Introduction to Computer Programming in Python

Dr. William C. Bulko

Loops

The `for` statement

The general form of a `for` statement is:

```
    for <var> in <some kind of series>:
```

The easiest way to explain this is with an example:

```
    for i in [1, 2, 3]:
          print(i)
```

Produces the output:

```
1
2
3
```

- The thing in [ ] is called a *list*
- The number of times you go through the loop = the number of items in the list
- Each time you go through the loop, you assign the value of the next item to the variable in the `for` statement
- Don’t forget the colon
- Indentation is important!

The `range` function

`range(<number>)` produces a list of `int`s counting from zero up to `<number>-1`.

For example, `range(5)` gives you the list 0, 1, 2, 3, 4.

Example:

```
    for number in range(3):
            print(number)
```

produces the output:

```
0
1
2
```

The `range` function: more features

`range(<num1,num2>)` produces a list of `int`s counting from `<num1>` up to `<num2>-1`.

For example, `range(3,7)` gives you the list 3, 4, 5, 6.

start here

end at this number minus one

Finally, `range(<num1,num2,num3>)` produces a list of `int`s counting from `<num1>` up to `<num2>-1` counting by `num3's`.

For example, `range(4,36,5)` gives you the list 4, 9, 14, 19, 24, 29, 34.

start here

count by

end at this number minus one

Quick check:

What is the output?

```
def main():
    for i in [2, 4, 6]:
        print(i)
main()
```

What is the output?

```
def main():
    for i in range(3,22,3):
        print(i+1)
main()
```

Another way to iterate: the `while` statement

The general form of a `while` statement is:

```
while <boolean expression>:
```

Again, the easiest way to explain this is with an example:

```
powerOf2 = 1
while (powerOf2 < 100):
    print(powerOf2)
    powerOf2 = powerOf2 * 2
```

Produces the output:

```
1
2
4
8
16
32
64
```

Functions

In general:

```
# define all of your functions
def <function>():
def <function>():
def <function>():
def <function>():
   .
   .
   .
# define your main program
def main():
   .
   .
   .
# call your main program to start execution
main()
```

Things to remember about `def`

The `def` statement ends in a colon (":")

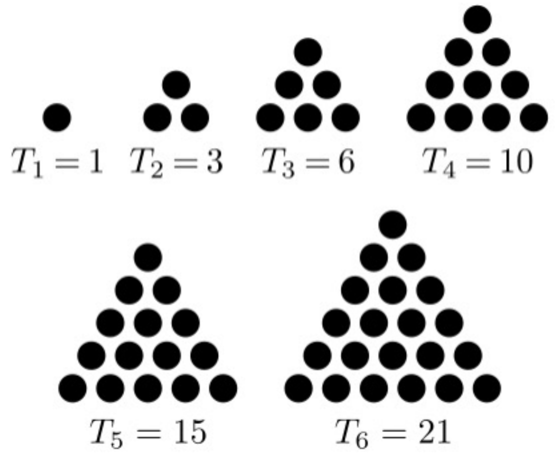
```
def printMyName(name):
    print("***************")
    print(name)
    print("***************")
```

All statements that are part of the function definition must be <u>indented</u>. This is how the Python interpreter knows that these statements are part of the definition of the function (and not part of some other part of your program).

Exercise:

A *triangular number* is an integer that corresponds to the number of objects arranged in an equilateral triangle, as shown in the picture.

The *n*th triangular number T_n is the number of dots in the triangular arrangement with n dots on a side, and is equal to the sum of the n natural numbers from 1 to n.



Write a function `triangle(n)` that accepts as an argument an `int`, and returns the nth triangular number.

Then write a main program that makes a table showing the first 20 triangular numbers.

Factorial Exercise:

The factorial function n! in math is defined this way:

1! = 1 = 1

2! = 1 x 2 = 2

3! = 1 x 2 x 3 = 6

4! = 1 x 2 x 3 x 4 =24

Write a function `factorial(myNumber)` that accepts as a parameter an `int`, and returns the factorial of that `int`.

Then write a main program that uses the `factorial` function to print out the factorials for numbers from 1 to 10.

Introduction to Computer Programming in Python

Dr. William C. Bulko

Strings

String Length Exercise:

Write a program that asks the user to enter a series of strings one at a time, and then prints out either

```
The string has an even number of characters.
```

or

```
The string has an odd number of characters.
```

as appropriate for each one. When the user hits “enter” without typing in a string, terminate the program.

The `for` statement

The general form of a `for` statement is:

```
for <var> in <some kind of series>:
```

The easiest way to explain this is with an example:

```
for i in [1, 2, 3]:
      print(i)
```

Produces the output:

```
1
2
3
```

- The thing in [ ] is called a *list*
- The number of times you go through the loop = the number of items in the list
- Each time you go through the loop, you assign the value of the next item to the variable in the `for` statement
- Don’t forget the colon
- Indentation is important!

Testing Strings

There are a number of useful methods (functions) that apply to strings.

`isalpha()`	returns True if there is at least one character in this string and all characters are alphabetic.
`isdigit()`	returns True if this string contains only number characters.
`islower()`	returns True if there is at least one character in this string and all characters are lowercase.
`isupper()`	returns True if there is at least one character in this string and all characters are uppercase.
`isspace()`	returns True if this string only contains whitespace characters.

Additional Useful String Methods

`upper()`	convert all alphabetic characters to uppercase
`lower()`	convert all alphabetic characters to lowercase
`capitalize()`	make first character uppercase and the rest lowercase
`replace(old,new)`	replace all instances of `old` in string with `new`
`lstrip ()`	remove all leading (left) whitespace from string
`rstrip ()`	remove all trailing (right) whitespace from string
`strip ()`	remove all leading and trailing whitespace from string
`center(length)`	creates a new string of the given length with the given string centered between spaces
`ljust(length)`	creates a new string of the given length consisting of the given string followed by spaces
`rjust(length)`	creates a new string of the given length consisting of spaces followed by the given string

Palindrome Exercise:

A string is a *palindrome* if it reads the same backwards and forwards. "`mom`", "`dad`", and "`noon`" are all palindromes.

Write a program that prompts the user to enter a string and reported whether or not the string is a palindrome. Ignore blanks at the start and end of the string.

Have the program check whether the first character is the same as the last character. If they are, move to the second character and the second to the last character. Continue until either:

- a mismatch is found
- all characters have been checked
- the string has an odd number of characters, and all characters except the middle character have been checked.

Introduction to Computer Programming in Python

Dr. William C. Bulko

Lists

Chart from earlier

The `for` statement

The general form of a `for` statement is:

```
for <var> in <some kind of series>:
```

The easiest way to explain this is with an example:

```
for i in [1, 2, 3]:
      print(i)
```

Produces the output:

```
1
2
3
```

- The thing in [ ] is called a *list*
- The number of times you go through the loop = the number of items in the list
- Each time you go through the loop, you assign the value of the next item to the variable in the `for` statement
- Don’t forget the colon
- Indentation is important!

Summary of Common List Operations

`list1 + list2`	concatenates two lists `list1` and `list2`
`myList * n, n * myList`	`n` copies of `myList` are concatenated together
`len(myList)`	returns the number of elements in `myList`
`min(myList)`	returns the smallest element in `myList`
`max(myList)`	returns the largest element in `myList`
`sum(myList)`	returns the sum of all elements in `myList`
`<, <=, > >=, ==, !=`	used to compare two lists
`x in myList`	True if `x` is an element of the list `myList`
`x not in myList`	True if `x` is not an element of the list `myList`
`myList[i]`	the `i`th element of `myList`
`myList[i:j]`	a sublist of `myList` consisting of the elements from `i` to `j` of `myList`