Columbia Business School
AT THE VERY CENTER OF BUSINESS™
EXECUTIVE EDUCATION

Week 2 - Summary

Columbia Business School

Conditional Statement: if (1/3)

'if' checks if the condition is true and then runs the code after the ':' symbol

code

if.py — ~/Desktop/code

```
answer = input("Do you want to hear a joke? ")

if answer == "Yes":
    print("I'm against picketing, but I don't know how to show it.")
    # Mitch Hedberg (RIP)
```

result(s)

code — -bash — 65×38

```
Last login: Mon Aug 12 14:19:06 on ttys000
(base) Mattans-MacBook-Pro:~ mattan$ cd Desktop
(base) Mattans-MacBook-Pro:Desktop mattan$ cd code
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Yes
I'm against picketing, but I don't know how to show it.
```

'if' condition validates based on case sensitivity too

Columbia Business School

Conditional Statement: if (2/3)

If the condition is not met, Python provides no result

```
Last login: Mon Aug 12 14:19:06 on ttys000
(base) Mattans-MacBook-Pro:~ mattan$ cd Desktop
(base) Mattans-MacBook-Pro:Desktop mattan$ cd code
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Yes
I'm against picketing, but I don't know how to show it.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? No
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? ;lkdsflk;asf
(base) Mattans-MacBook-Pro:code mattan$
```

```
answer = input("Do you want to hear a joke? ")

if answer == "Yes":
    print("I'm against picketing, but I don't know how to
    # Mitch Hedberg (RIP)
```

':" symbol allows you to tab the next line to enter the code

'==' symbol checks to see if the two things on the left and right are equal

Columbia Business School

Conditional Statement: if (3/3)

'=' symbol creates a variable

'!=' symbol checks to see if the answer is not equal to the variable

```
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. c
n
Type "help", "copyright", "credits" or "license" for more
tion.
>>> answer == "Yes"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'answer' is not defined
>>> answer = "Yes"
>>> answer == "Yes"
True
>>> answer = "No"
>>> answer == "Yes"
False
>>> answer != "Blue"
True
>>> answer != "No"
False
```

Variables can be rewritten in Python

Columbia Business School

Conditional Statement: else and elif (1/2)

'else' condition is triggered when the 'if' condition is not met

```
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Yes
I'm against picketing, but I don't know how to show it.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? No
Fine.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? yes
Fine.
(base) Mattans-MacBook-Pro:code mattan$
```

```
answer = input("Do you want to hear a joke? ")

if answer == "Yes":
    print("I'm against picketing, but I don't know how to
    # Mitch Hedberg (RIP)
else:
    print("Fine.")
```

Columbia Business School

Conditional Statement: else and elif (2/2)

'elif' allows you to set a second 'if' condition

```
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Yes
I'm against picketing, but I don't know how to show it.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? No
Fine.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? yes
Fine.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Yes
I'm against picketing, but I don't know how to show it.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? No
Fine.
(base) Mattans-MacBook-Pro:code mattan$ python if.py
Do you want to hear a joke? Blue
I don't understand.
(base) Mattans-MacBook-Pro:code mattan$
```

```
answer = input("Do you want to hear a joke? ")

if answer == "Yes":
    print("I'm against picketing, but I don't know how to
    # Mitch Hedberg (RIP)
elif answer == "No":
    print("Fine.")
else:
    print("I don't understand.")
```

Columbia Business School

Conditional Statements: if, else and elif

'if' is independent of 'elif' and 'else'

All the three conditions have to be in the order of – 'if', 'elif' and 'else'

There can be several 'elif's but only one 'else'

Columbia Business School

Operators in Python

To produce True or False statement

Three additional terms are *not*, *and*, and *or*.

Operator	Description
=	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Columbia Business School

Truth Table: not

Code (not)	Result
not True	False
not False	True

True and False start with capital letters in Python to make them work properly.

Columbia Business School

Truth Table: and

Code (and)	Result
True and True	True
True and False	False
False and True	False
False and False	False

Columbia Business School

Truth Table: or

Code (or)	Result
True and True	True
True and False	True
False and True	True
False and False	False

Columbia Business School

Logical Function: or

Any string value in Python is, by default, true

if.py

```
answer = input("Do you want to hear a joke? ")

if answer == "Yes" or answer == "yes":
    print("I'm against picketing, but I don't know how to
    # Mitch Hedberg (RIP)
elif answer == "No" or answer == "no":
    print("Fine.")
else:
    print("I don't understand.")
```

Columbia Business School

Lists

Lists are a way of grouping similar things together. They start and end with square brackets and have 'elements' in between.

lists.py — ~/Desktop/code

lists.py

```python
# In Python, lists are ways of grouping together
# similar things (usually)
the_count = [1, 2, 3, 4, 5]
stocks = ["FB", "AAPL", "NFLX", "GOOG"]
random_things = [55, 1/2, "Puppies", stocks]

people = []

people.append("Mattan")
people.append("Daniel")
people.append("Sam")
people.remove("Daniel")

print(people)
```

Creating a list

Use '<listname>.append' to add entries to the list and '<listname>.remove' to remove them

Columbia Business School

More Ways to Create Lists

```
print("New York, San Francisco, London".split(", "))
```

```
['New York', 'San Francisco', 'London']
(base) Mattans-MacBook-Pro:code mattan$
```

Splitting a list of items based on specific conditions

```
print("New York, San Francisco, London".split(", "))
print(", ".join(["Milk", "Eggs", "Cheese"]))
```

```
Milk, Eggs, Cheese
(base) Mattans-MacBook-Pro:code mattan$
```

Joining items to make a list

Columbia Business School

Accessing Elements of a List

You can access elements of a list using square brackets [ ]

```
print("New York, San Francisco, London".split(", "))
```

```
first_city = cities[0]
second_city = cities[1]
last_city = cities[-1]
```

The first item of the list is numbered as zero and to access the last item in the list use '-1'

```
first_two_cities = cities[0:2]
print(first_two_cities)
```

```
['New York', 'San Francisco']
(base) Mattans-MacBook-Pro:code mattan$
```

Slice notation refers to accessing parts of a list. Here '0:2' tells Python to show cities from the first until the third one but excluding the last.

Columbia Business School

Loops in Lists (1/2)

- A loop is a faster and more concise way of repeating an action
- Use 'for' to loop over lists

```
(base) Mattans-MacBook-Pro:code mattan$ python loops.py
1
2
3
(base) Mattans-MacBook-Pro:code mattan$ python loops.py
1
2
3
1
2
3
(base) Mattans-MacBook-Pro:code mattan$ python loops.py
1
2
3
1
2
3
```

loops.py

```python
# Use for to loop over a list
numbers = [1, 2, 3]
for number in numbers:
    print(number)

numbers = [1, 2, 3]
number = numbers[0]
print(number)
number = numbers[1]
print(number)
number = numbers[2]
print(number)
```

Columbia Business School

Loops in Lists (2/2)

While looping, you can also change the casing of the items in the list

```
FB
AAPL
NFLX
GOOG
(base) Mattans-MacBook-Pro:code mattan$
```

```
stocks = ["fb", "aapl", "nflx", "goog"]
for stock in stocks:
    print(stock.upper())
```

Columbia Business School

Append Within a For Loop

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 2
56, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 8
41, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 160
0, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 26
01, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3
844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184,
5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889,
 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836
, 9025, 9216, 9409, 9604, 9801, 10000]
```

```python
# You can do more than just print in a for loop
squares = []
for number in range(1, 101):
    squares.append(number**2)
print(squares)
```

Columbia Business School

Creating a List From Another List

List comprehension is when we create a list using an existing list

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 2
56, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 8
41, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 160
0, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 26
01, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721, 3
844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184,
5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889,
 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836
, 9025, 9216, 9409, 9604, 9801, 10000]
['FB', 'AAPL', 'NFLX', 'GOOG']
```

```python
squares = []
for number in range(1, 101):
    squares.append(number**2)
print(squares)

# List comprehension
print([stock.upper() for stock in stocks])
```