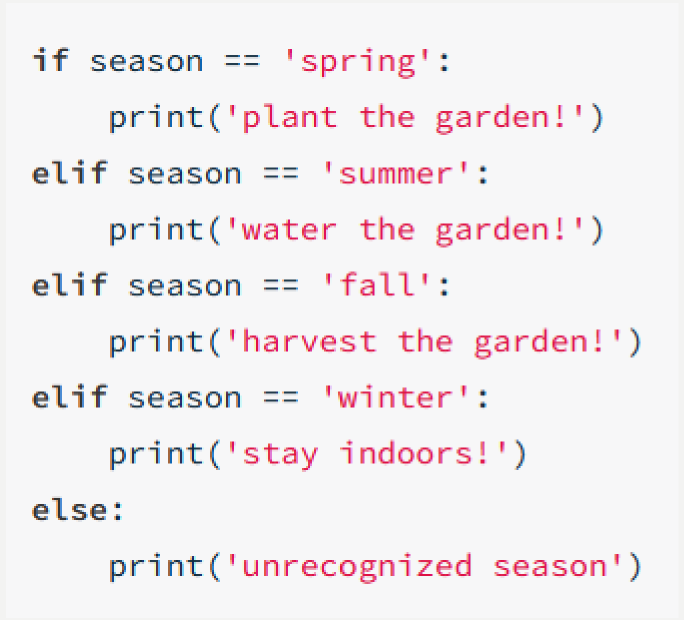
WEEK 4

CONDITIONAL STATEMENTS & LOOPS

IF STATEMENTS

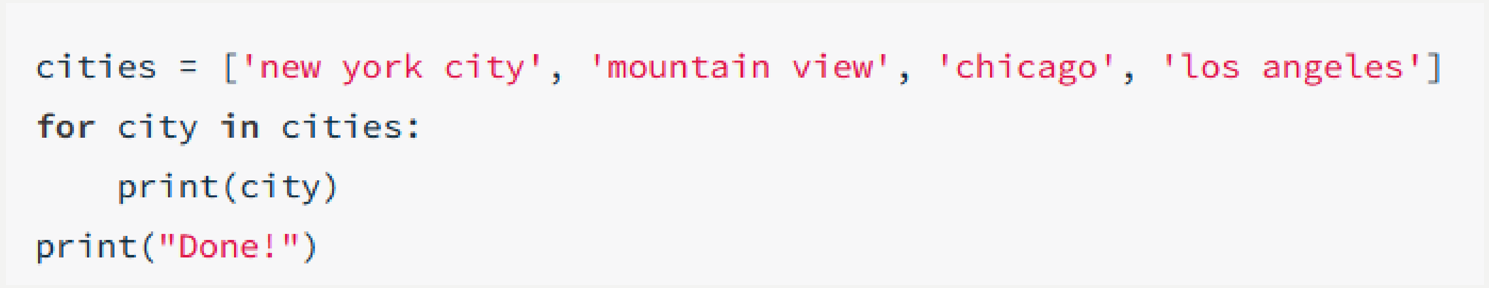
- Act according to a condition
- Take care of indention (4 spaces/Tab)
- Check the conditions from top to botton
- Stop if a condition is met



```python
if season == 'spring':
    print('plant the garden!')
elif season == 'summer':
    print('water the garden!')
elif season == 'fall':
    print('harvest the garden!')
elif season == 'winter':
    print('stay indoors!')
else:
    print('unrecognized season')
```

FOR STATMENTS

- Use for if you know the number of iterations or you want to iterate over a pre-defined set of objects
- Use range(N) to repeat the loop for N-iterations
  - range(start, stop, step)
  - range(start, stop)
  - range(stop)



```
cities = ['new york city', 'mountain view', 'chicago', 'los angeles']
for city in cities:
    print(city)
print("Done!")
```

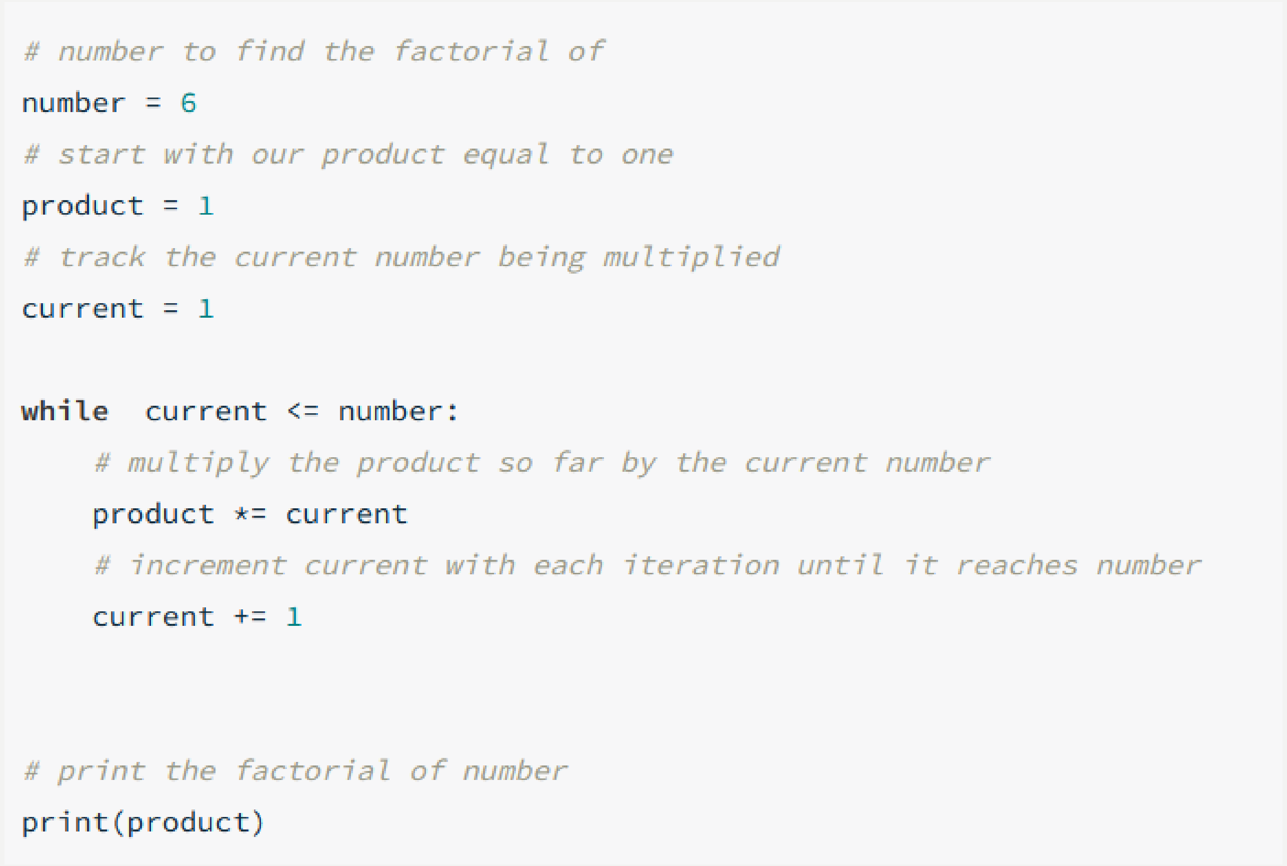
ITERATING OVER DICTS

```
for key, value in cast.items():
    print("Actor: {}    Role: {}".format(key, value))
```

```
for key in cast:
    print(key)
```

WHILE STATEMENT

- Iterate till a condition is met



```
# number to find the factorial of
number = 6
# start with our product equal to one
product = 1
# track the current number being multiplied
current = 1

while  current <= number:
    # multiply the product so far by the current number
    product *= current
    # increment current with each iteration until it reaches number
    current += 1


# print the factorial of number
print(product)
```

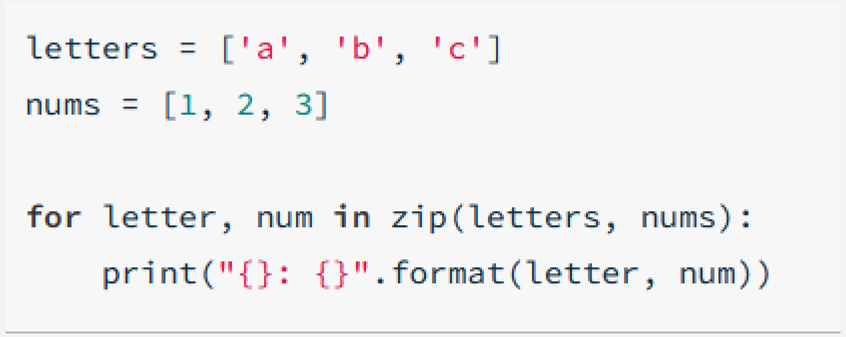
BREAK & CONTINUE

- Break: breaks a loop
- Continue: skip a single iteration of loop

GUSS THE NUMBER

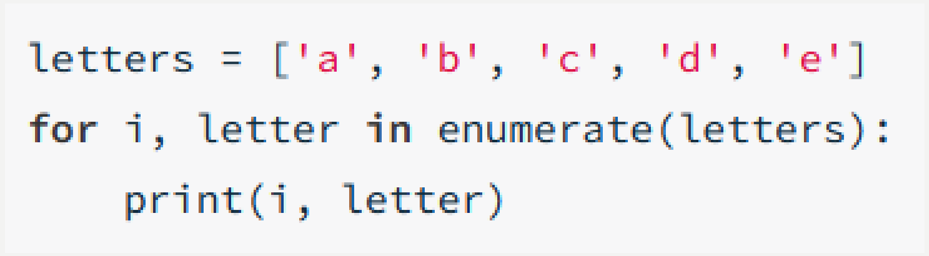
- Select a random value between 1, 100
- Ask user to guess the number
- If the guess is correct print
  - „Congrats! You win“
- If the guess is incorrect print
  - „Try again! You have <x> more choices“
  - x is the number of remaining guesses
  - The maximum of number of guesses is 6
- After 6 incorrect guesses print
  - „You lost!“

ZIP & ENUMERATE



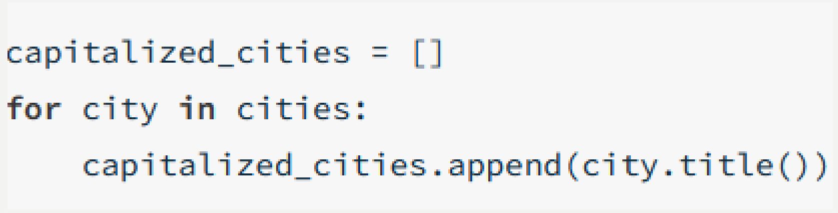
```python
letters = ['a', 'b', 'c']
nums = [1, 2, 3]

for letter, num in zip(letters, nums):
    print("{}: {}".format(letter, num))
```

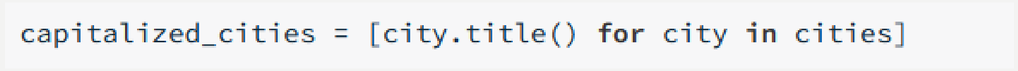


```python
letters = ['a', 'b', 'c', 'd', 'e']
for i, letter in enumerate(letters):
    print(i, letter)
```

LIST COMPREHENSIONS



```python
capitalized_cities = []
for city in cities:
    capitalized_cities.append(city.title())
```



```python
capitalized_cities = [city.title() for city in cities]
```

DOOZ 2.0

- Modifiy the previous version of dooz to:
  - Play till there is a winner
  - Check the validity of user inputs
    - If the cell is already ticked or is out of range, ask user to re-enter the value
  - Check if there is a winner, if yes print „User<x> wone the game!“ and exit the program
  - After getting valid inputs from user print out the dooztable