Python Loops

Petr Svarny, 2020

Mutable versus immutable data types

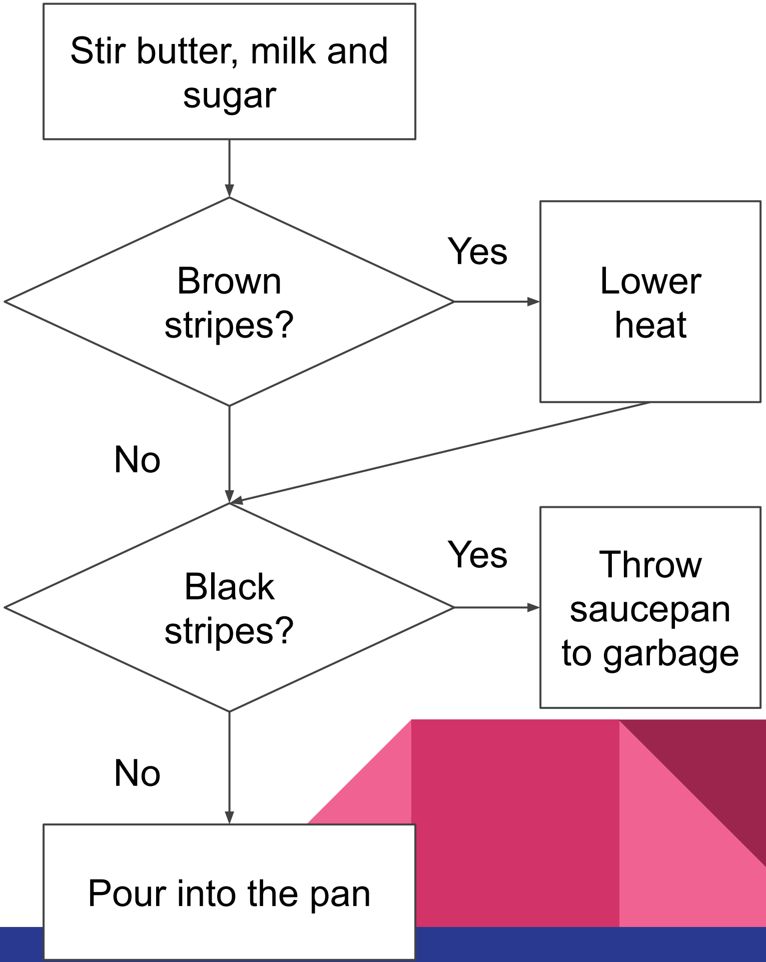
- str, float, int and tuple are immutable data type
- Immutable data types are hashable - it has a hash value which never changes during its lifetime
- list, dict and set are mutable data types

```
>>> x = 'a'
>>> id(x)
...608
>>> x = x + 'b'
>>> id(x)
...888
```

```
>>> y = ['a', 'b']
>>> id(y)
...424
>>> y = y + ['c']
>>> id(y)
...320
>>> y.append('d')
>>> id(y)
...320
```

Conditions

BakeThenEat.com
Traditional Homemade Scottish Tablet
An easy to make, melt in the mouth medium hard candy, perfect for gift giving - or not!


Stir butter, milk and sugar
Brown stripes?
Yes
Lower heat
No
Black stripes?
Yes
Throw saucepan to garbage
No
Pour into the pan

Conditions

- Multiple ***if*** blocks are possible
- ***elif*** and ***else*** blocks are optional
- ***else*** block can be only one and must be last

```
if condition:
    block body
    block body
elif condition:
    block body
elif condition:
    block body
else:
    block body
```

Indentation

- Is used in python to create code blocks
- Indent by **spaces** or tabs
  - Depends on code editor
  - **Replace tab by spaces**
  - **DO NOT MIX**

Python

```
if condition:
    block body
else:
    block body
```

C ++

```
 if(condition) {
   block body
} else {
   block body
}
```

Indentation

```
>>> x = 1
>>> if x > 2:
...     print('x is more than 2')
... print('I have no idea what value does x have')
I have no idea what value does x have

>>> if x > 2:
...     print('x is more than 2')
...     print('I have no idea what value does x have')
... print('seriously, I don't know')
seriously, I don't know
```

Indentation

```
>>> if 4 < 5:
... print('It is smaller')
File "<ipython-input-37-38507da79ee2>", line 2
print('It is smaller')
^
IndentationError: expected an indented block
```

Conditions

```
x = int(input('Type number: ')) # input function reads a line from
input (keyboard) and converts the line into a string

# int function converts number to integer

>>> if x % 2 == 0:
...     print(x, 'is even')
... else:
...     print(x, 'is odd')

Type number: 5
5 is odd
```

Conditions

```
>>> x = int(input("Type number: "))
... if x > 5:
...     print('x is more than 5!')
... if x == 2:
...     print('hmm, x is 2')
... if x < 5:
...     print('x is less than 5!')
```

```
Type number: 2
hmm, x is 2
x is less than 5!
```

Conditions

```
>>> x = int(input("Type number: "))
... if x > 5:
...     print('x is more than 5!')
... if x == 2:
...     print('hmm, x is 2')
... elif x < 5:
...     print('x is less than 5!')

Type number: 2
hmm, x is 2
```

Conditions

```
>>> x = int(input("Type number: "))
... if x > 5:
...     print('x is more than 5!')
... if x == 2:
...     print('hmm, x is 2')
...     if x < 5:
...         print('x is less than 5!')

Type number: 2
hmm, x is 2
x is less than 5!
```

Exercise

- You have list of top 20 names in Czech Republic
- `names_list = ['Jiri', 'Jan', 'Marie', 'Petr', 'Jana', 'Josef', 'Pavel', 'Martin', 'Jaroslav', 'Tomas', 'Eva', 'Miroslav', 'Hana', 'Anna', 'Zdenek', 'Frantisek', 'Vaclav', 'Michal', 'Lenka', 'Katerina']`

- Write code that
    - Ask user for its name (reminder use: **`input(`**`‘Your name’`**`)`**`)`
    - Check if name is in the list (reminder use: **`in`**)
        - If name is in the list then it prints reply
        - If name is not in the list then it prints another reply

Range

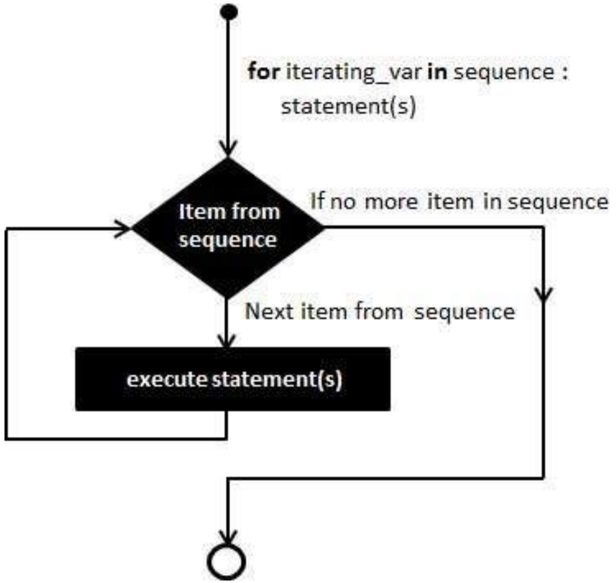
- ***range*** function returns an immutable sequence object of numbers that can be used for loops
  - range(stop) or range([start], stop[, step])
  - Default settings: range(0, stop, 1)

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(0,10,1))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(0,10,2))
[0, 2, 4, 6, 8]
```

Loops - for

- Is used for repeated steps
- For statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence

```
for i in iterated_object:
      block body
      block body
```



https://www.tutorialspoint.com/python/python_for_loop.htm

Loops - for

```
>>> word = 'python'
>>> for letter in word:
...     print(letter)

p
y
t
h
o
n
```

Loops - for

```
>>> fruits = ['apples', 'pears', 'apricots', 'peaches', 'oranges']

>>> for fruit in fruits:
...     print('I like ' + fruit)
I like apples
I like pears
I like apricots
I like peaches
I like oranges
```

Loops - for

```
>>> fruits = ['apples', 'pears', 'apricots', 'peaches', 'oranges']

>>> for i in range(len(fruits)):
...     print('I like ' + fruits[i])
I like apples
I like pears
I like apricots
I like peaches
I like oranges
```

Loops - enumerate

```
>>> fruits = ['apples', 'pears', 'apricots', 'peaches', 'oranges']

>>> for i, fruit in enumerate(fruits):
...     print('I like ' + fruit)
...     print('I like ' + fruits[i])
I like apples
I like apples
I like pears
I like pears
I like apricots
I like apricots
I like peaches
I like peaches
I like oranges...
```

Dictionary iteration

```
>>> kids = {'Sedlak': 'David', 'Iohanescu': 'Julie'}

>>> for key, value in kids.items():
...     print(value, key)
David Sedlák
Julie Iohanescu
```

Exercise

- You have international spelling alphabet
    - d = {'a':'alfa', 'b':'bravo', 'c':'charlie', 'd':'delta', 'e':'echo', 'f':'foxtrot', 'g':'golf', 'h':'hotel', 'i':'india', 'j':'juliett', 'k':'kilo', 'l':'lima', 'm':'mike', 'n':'november', 'o':'oscar', 'p':'papa', 'q':'quebec', 'r':'romeo', 's':'sierra', 't':'tango', 'u':'uniform', 'v':'victor', 'w':'whiskey', 'x':'x-ray', 'y':'yankee', 'z':'zulu'}
- Write code that will
    - Ask user name
    - Spell user's name

Exercise

- Transpose following list using both nested loops and list comprehensions

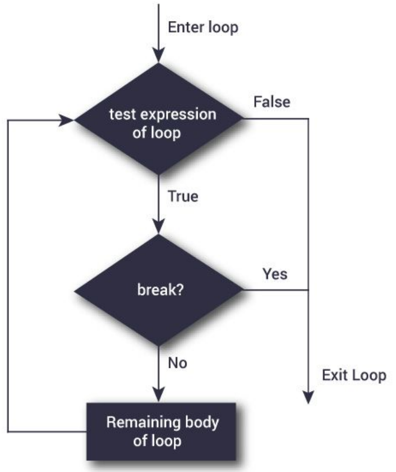
```
a = [[1,2,3],
     [4,5,6],
     [7,8,9]]
```

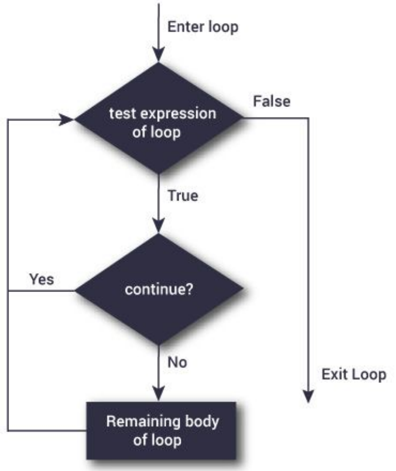
To this list

```
b = [[1,4,7],
     [2,5,8],
     [3,6,9]]
```

Break, continue, pass

- ***break*** terminates the loop containing it
- ***continue*** continues with the next iteration of the loop
- ***pass*** does nothing
  - Is used as a placeholder when you are working on new code, allowing you to keep thinking at a more abstract level





Break

```
for letter in 'Python':
...    if letter == 'h':
...       break
...    print('Current letter is ', letter)
```

```
Current letter is  P
Current letter is  y
Current letter is  t
```

Continue

```
for letter in 'Python':
...     if letter == 'h':
...         continue
...     print('Current letter is ', letter)
```

```
Current letter is  P
Current letter is  y
Current letter is  t
Current letter is  o
Current letter is  n
```

Exercise

- Create shopping list
- Using for and break write code that
    - Will ask for new item
    - Go through the list
    - If item is found then
        - Print item
        - Stop searching
    - If item is not found
        - Append item to the list

Loops - while

- Is needed for executing repeated actions
- **Be careful of infinite loops!**
    - Evaluate if condition in **while** is False or True
    - If True, run block and return to step 1
    - If False, exit **while** loop and continue in code

```
>>> while n > 0:
...     print(n)
...     n = n-1
... print('COOL')
```

Loops - list comprehensions

- Create lists from another lists based on various conditions

```
>>> all_numbers = list(range(15))
>>> all_numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

>>> odd_numbers = [x for x in all_numbers if x % 2 == 1]
>>> odd_numbers
[1, 3, 5, 7, 9, 11, 13]
```

Exercise

- Create list containing 5 numbers
  - Using list comprehensions create list where:
    - Each element is multiplied by itself
      - E.g. 5 → 25
    - 'is my favorite number!' is added to each element of the list'
      - E.g. '5 is my favorite number!'
- Print both lists

Exercise

- Using list comprehensions write code that
  - Takes string as an input, e.g. seq = 'ACTGCTCAAG'
  - Creates list with positions where 'A' is occurring, e.g. pos = [0, 7, 8]
  - Prints created list
  - Hint: use enumerate()
- BONUS task: come up with the second solution

Dictionary comprehensions

- Create dictionaries from another sequences based on various conditions

```
>>> fruits = ['apple', 'mango', 'banana','cherry']
>>> {f:len(f) for f in fruits}
{'cherry': 6, 'mango': 5, 'apple': 5, 'banana': 6}
```

Exercise

- You have dictionary of points in competition
  - `scores = {'John' : 10, 'Emily' : 35, 'Matthew' : 50}`
  - Using dictionary comprehensions, create dictionary, where everyone gets triple amount of points