Hacettepe University

Computer Engineering Department

Programming in  python

BBM103 Introduction to Programming Lab 1

Week 5

Fall 2018

Control Flow – For Loops

```
for <variable> in range(some_number):
    <expression>
    <expression>
    ...
```

- Each time through the loop, `<variable>` takes a new value. It starts with the smallest value, and in the next loop it gets incremented, and so on, until it reaches the final value in the specified range.

- Example 1: **printing numbers in a given range**

```
for i in range(10):
    print(i)
```

New function:
`range()`

- Example 2: **printing numbers grater than a specified value**

```
numbers = "123456789"
for i in numbers:
    if int(i) > 3:
        print(i)
```

`range(number)` generates integers from 0 up to, **but not including** `number`.

- Example 3: **printing characters that are not in a string**

```
first_text = "This is a sample text for testing."
second_text = "This is another sample text."
for letter in first_text:
    if letter not in second_text:
        print(letter)
```

- Example 4: **printing numbers divisible by three**

```
for number in range(2,50):
    if int(number) % 3 == 0:
        print(number)
```

`range(start, stop)` generates integers from `start` up to `stop`, but not including `stop`.

- Example 5: **finding the cube root**

```
x = int(input('Enter an integer: '))
answer = None
cube_root_found = False
for i in range(0, abs(x)+1):
    if i**3 == abs(x):
        answer = i
        cube_root_found = True
if not cube_root_found:
    print (x, 'is not a perfect cube')
else:
    if x < 0:
        answer = -answer
    print('Cube root of', x,'is', answer)
```

New function:
`abs()`

This is not a very efficient algorithm, but it gets the job done!

Food for thought:

- Why?
- How can we make it more efficient?

`abs(number)` returns the absolute value of `number`.

- Example 6: **split**

```
sentence="Yürüdüğümüz yol bitmiş , bir başka sokağa açılmıştı ."

for word in sentence.split():
    print(word)
```

New function:
`split()`

- Example 7: **even numbers**

```
numbers="12,15,47,86,98"

for number in numbers.split(","):
    if int(number)%2 ==0:
        print(number,"is even")
```

Control Flow – While Loops

```
while (condition is True):
    <expression>
    <expression>
    ...
```

- *While* loops are used for repeating sections of code until a defined condition is no longer met. If the condition is initially false, the loop body will not be executed at all.

A loop becomes infinite loop if a condition never becomes FALSE. You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

- Example 8: **inputcondition**

```
n = input("Please enter 'hello':")
while n.strip() != 'hello':
    n = input("Please enter 'hello':")
```

- Example 9: **subtraction**

```
i = 0

# While loop condition.
while i > 100:
    print(i)
    # Subtract two.
    i -= 2
```

- Example 10: **guess**

```
import random

number = random.randint(1, 25)

number_of_guesses = 0

guess=0

while number_of_guesses < 5 and guess!=number:
    print('Guess a number between 1 and 25:')

    guess = input("Please enter a number")
    guess = int(guess)

    number_of_guesses = number_of_guesses + 1
```

New function:
`randint()`

Functions

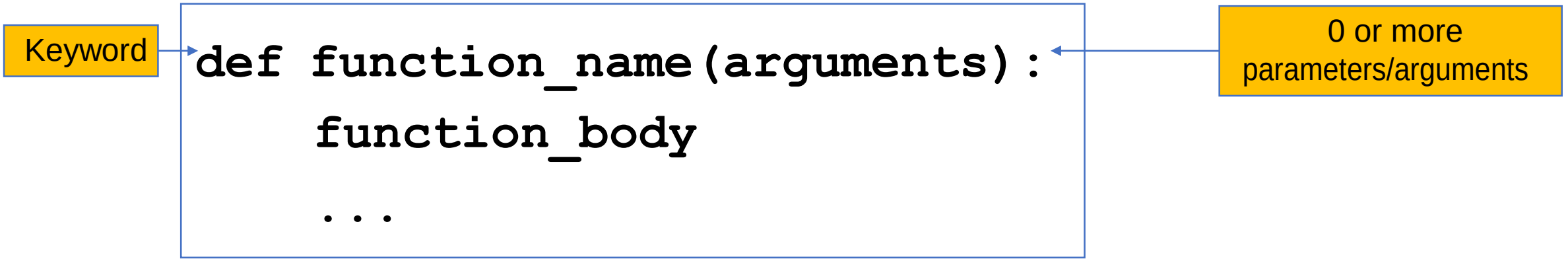
- **Good programming practice**: It is **functionality** that is important, not the amount of code!
- **The importance of functions**:
  - Break your code into separate, independent parts that will work together to solve the ultimate problem (**DECOMPOSITION**).
  - Hide the details of your computation as long as you know what it produces (**ABSTRACTION**)

Functionscont.

- **The advantages of functions**:
  - Break your code into **simpler independent modules**
  - These modules can be **reused** as many times as you like
  - And they need to be **debugged only once**
  - Keep your code **more organized and easier to understand**

Functionscont.

Defining functions:



Calling functions:

```
function_name(arguments)
```

- Example 1: **defining a <u>void</u> function (function that does not return a value)**

```
def greeting(name):
    print("Good afternoon, " + name + ".")

greeting("Emre")
```

Defining function

Calling function

  - **Output:** `Good afternoon, Emre.`

- Example 2: **defining a <u>fruitful</u> function (function that returns a value)**

```
def maximum(x, y):
    if x > y:
        return x
    else:
        return y
max_number = maximum(1, 5)
print("The maximum of two numbers is", max_number)
```

The function **returns** a value

- Example 3: **an example of a Boolean function (function that returns True or False)**

```
def is_even(number):
    return number%2 == 0
```

  - This function will return True if number is even, and False otherwise.

- An example use of this function:

Calling the function from within an **`if`** statement

```
if is_even(number):
    print (number," is an even number.")
else:
    print (number," is an odd number.")
```

- Example 4: **a function that calculates the factorial of a number**

```
def factorial(number):
    product = 1
    for i in range(1, number+1):
        product = product * i
    return product
```

This line can be written more compactly as:

```
product *= i
```

- An example use of this function:

```
print("The factorial of 6: 6! = ", factorial(6))
```

* The **factorial** of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n. For example, 4! = 4 × 3 × 2 × 1 = 24

- Example 5: **Calculating area of plane shapes**

```
def triangle_area(b, h):
    return b*h/2
def square_area(a):
    return a*a
def rectangle_area(a, b):
    return a*b
user_choice = int(input("""Choose a shape you wish to calculate the area of:
(1) Triangle
(2) Square
(3) Rectangle\n1-3: """))
if user_choice == 1:
    base = int(input("Enter the length of the triangle base: "))
    height = int(input("Enter the height of the triangle: "))
    print("The area of the triangle is", triangle_area(base, height))
elif user_choice == 2:
    side = int(input("Enter the length of the square side: "))
    print("The area of the square is", square_area(side))
elif user_choice == 3:
    width = int(input("Enter the width of the rectangle: "))
    height = int(input("Enter the height of the rectangle: "))
    print("The area of the triangle is", rectangle_area(width, height))
else:
    print("Sorry, there is no such option.")
```

`\n` is the newline character

Exercises

1. Write a program that asks for a number N as a user input, and calculates the sum of odd numbers, and the average of even numbers starting from 1 up to and including N.
2. Write a Boolean function that checks if a string contains '@' sign and at least one '.' dot (disregard the order for the sake of simplicity). Use that function to check if a user input is a valid e-mail or not.
3. Guessing game! Pick a number randomly. While user does not guess the number correctly give an instruction about the number and take another guess from user.

   Instruction: If the guessed number is lower than the picked number print

   «increase your number»

   else print

   «decrease your number»

Things to remember

- Indentation is very important in Python! To indicate a block of code in Python, you **must indent each line of the block by the same amount**.

- **Practice makes perfect**: the more you practice programming the easier it gets. It is easy to get stuck in the beginning, but don't get discouraged. Work with simple examples first. Move on to the harder examples when you have fully grasped the simple ones.

- It is a lot of fun telling your computer what to do! **Stay motivated**.