Python Building Blocks

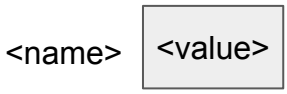
There are some basic building blocks of programming. This sheet will show you what they look like in Python. Things surrounded by `< >` are meant to be replaced when you type out the statement. For example, you might want to print something to the screen. The pattern for that looks like `print(<text>)` and you might replace that with `print("Hello World")`.

Printing to the Screen

`print(<text>)`
Prints the `<text>` to the screen.

Variables

`<name> = <value>`

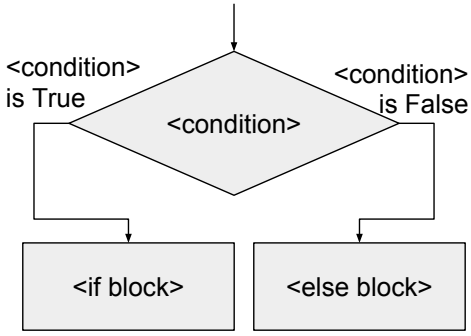


Assigns the `<value>` to the variable called `<name>`. This makes a new variable if one called `<name>` doesn't already exist.

Branches

Single-Condition

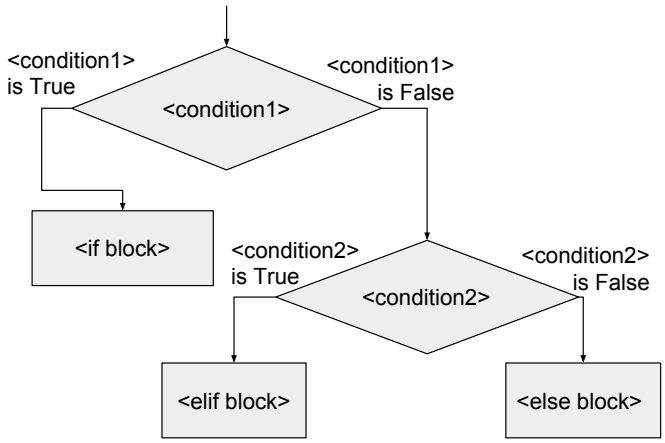
```
1 if <condition>:
2     <if block>
3 else:
4     <else block>
```



Runs the `<if block>` if the `<condition>` is true, runs the `<else block>` if the `<condition>` is false.

Multiple-Condition

```
if <condition1>:
    <if block>
elif <condition2>:
    <elif block>
else:
    <else block>
```

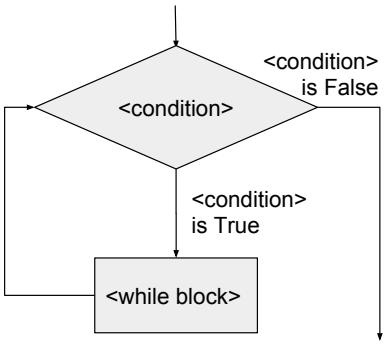


Runs the `<if block>` if `<condition1>` is true, runs the `<elif block>` if `<condition1>` is false and `<condition2>` is true, and runs the `<else block>` if both `<condition1>` and `<condition2>` are false.

Loops

While Loop

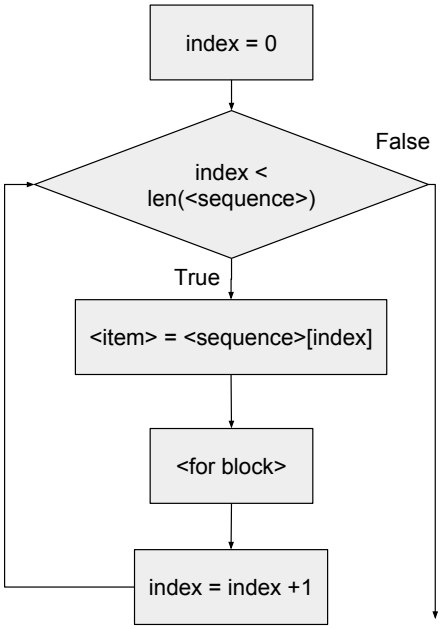
```
while <condition>:
    <while block>
```



Checks the `<condition>`, runs the `<while block>` if the `<condition>` is true, then checks the `<condition>` again. This continues until the `<condition>` is false.

For Loop

```
for <item> in <sequence>:
    <for block>
```



Runs the `<for block>` for each `<item>` that is in the `<sequence>`. Each time it runs `<for block>` the `<item>` will be the next item in the `<sequence>`. So we can say that it runs *while* there are more items in the sequence.

We can imagine that we make a variable `index = 0`. Then, if `index` is less than the length of the `<sequence>`, we set `<item>` to `<sequence>[index]` and run the `<for block>`. If the `index` is greater than or equal to the length of the `<sequence>`, then we exit the loop. Finally, we add one to `index` and loop back to check if `index` is still less than the length of the `sequence`.

Functions

```
def <name>(<parameters>):
    <function body>
```

Defines a function called `<name>` that takes a list of `<parameters>` (if you don't need any parameters, then you can just type `def <name>():` which are the data you want to pass into the function. For example, the `print("Hello World")` function takes a parameter that it prints to the screen. Here, that parameter is the string `"Hello World"`

```
<name>()
```

When you call a function, by typing its name as shown above, the runs and then we return to where the function was called. If the contains a `return` statement, then you can imagine that the call to the function is replaced by whatever value the function returns. For example, a call to a function that adds two numbers might look like `add(2,2)`, which would return the value `4`.