PYTHON CRASH COURSE:
SEQUENCE CONTROL

By: Lekiz Rai

Date: 24/05/2024

Contents

Two-way selection

If-else statement

If-else statement format

if *<bool>* **:**

    <block0>

else:

    <block1>

Sample code

```
# <block0> will run if <bool> is true. Otherwise,
# <block1> will run.
if True:
    print("True")  # This block will run
else:
    print("False") # This block will not run
```

Multiple selection

If-elif-else statement

If-elif-else statement format

if *<bool0>* **:**
    <block0>
elif *<bool1>* **:**
    <block1>
elif *<bool2>* **:**
    <block2>
...
else:
    <blockn>

Sample code

```
# <block0> will run if <bool0> is true. Otherwise,
# <block1> will run if <bool1> is true and so on.
# Finally, if all boolean expressions are false,
# <blockn> will run.
if False:
    print("0") # This block will not run
elif True:
    print("1") # This block will run
else:
    print("2") # This block will not run
```

Match-case statement

Match-case format

match *<case>* **:**
    case *<case0>* **:**
        <block0>
    case *<case1>* **:**
        <block1>
    ...
    case _ **:**
        <blockn>

Sample code

```
# <block0> will run if <case> == <case0>. Otherwise,
# <block1> will run if <case> == <case1> and so on.
# Finally, if all cases are different from <case>,
# <blockn> will run as set default.
lang = "Python"
match lang:
    case "C++":
        print("C++") # This block will not run
    case "Python":
        print("Python") # This block will run
    case _:
        print("Java") # This block will not run
```

Loops

For statement

For statement format

for *<element>* **in** *<iterator>* **:**

    <block0>

else:

    <block1>

Sample code

```
# The for statement will go through <iterator> and
# store respectively each element into <element>.
# The <block0> will run at each iteration and finally
# after going through the whole lst, <block1> will run.
res = ""
lst = [1, 2, 3]
for e in lst:
    res = res + str(e)
else:
    res = res + "4"
print(res) # "1234"
```

Note: In *for* statement, *<element>* is protected. That means if we have declared a variable named *i* out of the *for* statement, when we use it as *<element>* for *for* statement then its initial value will not be modified.

While statement

While statement format

while *<bool>* **:**

    <block0>

else:

    <block1>

Sample code

```
# The while statement will run <block0> time by time
# until <bool> is false. When <bool> gets false,
# <block1> will run.
res = ""
count = 1
while count < 4:
    res = res + str(count)
    count = count + 1
else:
    res = res + "4"
print(res) # "1234"
```

Break and continue statment

Break statement

The **break** statement will **break** and **jump out of** the **most nested loop**. When we do that, at both cases of *for* and *while* statement, **block** at *else* will **not run**.

Continue statement

The **continue** statement will **ignore** the **remained parts** and **jump back** to the start of the **most nested loop**, in *for* statement it will start with the next value of <*element*>.

Sample code

```
# Break statement
res = ""
lst = [1, 2, 3]
for e in lst:
    if e == 2:
        break
    res = res + str(e)
else:
    res = res + "4"
print(res) # "1"
```

Sample code

```
# Continue statement
res = ""
lst = [1, 2, 3]
for e in lst:
    if e == 2:
        continue
    res = res + str(e)
else:
    res = res + "4"
print(res) # "134"
```

Some common sample codes for loop

Sample code

```
# Loop from 1 to 3
for i in range(1, 4):
    print(i, end="") # 123

# Loop from 3 to 1
for i in range(3, 0, -1):
    print(i, end="") # 321

# Loop from 1 to 5 step 2
for i in range(1, 6, 2):
    print(i, end="") #135
```

Sample code

```
# Loop from 5 to 1 step 2
for i in range(5, 0, -2):
    print(i, end="") #531

# Loop through a list/tuple
for e in [1, 2, 3]:
    print(e, end="") #123

# Loop through a dictionary
for e in {"a": 13, -2: True}:
    print(e, end="") # a-2
```