Conditional Execution and Loops

Conditional execution (`if` , `elif` , `else` statement)

```
In [ ]: val1 = 20

        if val1 > 15:
            print("above 15")
        else:
            print("not above 15")
```

```
above 15
```

```
In [ ]: n = int(input())
```

```
10
```

```
In [ ]: if n > 6:
            print("above six")
        elif n < 2:
            print("below two")
        elif n==5:
            print("equal to 5")
        else:
            print("neither")
```

```
above six
```

Loops

Two kind of loops in python:

- `while` loop
- `for` loop

`while` loop

In `while` loop, the commands inside the body of the loop are executed repeatedly as long as the condition of the loop holds true.

```
In [ ]: i = 1
        while i <= 10:
            print(i**2)
            i += 1
```

```
1
4
9
16
25
36
49
64
81
100
```

`continue` statements

`continue` can be used to skip parts of the body of the loop based on certain condition and move on to the next iteration of the loop.

In [ ]:
```
i = 0
while i <= 20:
    i += 1
    if i % 5 == 0:
        continue
    print(i)
```

```
1
2
3
4
6
7
8
9
11
12
13
14
16
17
18
19
21
```

`break` statement

It can be used to exit out of a loop based on some condition

In [ ]:
```
i = 1
while i <= 10:
    if i==5:
        break
    print(i*i)
    i += 1
```

```
1
4
9
16
```

for loop

It is used to execute a block of code repeatedly for each item/element in a data structure (list, tuple, set, dictionary etc.)

In [ ]:
```python
# example using a list:
lst1 = [1, 2, 3.5, 4]    # lists are defined using square brackets, can combine
for i in lst1:
    print(i**2)
```

```
1
4
12.25
16
```

In [ ]:
```python
# example using a dictionary (dict)
dct1 = {
    "a" : 1,
    "b" : 2,
    "c" : 3.5,
    "d" : 4
}
```

In [ ]:
```python
for j in dct1:
    print("the square of element", j, "is", dct1[j]**2)
```

```
the square of element a is 1
the square of element b is 4
the square of element c is 12.25
the square of element d is 16
```

In [ ]:
```python
dct1.items()
```

Out[ ]:
```
dict_items([('a', 1), ('b', 2), ('c', 3.5), ('d', 4)])
```

In [ ]:
```python
a, b = 1, 2
```

In [ ]:
```python
for key, val in dct1.items():
    print("the square of element", key, "is", val**2)
```

```
the square of element a is 1
the square of element b is 4
the square of element c is 12.25
the square of element d is 16
```

List comprehension

A way of creating new lists based on old lists by usually by applying some form of function or by performing some computation/calculation on each element of a list

In [ ]:
```python
lst2 = [x**2 for x in lst1]
lst2
```

Out[ ]:
```
[1, 4, 12.25, 16]
```