


Lesson	Can you?
1 First steps	Describe what algorithms and programs are Recall that a program written in a programming language needs to be translated in order to be executed Write simple Python programs that display messages, assign values to variables, and receive keyboard input Locate and correct common syntax errors
2 Crunching numbers	Describe the semantics of assignment statements Use simple arithmetic expressions in assignment statements to calculate values Receive input from the keyboard and convert it to a numerical value
3 At a crossroads	Use relational operators to form logical expressions Use binary selection (if, else statements) to control the flow of program execution Generate and use random integers
4 More branches	Use multi-branch selection (if, elif, else statements) to control the flow of program execution Describe how iteration (while statements) controls the flow of program execution
5 Round and round	Use iteration (while loops) to control the flow of program execution Use variables as counters in iterative programs
6 Putting it all together	Combine iteration and selection to control the flow of program execution Use Boolean variables as flags

Useful websites

- www.scratch.mit.edu
- www.en.wikipedia.org
- www.teachinglondoncomputing.org/lego-braille
- www.csunplugged.org/en
- www.csfieldguide.org.nz/en
- www.archive.org/details/advancementof100baco/page/256
- www.curriculum.code.org
- www.cs4fn.org
- www.denninginstitute.com/pjd/GP/GP-site/welcome.html
- www.futurelearn.com/courses/how-computers-work



```

default = {
    }
    global_scale_setting = BlenderPropertyPanel(
        name="Scale",
        min=0.01, max=1000.0,
        default=1.0,
    )

def execute(self, context):
    # get the folder
    folder_path = (os.path.dirname(self.filepath))

    # get objects selected in the viewport
    viewport_selection = bpy.context.selected_objects

    # get export objects
    obj_export_list = viewport_selection
    if self.use_selection_setting == False:
        obj_export_list = [i for i in bpy.context.scene.objects]

    # deselect all objects
    bpy.ops.object.select_all(action='DESELECT')

    for item in obj_export_list:
        item.select = True
        if item.type == 'MESH':
            file_path = os.path.join(folder_path, "{}.obj".format(item.name))
            bpy.ops.export_scene.obj(filepath=file_path, use_selection=True,
                axis_forward=self.axis_forward_setting,
                axis_up=self.axis_up_setting,
                use_animation=self.use_animation_setting,
                use_mesh_modifiers=self.use_mesh_modifiers_setting,
                use_edges=self.use_edges_setting,
                use_smooth_groups=self.use_smooth_groups_setting,
                use_smooth_groups_as_faces=self.use_smooth_groups_as_faces_setting,
            )

```

Just like **algorithms**, **programs** are designed to solve a problem. In order to solve the problem, the program needs to perform several tasks. **Code** is written to perform each of these tasks.

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data that can be changed later in the program. For example,

```
number = 10
```

Here, we have created a variable named `number`. We have assigned the value `10` to the variable.

The `if..else` statement evaluates `test expression` and will execute the body of `if` only when the test condition is `True`.

If the condition is `False`, the body of `else` is executed. Indentation is used to separate the blocks.

for loops are used when you have a block of code which you want to repeat a **fixed number of times**. The for-loop is always used in combination with an **iterable object**, like a list or a range. The Python **for** statement iterates over the members of a sequence in order, executing the block each time. Contrast the **for** statement with the **"while" loop**, used when a condition needs to be checked each iteration or to repeat a block of code forever. For example:

For loop from 0 to 2, therefore running 3 times.

```
for x in range(0, 3):
    print("We're on time %d" % (x))
```

While loop from 1 to infinity, therefore running forever.

```
x = 1
while True:
    print("To infinity and beyond! We're getting
        close, on %d now!" % (x))
    x += 1
```

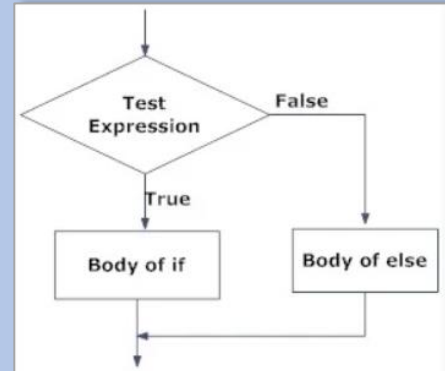


Fig: Operation of if...else statement
Flowchart of if...else statement in Python

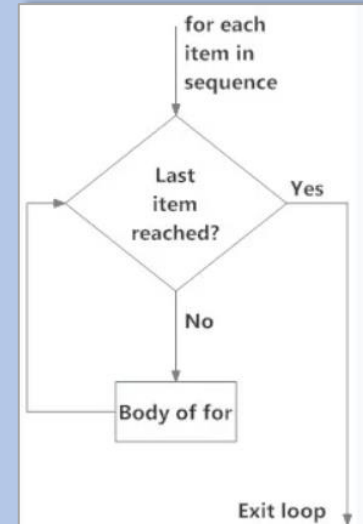


Fig: operation of for loop
Flowchart of for Loop in Python

```
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

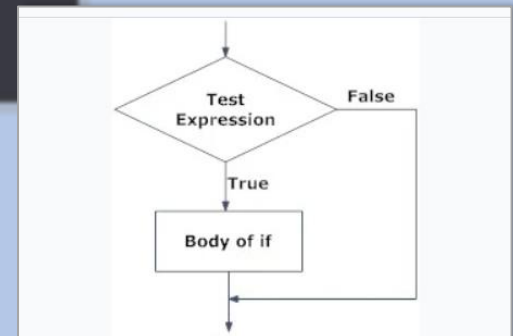


Fig: Operation of if statement
Flowchart of if statement in Python programming