer

```

x = 'hello'
idx = x.find('l')                                # 2
# str -> str find position in str -> int

```

- **.format():** *insert values into a string template*

argument: any object(s)

return value: string

```
x = '{} is {}'.format('bill',3.91)
# str, int -> str insert into str -> str
print(x)          # bill is 3.91
```

- **.isdigit():** *return True if this string is all digits*

argument: N/A (works with this string)

return value: True or False (boolean)

```
x = '12345'
if x.isdigit():
# str check if isdigit -> bool
print('it is all digits')
```

- **.lower():** *return a copy of this string, uppercased*

argument: N/A (works with this string)

return value: string

```
x = 'Joe'
y = x.lower()
# str lowercase -> str
print(y)                                # joe
```

- **.replace():** *return a copy of this string with substring replaced with another substring*

argument: 2 strings

return value: string

```
x = 'Hello, world!'
y = x.replace('world', 'Mars')
# str, str -> str replace with str -> str
print(y)          # Hello, Mars!
```

- **.startswith():** *return True if this string ends with a substring*

argument: string

return value: True or False (boolean)

```
x = 'quit'
if x.startswith('q'):
# str -> str check if startswith -> bool
    print('looks like quitting')
```

- **.upper():** *return a copy of this string, uppercased*

argument: N/A (works with this string)

return value: string

```
x = 'Joe'
y = x.upper()
# str uppercase -> str
print(y)                                # JOE
```