Flow Control

April 23, 2020

1 Script & Flow Control

2 Another Data Type - Boolean

- Boolean is a data type that can be only `True` or `False`. For example, `pythonIsEasy = True`
- Boolean in conjunction with logical and relational operators are often used as condition in flow control

3 Boolean Operators - `and`, `or`, `not`

- x `and` y: if x is `False`, then x, else y.

- x `or` y: if x is`False`, then y, else x.
- `not` x: if x is `False`, then `True`, else `False`

4 `and`, `or` - short-circuit operators

- `and`: it only evaluates the second argument if the first one is `True`.
- `or`: it only evaluates the second argument if the first one is `False`

[14]:
```
x = 15 # also test for 5, 15
test = (not(x % 3)) and (x % 5);
print('not(x % 3) is: ',not(x % 3) )
print('x%5 is:  ', x % 5)
print('test is: ', test)
```

```
not(x % 3) is:  True
x%5 is:   0
test is:  0
```

5 Relational Operators

- A simple **comparison** that is evaluated as **Boolean**
- The standard comparison operators are: `<` (less than), `>` (greater than), `==` (equal to), `<=` (less than or equal to), `>=` (greater than or equal to) and `!=` (not equal to).

6 Student Exercise

An boolean expression that is evaluated as `True` only if a number is multiple of 3, but not multiple of 5

[15]:
```
x = 18
test = (x % 3) == 0 and (x % 5) != 0
print(test)
```

```
True
```

7 Working with Python - Script Mode

- Limitation of Interactive mode: When quit from the session, the definitions made (functions and variables) are lost.
- Script Mode:
  - save them in files/scripts and run a file/script as input for the interpreter.
  - filename.**py**
- Introducing IDLE Editor

8 Loop

- Loop is used to specify a sequence of statements once but the same sequence may be carried out many times in succession.
- `for` Loop: **Collection-controlled**, to loop over each item in a collection and terminate when collection is run through
- `while` loop: **Condition-controlled**, to loop till some condition is met and terminate
- **Indefinite** Loop: Never terminate

[16]:
```
#Print integers from 0 to 5.
#Without using loop, 6 print statement
#What if from 0 to 1,000? NOT SCALABLE!
print(0)
print(1)
print(2)
print(3)
print(4)
print(5)
```

```
0
1
2
3
4
5
```

[17]:
```
#Print integers from 0 to 5
#With for loop, iterate over list[0,1,2,3,4,5] returned by range(0,6)
#2 lines of code, scalable. range(0, 1001) for [0,1...,999,1000]
for x in range(0,6):
    print(x)
```

```
0
1
2
3
4
5
```

9 Block

- Every line in the body of the loop is indented: indentation is Python's way of grouping statements as **block** of code
- Blocks of code are denoted by line indentation, which is rigidly **enforced**.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount, either by tab or spaces

[18]:
```
#Print integers from 0 to 5
#With while loop, iterate while x<6; terminate when x=6
#4 lines of code, scalable. x<1001 for [0,1...,999,1000]
x = 0
while (x < 6): #truth value testing
    print(x)
    x = x + 1 #what if omitting this line?
```

```
0
1
2
3
4
5
```

10 Truth value testing

- A Boolean
- `None` is `False`
- An integer: Any **non-zero** integer value is `True`; zero is `False`
- For a string or list value anything with a **non-zero length** is `True`; **empty** sequences are `False`

[21]:
```
if not "": print ("Not an empty string")
if not []: print ("Not an empty list")
if not 0: print("Not zero")
```

```
Not an empty string
Not an empty list
Not zero
```

[6]:
```python
#Print integers from 0 to 5
#With while(true) and break loop, iterate forever, but break out of loop and␣
 ↪terminate when x=6
#5 lines of code, scalable. if x== 1001 for [0,1...,999,1000]
x = 0
while (True):
    if x == 6: break
    print(x)
    x = x+1
```

```
0
1
2
3
4
5
```

11 Student Exercise - `while` loop

Write a program to display an initial sub-sequence up to 20 of the Fibonacci series, i.e the sum of two elements defines the next. For example, 1,1,2,3,5...X(n) = X(n-1)+X(n-2)

[22]:
```python
# Fibonacci series: the sum of two elements defines the next.
a = 0; b = 1; # set up base
while b < 20:   # condition
    print(b)
    c = a + b
    a = b
    b = c
```

```
1
1
2
3
5
8
13
```

12 Choice/Decision - `if..else`

- Conditional constructs are to perform different computations or actions depending on whether a Boolean condition evaluates to be `True` or `False`.
- `if-else` statement in Python is the most common construct that allows this.

- `else` if optional.

[7]:
```
a = 3
if a > 3:
    print("a is greater than 3")
else:
    print("a is not greater than 3")
```

```
a is not greater than 3
```

13 Choice/Decision - `elif`

- Short for `else if`, quivalent to **nested if**, but more compact and less excessive indentation.
- Inserting as many as `elif` to give many choices as you need
- Python's subtitute for `switch...case`in other languages

[8]:
```
a = 3
if a > 3:
    print("a is greater than 3")
elif a == 3:
    print("a is equal to 3")
else:
    print("a is less than 3")
```

```
a is equal to 3
```

14 Student Exercise

- Rewrite preious example with elif using **nested if**

[9]:
```
a = 3
if a > 3:
    print("a is greater than 3")
else:
    if a == 3:
        print("a is equal to 3")
    else:
        print("a is less than 3")
```

```
a is equal to 3
```

15 Student Exercise

Write a program that takes percentage score from users and convert it to letter grade. A: 90%-100%; B: 80%-89%; C: 70%-79%;D: 60%-69%; F: 0%-59%

[9]:
```
result = int(input("Please enter raw scores: "))

if result >= 90:
    print("A")
elif result >= 80:
    print("B")
elif result >= 70:
    print("C")
elif result >= 60:
    print("D")
else:
    print("F")
```

```
Please enter raw scores: 98
A
```

16 Student Exercise

Use a `for` loop and `if` statement to identify and print the odd number in a list num_list that contains the following 10 numbers: 76, 64, 25, 23, 4, 17, 56, 14, 90, 28

[11]:
```
num_list = [76, 64, 25, 23,  4, 17, 56, 14, 90, 28]
for x in num_list:
    if x % 2 != 0:
        print(x)
```

```
25
23
17
```