Introduction to Python for Economists

Session 2: Coding, conditionals, dictionaries & loops

Roland Mühlenbernd

26. Februar 2020

Comments & the Zen of Python

1. `# Say Hello to everyone`
2. `print("Hello Python people.")`

The Zen of Python

3. `import this`

Output (extracts)

- Simple is better than complex.
- Readability counts.
- Now is better than never.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

The Python Style Guide

- Python Enhancement Proposal (PEP)
- PEP 8: How to style your code
  https://python.org/dev/peps/pep-0008/
- Given choice between code that is easier to write or easier to read, be encouraged to write code that is easier to read
- Some conventions:
  - Set indentation level (TAB) to four spaces
  - Use always TAB and never spaces for indentation
  - Limit line to 79 characters (comments to 72)
  - Use blank lines to group different parts of your program

A first If statement

```
1. cars = ['audi', 'bmw', 'subaru', 'toyota']
2. for car in cars:
3.     if car == 'bmw':
4.         print(car.upper())
5.     else:
6.         print(car.title())
```

Equality test

```
7. car = 'bmw'
8. print(car == 'bmw')
9. print(car == 'audi')
10. print(car == 'BMW')
11. print(car.upper() == 'BMW')
12. print(car)
13. print(car != 'audi')
14. print(car != 'bmw')
```

Numerical Comparisons

1. `age = 17`
2. `print(age == 17)`
3. `driving_age = 18`
4. `print(age == driving_age)`
5. `drinking_age = 16`
6. `print(age == drinking_age)`
7. `print(age >= driving_age)`
8. `print(age >= drinking_age)`
9. `print(age < driving_age)`

Checking Multiple Conditions

10. `print(age >= driving_age and age >= drinking_age)`
11. `print(age < driving_age and age >= drinking_age)`
12. `print(age >= driving_age or age >= drinking_age)`
13. `print((age < driving_age) or (age >= drinking_age))`

Checking Whether value is in a List

```
1. my_cars = ['audi', 'bmw', 'subaru', 'toyota']
2. print('bmw' in my_cars)
3. print('volkswagen' in my_cars)
4. print('volkswagen' not in my_cars)
5. my_cars.append('volkswagen')
6. print('volkswagen' in my_cars)
7. my_new_car = 'mercedes'
8. if my_new_car not in my_cars:
9.     my_cars.append(my_new_car)
10.    print(f"{my_new_car.title()} is added...")
```

Boolean and If-else statements

```
1. condition_fulfilled = True
2. if condition_fulfilled:
3.     print("You did it!")
4. score = 48
5. condition_fulfilled = (score >= 50)
6. if condition_fulfilled:
7.     print("You did it!")
8. else:
9.     print("You failed...")
```

If-elif-else statements

```
10. score = 50
11. if score >= 60:
12.     print("You did it terrifically!")
13. elif score >= 50 and score < 60:
14.     print("You only just did it.")
15. else:
16.     print("You failed...")
```

Change the score to 48 (88) and run the program again.

Using multiple elif blocks

```
10. score = 50
11. if score >= 60:
12.     print("You did it terrifically!")
13. elif score >= 50 and score < 60:
14.     print("You only just did it.")
15. elif score >= 10 and score < 50:
16.     print("You failed...")
17. else:
18.     print("You hit a new all-time low!")
```

Testing multiple conditions

```
19. my_cars = ['audi', 'bmw', 'subaru', 'toyota']
20. if 'audi' in my_cars:
21.     print("I have an Audi!")
22. if 'bmw' in my_cars:
23.     print("I have a BMW!")
24. if 'toyota' in my_cars:
25.     print("I have a Toyota!")
```

Change the last two if's to elif's. What happens now?

Using multiple lists

```
26. my_cars = ['audi', 'bmw', 'subaru', 'toyota']
27. requested_cars = ['audi', 'bmw', 'toyota', 'vw']
28. for car in requested_cars:
29.     if car in my_cars:
30.         print(f"I can deliver the {car.title()}")
31.     else:
32.         print(f"I can't deliver the {car.title()}")
```

The output is not perfect. Why?

Exercise IV

- Correct the code from the last slide by checking for the car name is an abbreviation; by introducing a new variable `printed_car_name`
- check if 'mercedes' is in `requested_cars`, and if not, add it
- create a new empty list `sold_cars` and move all requested cars that are available from `requested_cars` to `sold_cars`
- run again through the list `my_cars` and print out for each car if it is still requested, already sold, or none of both
- print out if the length of the list of requested cars and sold cars is the same, or which one contains more items
- add comments before each block of your code

A first Dictionary

1. `my_car = {'type': 'vw', 'color': 'red', 'top': 190}`
2. `print(my_car['type'])` ... (print the values of all keys)
3. `my_top_speed = my_car['top']`
4. `my_car['color'] = 'yellow'`
5. `print(my_car)`
6. `my_car['model'] = 'lupo'`
7. `print(my_car)`
8. `my_car['top'] = my_car['top']+5`
9. `print(my_car)`
10. `del my_car['color']`
11. `print(my_car)`

A second Dictionary

```
favorite_languages = {
    'jen' : 'python',
    'ben' : 'java',
    'max' : 'c',
    'kai' : 'python',
    }

print(f"Kai likes {favorite_languages['kai'].title()}")

print(f"Tim likes {favorite_languages['tim'].title()}")
```

A second Dictionary

```
1. favorite_languages = {
2.     'jen' : 'python',
3.     'ben' : 'java',
4.     'max' : 'c',
5.     'kai' : 'python',
6.     }
7. name = 'ben'
8. output = favorite_languages.get(name, "User not in DB.")
9. print(output.title())
```

Replace 'ben' in line 7 with tim. What happens?
Remove the second argument of the get function in line 8. What happens?

Looping through a Dictionary

```
1. favorite_languages = {
2.     'jen' : 'python',
3.     'ben' : 'java',
4.     'max' : 'c',
5.     'kai' : 'python',
6.     }
7. for key, value in favorite_languages.items():
8.     print(f"\nName: {key.title()}")
9.     print(f"Language: {value.title()}")
```

Replace key and value with x and y (in lines 7-9). What happens?

Looping through keys in a Dictionary

```
1. favorite_languages = {
2.     'jen' : 'python',
3.     'ben' : 'java',
4.     'max' : 'c',
5.     'kai' : 'python',
6.     }
7. for name in favorite_languages.keys():
8.     print(f"The name {name.title()} is in the DB.")
```

Remove the '.keys()' method in line 7. What happens?

Looping through keys in a Dictionary

```
1. favorite_languages = {
2.     'jen' : 'python',
3.     'ben' : 'java',
4.     'max' : 'c',
5.     'kai' : 'python',
6.     }
7. names = ['jen', 'tim', 'max']
8. for name in favorite_languages.keys():
9.     print(f"Hi {name.title()}.")
10.    if name in names:
11.        language= favorite_languages[name].title()
12.        print(f"\tI see you love {language}.")
```

Looping through values in a Dictionary

```
1. favorite_languages = {
2.     'jen' : 'python',
3.     'ben' : 'java',
4.     'max' : 'c',
5.     'kai' : 'python',
6.     }
7. for v in favorite_languages.values():
8.     print(f"The language {v.title()} is in the DB.")
9. for v in set(favorite_languages.values()):
10.    print(f"The language {v.title()} is in the DB.")
```

Nesting: Dictionaries in a List

```
1. car_0 = {'type': 'vw', 'color': 'red', 'top': 190}
2. car_1 = {'type': 'bmw', 'color': 'blue', 'top': 270}
3. car_2 = {'type': 'audi', 'color': 'gray', 'top': 240}
4. cars = [car_0, car_1, car_2]
5. for car in cars:
6.     print(car)
```

Nesting: Lists in a Dictionary

```
1. car_0 = {'type': 'vw', 'color': 'red', 'top': 190}
2. car_1 = {'type': 'bmw', 'color': 'blue', 'top': 270}
3. car_2 = {'type': 'audi', 'color': 'gray', 'top': 240}
4. cars = [car_0, car_1, car_2]
5. car_0['features'] = ['abs']
6. car_1['features'] = ['abs', 'navigation system']
7. car_2['features'] = ['abs', 'snow tire']
8. for car in cars:
9.     print(car)
```

Nesting: Lists in a Dictionary

```
1. favorite_languages = {
2.     'jen' : ['python', 'ruby'],
3.     'ben' : ['java'],
4.     'max' : ['c', 'haskell'],
5.     'kai' : ['python'],
6.     }
```

Note: Try to avoid to nest too deeply! Most likely a simpler and more comprehensible way to solve the problem exists!

Nesting: Dictionaries in a Dictionary

```
1.  users = {
2.      'aeinstein' : {
3.          'first' : 'albert',
4.          'last' : 'einstein',
5.          'location' : 'princeton',
6.          },
7.      'mcurie' : {
8.          'first' : 'marie,
9.          'last' : 'curie',
10.         'location' : 'paris',
11.         },
12.     }
```

Exercise V

- open **slides022324_dict3.py**
- create car_3 as a black Ford with top speed 195 and the features abs, snow tire, seat heating; and add it to the list cars
- create the lists car_features low_speeds and high_speeds
- write a double for loop that runs through each car's features and print
- in the loop body, add each car's features to the list car_features, whereby avoid to add items more than one time
- add each car's speed value to the list low_speeds if it is below 200, else to the list high_speeds
- open **slides022626_dict4.py**
- add a new user cdarwin with the name Charles Darwin located in London and the new key 'research' with the value 'evolution'
- loop through the dictionaries and check for the key research - if it is not given, add it with the value 'unknown'

Introducing while loops

1. `current_number = 0`
2. `while current_number < 10:`
3. `    current_number += 1`
4. `    print(current_number)`

Note: ‘a += 1’ stands for ‘a = a + 1’

The continue statement

```
1. current_number = 0
2. while current_number < 10:
3.     current_number += 1
4.     if current_number % 2 == 0:
5.         continue
6.     print(current_number)
```

Incidental remark: The % (modulo) operator returns the remainder of an integer division.

Infinite while loops

1. `current_number = 0`
2. `while current_number < 10:`
3. `    print(current_number)`

Important: Every programmer writes an infinite while loop from time to time. Press CTRL-C to stop it. More importantly, set the break conditions with caution to avoid infinite while loops.

While loops with lists and dictionaries

```
# Start with users that need to be verified,
# and an empty list to add confirmed users
unconfirmed_users = ['alice', 'ben', 'conny']
confirmed_users = [ ]
# Verify each user until no one is unconfirmed.
# Move each verified user into confirmed user list.
while unconfirmed_users:
    current_user = unconfirmed_users.pop()
    print(f"Verifying user: {current_user.title()}")
    confirmed_users.append(current_user)
```

Important: When you modify a list while looping through it, don't use the 'for loop', since Python will have trouble keeping track of the items.

Removing all instances from a list

1.
```
users = ['alice', 'ben', 'conny', 'ben', 'ben']
```
2.
```
users.remove('ben')
```
3.
```
print(users)
```
4.
```
while 'ben' in users:
```
5.
```
    users.remove('ben')
```
6.
```
print(users)
```

Exercise VI

- open **slides021722_dict2.py**
- create a list of people with Jen, Freddy, Max and Steve
- while-loop through the list and print out a message that (i) either tells the favorite languages of the person, (ii) or remarks that the person is not in the dictionary favorite_languages and: remove the person from the list and add them to the dictionary with ‘unknown’ as value
- loop through the dictionary favorite_languages until you have found ‘java’ in one of the language lists and return a message that you found it, otherwise return a message that you didn’t