School Nova
Computer Science

Definite iteration: “for” loop

Classwork #5

By Oleg Smirnov

Some comments

```
number = 5
number = int(5)     # unnecessary int()

name = input("What is your name?")
name = str(input("What is your name?")     # unnecessary str()
```

Iterations: Definite loops

Definite iteration – the loop is repeated a certain number of times that you define.

```
for age in range(8):
    print(f"You are {age} years old.")
print(f"Hurray! You are seven years old!")
```

```
for age in (0, 1, 2, 3, 4, 5, 6, 7):
    print(f"You are {age} years old.")   # possible but inefficient
```

Iterations: Definite loops

range(x, y) is a sequence of integers from *x* (**included**) to *y* (**excluded**)
You can see all elements in the sequence: **print(list(range(x, y)))**

```
for i in range(x, y):
    print(i)
```

range(x) is a sequence of integers from zero (!) to *x* (excluded). The last elements in the sequence is *x* – 1.

The below lines do the same thing:

```
for i in range(4): print(i)
for i in range(0, 4): print(i)
for i in (0, 1, 2, 3): print(i)
```

For loops: Using step

You can add a **step** to the range function (and, therefore, the for loop).

```
print(list(range(0, 105, 5)))     # here the step is 5
for i in range(0, 105, 5): print(i)
```

For **reverse** loops you can you step = -1, for example:

```
for i in range(20, 10, -1): print(i)
```

You can only use **integers** with range(). You can't use float type!
However, you can go around this, for example:

```
# I need to print all tenths between 0 and 1
for i in range(0, 11):
    print(i / 10)
```

Classroom exercise I

Task:

Calculate and print the square (x ** 2) for all odd numbers between 1 and 19 (included!)

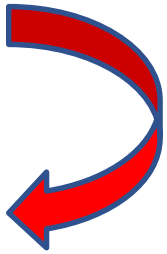
Solution:

Next page

Definite loops, break, and continue

Break and **continue** commands work similar to how they work with the indefinite loop while.

```
for i in <condition>:
    statement 1
    statement 2
    break
    statement 3
    statement 4
statement 5
```



```
for i in <condition>:
    statement 1
    statement 2
    continue
    statement 3
    statement 4
statement 5
```

For loop: Break and Continue Example

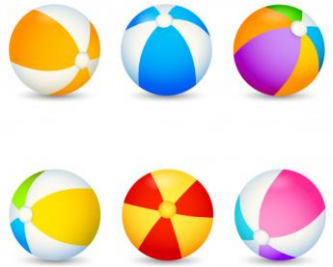

```
for i in range(5):
    print(i)
    break
    continue
```

```
>>>
0
```

```
for i in range(5):
    print(i)
    continue
    break
```

```
>>>
0
1
2
3
4
```

For loops: going over a finite collection of objects



Alternatively, a definite loop may go over a finite collection of objects.
One example of a collection of objects you have already seen: strings.
Strings consist of letters:

```
for i in “School Nova”: print(i)
```

There are many other types of finite collections of objects (which we will study closely soon), for example: lists, tuples, dictionaries, sets.

```
animals = [“cat”, “dog”, “cow”]     # this is a list; check type(animals)
for i in animals:
    print(i)
```

Above, you don’t define the number of iterations. Instead, the number of iterations is equal to the number of elements in the list.

Classroom examples

Please, see the Python code posted on Google Classroom