CS 38003 PYTHON PROGRAMMING

Ruby Tahboub

IF STATEMENTS AND NESTED IF-STATEMENTS

- The general form of `if`, `if/else`, and `if/elif/else`

```
if condition:
   body
```

```
if condition:
   body1
else:
   body2
```

```
if condition1:
    body1
elif condition2:
    body2
elif condition3:
    body3
else:
    body4
```

- Blocks are indented
- Colin after condition and else

IF STATEMENT

```
a = 20
if a > 10:
  print('a > 10 is True')

print('bye.')
```

```
a > 10 is True
bye.
```

```
a = 20
if a > 30:
  print('a > 30 is True')

print('bye.')
```

```
bye.
```

IF STATEMENT

```
a = 20
if a > 10:
  print('a > 10 is True')
  print('a > 10 is True')
  print('a > 10 is True')

print('bye.')
print('bye.')
```

```
a > 10 is True
a > 10 is True
a > 10 is True
bye.
bye.
```

```
a = 20
if a > 30:
  print('a > 30 is True')
  print('a > 30 is True')
  print('a > 30 is True')

print('bye.')
print('bye.')
```

```
bye.
bye.
```

IF-ELSE STATEMENT

```
a = 20
if a > 10:
  print('a > 10 is True')
else:
  print('a > 10 is False')

print('bye.')
```

```
a > 10 is True
bye.
```

```
a = 5
if a > 10:
  print('a > 10 is True')
else:
  print('a > 10 is False')

print('bye.')
```

```
a > 10 is False
bye.
```

IF-ELIF-ELSE STATEMENT

```
score = 84
if score >= 90:
  print('you got an A.')
elif score >= 80:
  print('you got a B.')
elif score >= 70:
  print('you got a C.')
else:
  print('you got an F.')

print('bye.')
```

```
you got a B.
bye.
```

```
score = 84
if score >= 70:
  print('you got a C.')
elif score >= 80:
  print('you got a B.')
elif score >= 90:
  print('you got an A.')
else:
  print('you got an F.')

print('bye.')
```

```
you got a C.
bye.
```

COMMON MISTAKES

```
a = 20
if a > 30:
  print('a > 10 is True')
  print('a > 10 is True')
    print('a > 10 is True')

print('bye.')
```

```
inconsistent identation
File "test.py", line 5
print('a > 10 is True')
      ^
IndentationError: unexpected indent
```

```
a = 20
if a > 30:
  print('a > 10 is True')
print('hello')
else:
  print('a > 10 is False')

print('bye.')
```

```
  File "test.py", line 5
    else:
       ^
SyntaxError: invalid syntax
```

LOOPS

THE RANGE FUNCTION

The general form:

```
range(start, end, step)
range(start, end)
range(end)
```

- The range function returns a list of numbers starting with ‘start’, ending with ‘end’ with a step ‘step’.
- Start is an integer representing the start of the range.
- End is an integer representing the end (exclusive) of the range.
- Step is an integer representing the value of the increment.

THE RANGE FUNCTION

```
range(10):  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(3,10):  [3, 4, 5, 6, 7, 8, 9]
range(2,10,2):  [2, 4, 6, 8]
range(15,10,1):  []
range(-15,-3,4):  [-15, -11, -7]
range(-2, 5):  [-2, -1, 0, 1, 2, 3, 4]
range(10,1,-1):  [10, 9, 8, 7, 6, 5, 4, 3, 2]
range(10,1,-2):  [10, 8, 6, 4, 2]
range(-2):  []
```

DEFINITE LOOPS

- A definite loop repeats for a specific number of times (for loop, counter-controlled while loop.

```
for var in sequence:  body
```

- Looping 10 times using `for` and `while`.

```
# Loop 10 times using for
for i in range(10):
    # do work
    print(i)
```

```
0
1
...
9
```

```
counter = 0
while counter < value :
    body
    counter = counter + 1
```

```
# Loop 10 times using while
# Begin with i = 0
i = 0
while i < 10:
    # do work till condition i < 10 is met
    print(i)
    i = i + 1
```

```
0
1
...
9
```

INDEFINITE LOOPS

```
while condition:
    body
```

- The condition is a Boolean expression, just like in if statements.
- The body is a sequence of one or more statements.
- The loop executes repeatedly as long as the condition remains `True`.
- When the condition is `False`, the loop terminates.

LOOPS THAT NEVER TERMINATE

- If the condition of the while loop is always true, then the result is an infinite loop

```
i = 0
while i < 10:
  print(i)
```

- The variable i never changes, and the condition is always true.

BREAK AND CONTINUE

- break is used to terminate a loop when the termination condition is satisfied.

```
for i in range(5):
    if i == 2:
        print("we are breaking the loop!")
        break
    print(i)
```

```
0
1
we are breaking the loop!
```

- continue is used to terminate the current iteration of a loop.

```
for i in range(5):
    if i == 2:
        print("we are skipping this iteration!")
        continue
    print(i)
```

```
0
1
we are skipping this iteration!
3
4
```

NESTED LOOPS

- A loop where the body is another loop.
- For each iteration of the outer loop, the inner loop will be executed.
- Example: print the prime numbers between 3 and 15.

```
for n in range(3, 16):

    isPrime = True
    for num in range(2, n):
        if n % num == 0:
            isPrime = False

    if isPrime == True:
        print(n)
```

```
3
5
7
11
13
```

THANK YOU!