COMP 204

Control flow - Conditionals

Mathieu Blanchette,
based on material from Yue Li, Carlos Oliver and Christopher Cameron

Quiz 4 password

Assignment #1 will be released later today!

Back to last lecture

Goal: Write a program that computes the body mass index (BMI) of a person: $BMI = weight/(height^2)$

```
weight = 69
height = 1.8
BMI = weight/(height**2)
print('A person with weight', weight, 'and height',
      height, 'has BMI =', BMI)
```

Variables - example 3 (user input)

Goal: Write a program that asks the user for their weight and height and then computes BMI.
How? Use the input(String) function, which prompts the user to enter data, and returns the string that was typed.

```
weight = input('Please enter your weight (in kg):')
height = input('Please enter your height (in m):')
BMI = weight/(height**2)
print('Your BMI is', BMI)
```

Problem: We get a *runtime error*:
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
Use the debugger to see what the type of weight and height is.
They are of type str, because the *input* function always produces a str output, irrespective of what is actually typed by the user.

Converting between types

Python allows data to be converted from one type to another using type conversion functions:

```
int(someObject) # convert someObject to an integer
float(someObject) # convert someObject to a float
str(someObject) # convert someObject to a string
```

Example,

```
name='Yue' # name is a String
weight='66' # weight is a String
height='1.8' # height is a String
weightInt = int(weight) # weightInt is integer 68
heightFloat = float(height) #heightInt is float 1.8
heightInt = int(height) #heightInt is an integer 1
#Note: int() truncates decimal values
nameInt = int(name) # this causes an error, because
# the content of name cannot be converted to number
```

BMI program corrected

We use the type conversion functions to convert the output of the input function to float.

```python
weight = input('Enter your weight (in kg): ')
weightFloat= float(weight)
height = input('Enter your height (in m): ')
heightFloat= float(height)
BMI = weightFloat/(heightFloat**2)
print('Your BMI is ' ,BMI)
```

Or more succinctly, we directly convert the output of the input function to a float, without saving the String in a variable:

```python
weight=float(input('Enter your weight (in kg): '))
height=float(input('Enter your height (in m): '))
BMI = weight/(height**2)
print('Your BMI is ' ,BMI)
```

Conditional execution

What if we want our program to print personalized recommendations to the user, based on the value of their BMI?

- BMI below 18.5 : You are underweight
- BMI between 18.5 and 25: Your BMI is normal
- BMI above 25: You are overweight

We need a way to tell the Python interpreter to execute certain lines of our program only if certain conditions hold.
$\rightarrow$ That's called conditional execution.

Control flow

Until now, every line of our programs was executed exactly once, from top to bottom. This is very limiting!

- Conditionals: we may want to only execute a piece of code if a particular condition holds (e.g. if BMI is low, do something)
- While Loops: We may want to re-use certain pieces of code multiple times (e.g. keep asking someone the same questions until we get the correct answer)
- For Loops: We may want to perform the same operation on a large number of objects (e.g. change every 'T' to an 'A' and every 'G' to a 'C' in a complementary DNA sequence)

This is achieved using control flow instructions. The control flow of a program determines :

- Which part of the code should be executed regardlessly
- Which blocks of code should be executed *only under certain circumstances* (conditional execution, **today lecture**)
- Which blocks of code should be executed repeatedly, and for how many times

Conditionals

We use conditional execution to only execute a block of code if a certain boolean expression is true.

```python
if booleanCondition:
    # this block of code is only executed
    # if booleanCondition is true
else:
    # this block of code is only executed
    # if booleanCondition is false

# this is outside the conditional
# this gets executed no matter what
```

IMPORTANT: In Python, we use indentation (tab character) to indicate what block a line belongs to.

Example 1 : BMI revisited (demo in class)

```
 1 weight = float( input('Please enter your weight: ') )
 2 height = float( input('Please enter your height: ') )
 3 bmi = weight/(height**2)
 4 print('Your BMI is ',bmi)
 5 
 6 if bmi < 18.5 :
 7     print("You are underweight")  # Lines 7 and 8 are only
 8     print("Try to gain weight")   # executed if BMI< 18.5
 9 else:
10     print("You are not underweight")
11 
12 print("Thank you for using the BMI calculator")
```

Notes:

- Lines 7 and 8 form a block of code. They are indented together.
- The block 7-8 only gets executed if BMI < 18.5
- The block 10 only gets executed is BMI is not < 18.5
- Line 12 is outside the conditional; it gets executed after the conditional.

Comparisons

A *comparison* is an operation that compares two objects and produces a *boolean* value. Comparisons are often used as conditions in an if-else statement.

```
my_age = 42
mike_jagger_age = 76
pi = 3.14
dna = 'ACGT'
```

Test equality: double-equal sign

```
my_age == 42 # True
my_age == 43 # False
my_age + 10 == 52 # True
mike_jagger_age == 2 * my_age - 8 # True
age == pi * 13 # False
dna == 'GTCA' # False
dna == 'acgt' # False
```

Comparisons: testing equality

Examples:

```
if my_age==76:
      print("I am the same age as Mick Jagger")

jagger_twice_my_age = mike_jagger_age == 2 * my_age
    # jagger_twice_my_age is a boolean variable

if jagger_twice_my_age:
      print("Wow, Jagger is twice my age!")

if dna=='ATG':
     print("This sequence is a Start codon")

# Remember: = means variable assignment;
# == means equality testing
# So the following is wrong:
if my_age = 43:
      print("Getting old!")
```

Comparisons: testing inequality

```
my_age = 42
mike_jagger_age = 76
pi = 3.14
dna = 'ACGT'
```

Testing non-equality

```
pi != 3.1416 # True
age != 42 # False
```

Greater-than, smaller-than

```
pi < 3.1416 # True
pi > 3.14 # False
pi <= 3.14 # True
'ACGA'< dna #True, because ACGA comes before ACGT
    in alphabetical order
```

Boolan expressions

Boolean variables can be combined to form complex expressions.

Suppose we have two variables a and b, of type boolean.

		Conjunction	Disjunction	Negation
a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Example 2 : BMI re-revisited

```
 1 weight = float( input('Please enter your weight: ') )
 2 height = float( input('Please enter your height: ') )
 3 BMI = weight/(height**2)
 4 print('Your BMI is ',BMI)
 5 
 6 if BMI < 18.5 :
 7     print("You are underweight")
 8     print("Try to gain weight")
 9 
10 if BMI >= 18.5 and BMI <24.9:
11     print("Your weight is normal")
12 
13 if BMI > 24.9:
14     print("You are overweight")
15 
16 print("Thank you for using the BMI calculator")
```

In line 10, we use logical key word "and" to combine two statements "BMI >= 18.5" **and** "BMI < 24.9"

Complex boolean expressions

We can form complex expressions with boolean variables, just like we can form complex arithmetic expressions with int/float.

Suppose we have two variables a and b, of type boolean.

a	b	(a and b) or (not b)
True	True	True
True	False	True
False	True	False
False	False	True

		(a or b) and not (a and b))
True	True	False
True	False	True
False	True	True
False	False	False

Example 2 : BMI re-revisited (a logical mistake)

This is almost the same code, but it won't work properly: why?

```
weight = float( input('Please enter your weight: ') )
height = float( input('Please enter your height: ') )
BMI = weight/(height**2)
print('Your BMI is ',BMI)

if BMI < 18.5 :
    print("You are underweight")
    print("Try to gain weight")

if BMI >= 18.5 and BMI <24.9:
    print("Your weight is normal")
else:
    print("You are overweight")

print("Thank you for using the BMI calculator")
```

Chained conditional

To execute exactly one of several blocks, we can use the if-elif-else structure.

```
if condition1:
    # this is executed only if condition1 is true
elif condition2:
    # this is executed only if condition1 is false and condition2 is true
elif condition3:
    # this is executed only if condition1 is false and confition2 is false and condition3 is true
else:
    # this is executed only if all three conditions are false
```

Example 2 : BMI re-re-revisited

This version works correctly.

```
weight = float( input('Please enter your weight: ') )
height = float( input('Please enter your height: ') )
BMI = weight/(height**2)
print('Your BMI is ',BMI)

if BMI < 18.5 :
    print("You are underweight")
    print("Try to gain weight")
elif BMI >= 18.5 and BMI <24.9:
    print("Your weight is normal")
else:
    print("You are overweight")
    print("Try to loose weight")

print("Thank you for using the BMI calculator")
```

Nested conditionals

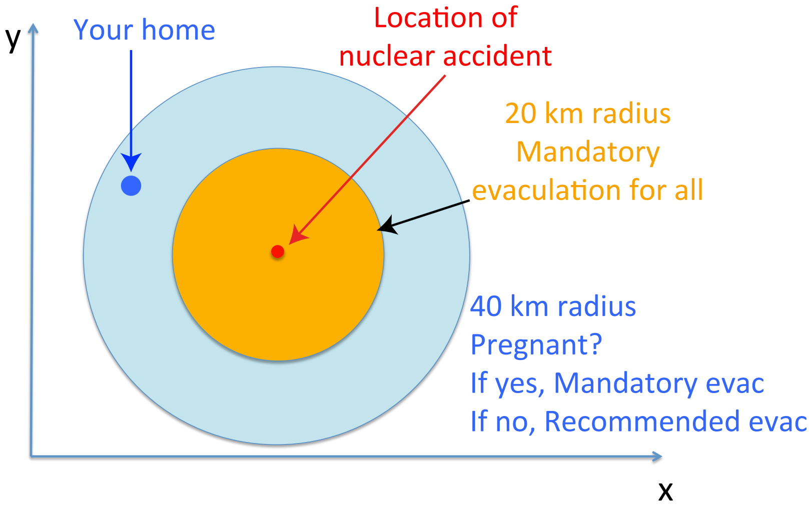
We can have conditionals inside conditionals:

```
if condition1:
    # this is executed only if condition 1 is true
    if condition2:
        # this gets executed only if
        # both conditions 1 and 2 are true
    else:
        # this gets executed only if
        # condition 1 is true but condition 2 is false
else:
    # gets executed only if condition1 is false
    # we could have more if/else here

# this is outside the conditional
# this gets executed no matter what
```

- Note double identation

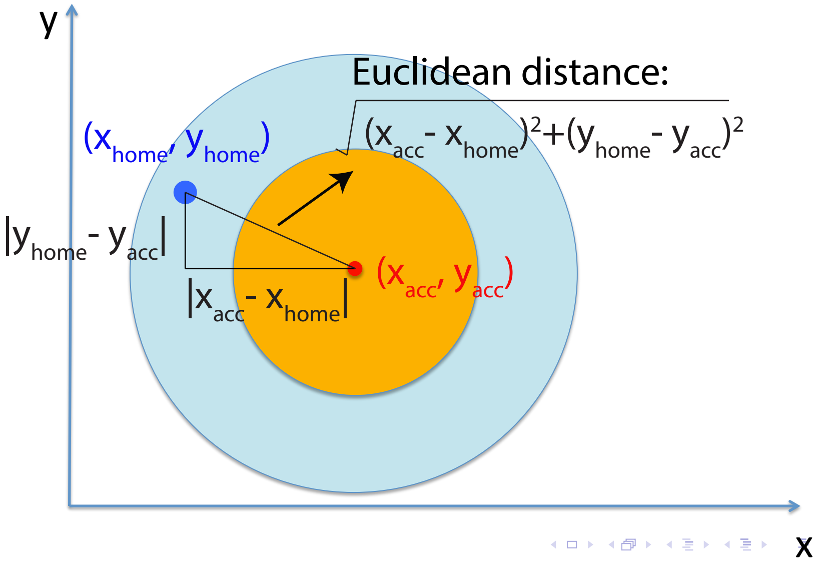
Example 3: Nuclear accident evacuation

Task: Write a program to provide the correct evacuation message following a nuclear accident.



Example 3: Nuclear accident evacuation

Task: Write a program to provide the correct evacuation message following a nuclear accident.



Example 3: Nuclear accident evacuation

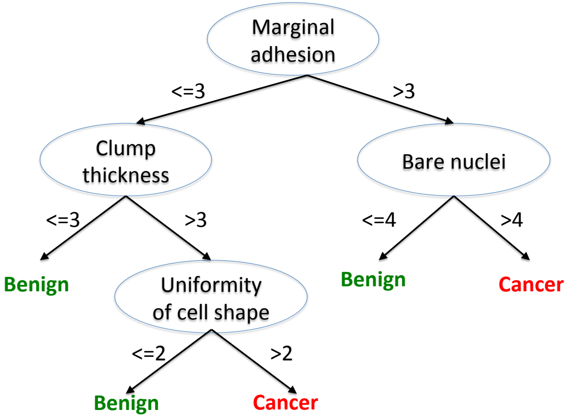
```python
import math  # this imports the math module
xAcc = float(input("Enter x coord. of nuclear accident: "))
yAcc = float(input("Enter y coord. of nuclear accident: "))
xHome = float(input("Enter x coordinate of home: "))
yHome = float(input("Enter y coordinate of home: "))
distance = math.sqrt((xHome - xAcc)**2 + (yHome - yAcc)**2)
if distance <= 20:
    print("You must evacuate")
elif distance <= 40:
    pregnant = input("Are you pregnant? (yes/no) ")
    if (pregnant == "yes"):
        print("You must evacuate")
    else:
        print("Evacuation is recommended")
else:
    print("No need to evacuate")
```

Example 3: Nuclear accident evacuation (flexible answers)

```python
import math  # this imports the math module

xAcc = float(input("Enter x coord. of nuclear accident: "))
yAcc = float(input("Enter y coord. of nuclear accident: "))
xHome = float(input("Enter x coordinate of home: "))
yHome = float(input("Enter y coordinate of home: "))

distance = math.sqrt((xHome - xAcc)**2 + (yHome - yAcc)**2)

if distance <= 20:
    print("You must evacuate")
elif distance <= 40:
    pregnant = input("Are you pregnant? (yes/no) ")
    if (pregnant == "yes" or pregnant == "Yes" or
        pregnant == "Y" or pregnant == "y"):
        print("You must evacuate")
    else:
        print("Evacuation is recommended")
else:
    print("No need to evacuate")
```

Example 4: Tumor classification by decision tree

Task: Write a program to guide doctors in their assessment of tumors.



Example 4: Tumor classification

```python
# the content of this variable
# will be changed by the code below
tumorType=""

adhesion = int(input("Enter marginal adhesion level: ") )
if adhesion<=3:
    clump = int(input("Enter clump thickness: "))
    if clump<=3:
        tumorType="Benign"
    else:
        uniformity = int(input("Enter uniformity of cell
    shape"))
        if uniformity<=2:
            tumorType="Benign"
        else:
            tumorType="Cancer"
else:
    bare = int(input("Enter level of bare nuclei"))
    if bare<=4:
        tumorType="Benign"
    else:
        tumorType="Cancer"
print("The tumor type is: ",tumorType)
```