# Conditional and Looping Statements

# Functions

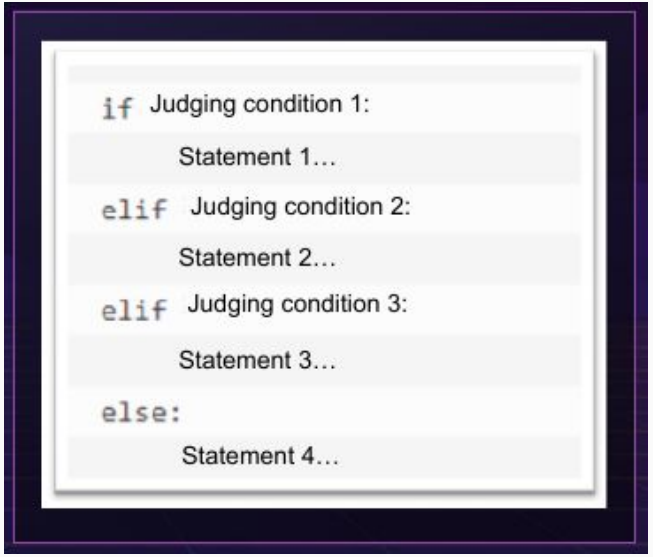
Wendley S. Silva

Agosto de 2020

Universidade Federal do Ceará

## if statements

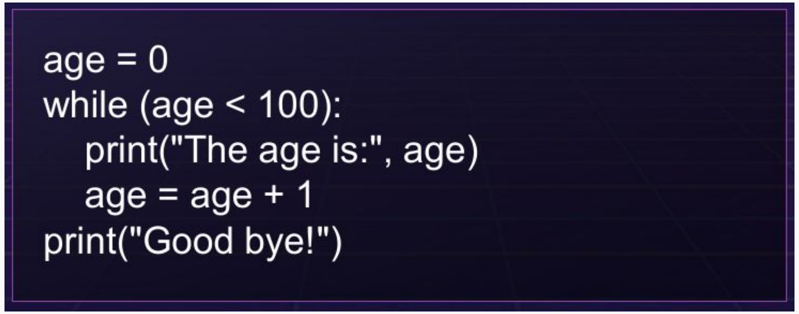
- Python supports three control structures: if, for, and while, but it does not support switch statements in the C language.
- In Python programming, if statements are used to control execution of control programs, and the basic form is:



```
if Judging condition 1:
        Statement 1…
elif  Judging condition 2:
        Statement 2…
elif  Judging condition 3:
        Statement 3…
else:
        Statement 4…
```

## while statements

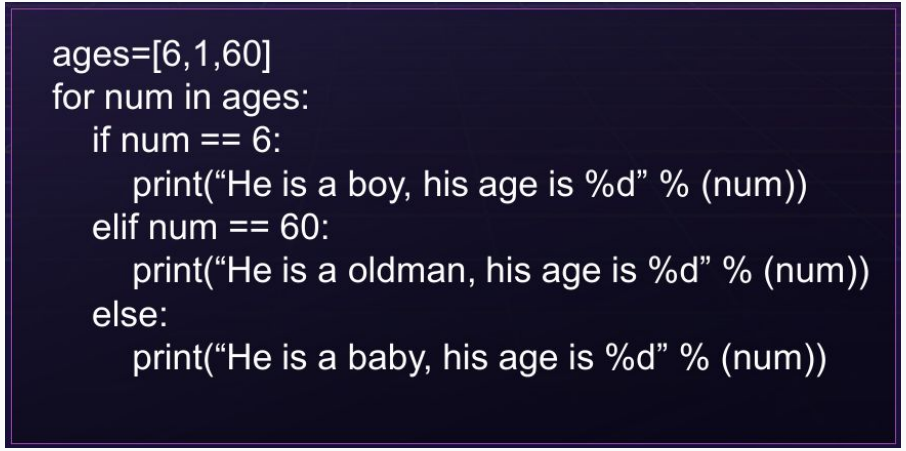
- The while statement in the Python language is used to execute a loop program that, under certain conditions, loops through a program to handle the same tasks that need to be repeated.
- When the condition of a while statement is never a Boolean false, the loop will never end, forming an infinite loop, also known as a dead loop.



```
age = 0
while (age < 100):
    print("The age is:", age)
    age = age + 1
print("Good bye!")
```

## for statements

- In the Python language, the for loop can traverse any items of a sequence, such as a list, a dictionary, or a string.
- The for statement is different from a traditional for statement. The former accepts an iterative object (such as a sequence or iterator) as its argument, and one element is iterated each time.



```
ages=[6,1,60]
for num in ages:
    if num == 6:
        print(“He is a boy, his age is %d” % (num))
    elif num == 60:
        print(“He is a oldman, his age is %d” % (num))
    else:
        print(“He is a baby, his age is %d” % (num))
```

## break and continue

With the **break** statement we can stop the loop before it has looped through all the items

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

## continue

With the **continue** statement we can stop the current iteration of the loop, and continue with the next:

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

    if x == "banana":

        continue

    print(x)
```

## Functions

## Functions

A function is a code segment that is organized, reusable, and used to implement a single function or associated functions.

Functions can improve the modularity of applications and reuse of code.

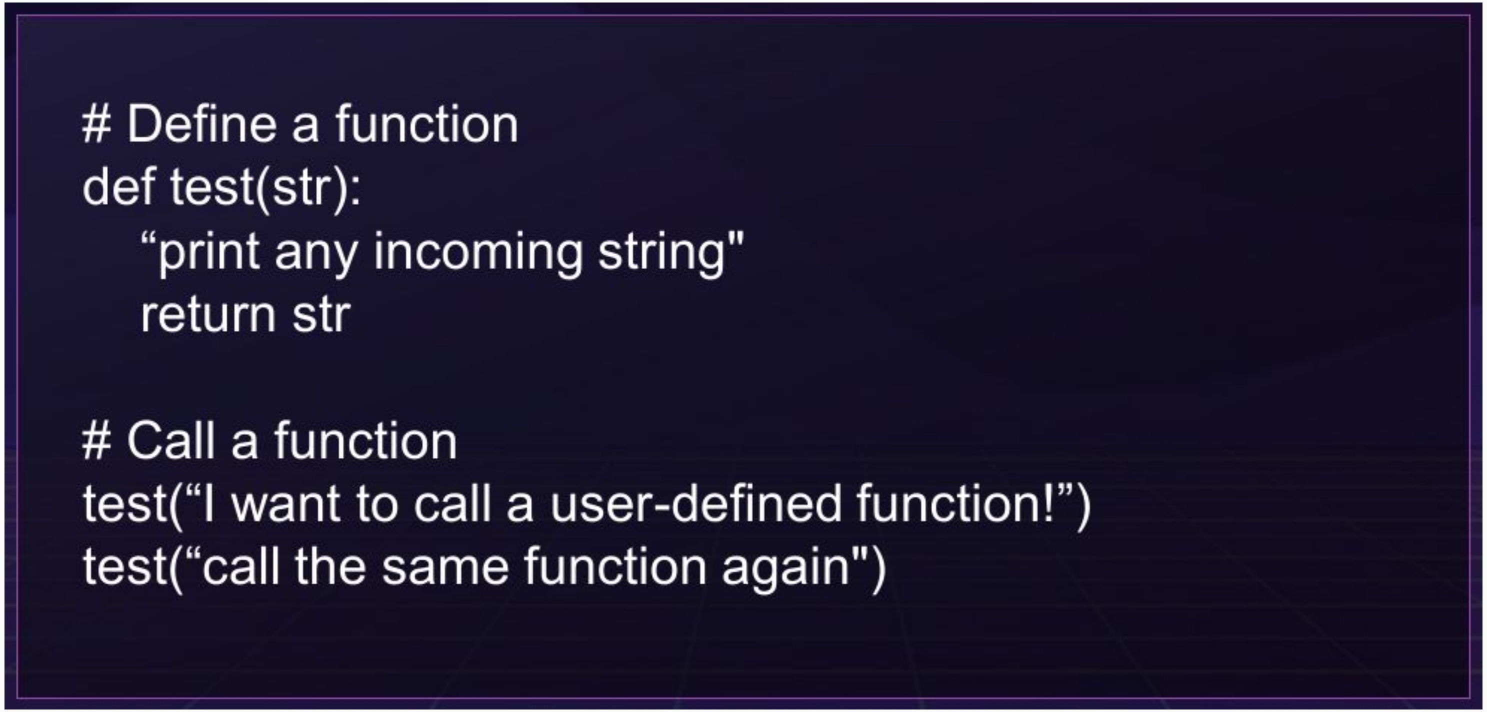
Python provides a number of built-in functions, such as print(). You can also create your own functions, which are called user-defined functions.

## Defining a function

Define a function with the following rules:

- The function code block begins with a **def** keyword, followed by the function name and parentheses ().
- Any incoming arguments and independent variables must be placed in the middle of the parentheses. Parentheses can be used to define arguments.
- return[expression] ends a function, and selectively returns a value to the caller. Returning without an expression is equivalent to returning none.

## Calling a function



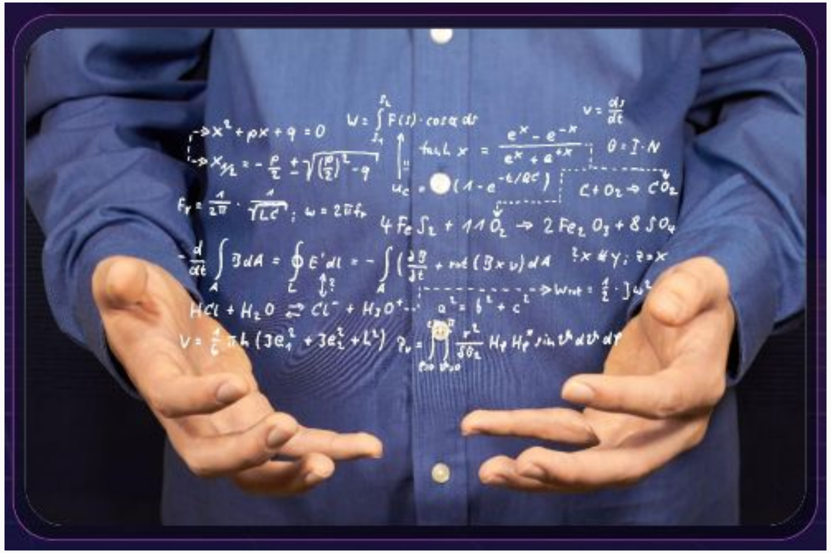
```
# Define a function
def test(str):
    “print any incoming string"
    return str

# Call a function
test(“I want to call a user-defined function!”)
test(“call the same function again")
```

# Global variables and local variables

A variable defined within a function has a local scope and is called a local variable.

A variable defined beyond a function has a global scope and is called a global variable.



## Obrigado pela atenção