



COMP 1002/COMP 1001

Lecture 4

Computation II

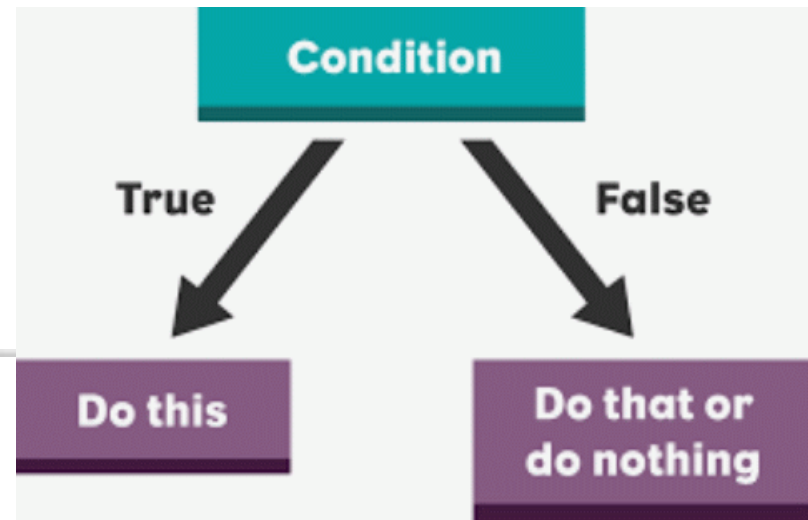
(Reference slides for flipped learning lecture 1)



Topics

- Conditionals (Another type of statement in Pseudo-code)
- Nested if statements
- Simple vs chained conditionals
- Indentation rules in Pseudo-code and Python
- Complex conditionals
- Decision trees

Conditionals



- **if cond then A**

- Check the condition **cond**.
- If it is true, then execute A.
- Otherwise, do nothing.

```
if a < 0 then  
    a = -a
```

```
if a < 0:  
    a = -a
```

```
if value < 0:  
    value = -value
```

- **if cond then A else B**

- Check the condition **cond**.
- If it is true, then execute A.
- Otherwise, execute B.

```
if a < b then  
    answer = a  
else  
    answer = b
```

```
if a < b:  
    answer = a  
else:  
    answer = b
```

```
if a < b:  
    small = a  
else:  
    small = b
```

What are those two
if-conditionals doing?

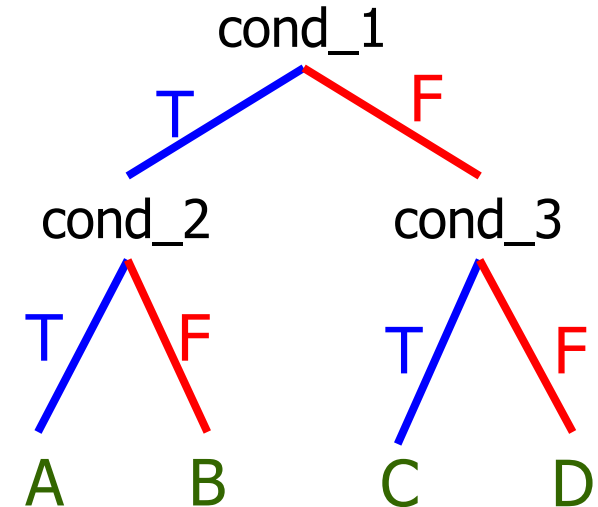
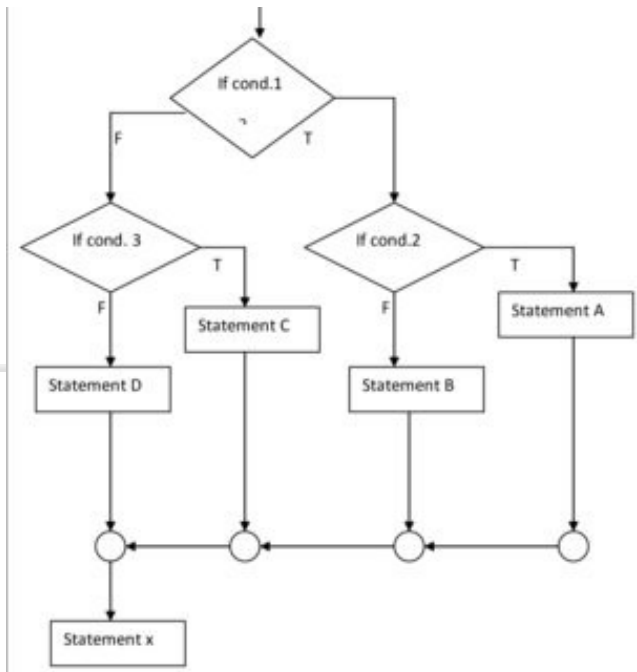
Conditionals

- Conditionals can be **nested**.

```
if cond_1 then
  if cond_2 then A
  else B
else
  if cond_3 then C
  else D
```

- Four types of situations.

- Cond_1 **true** and cond_2 **true** \Rightarrow A
- Cond_1 **true** and cond_2 **false** \Rightarrow B
- Cond_1 **false** and cond_3 **true** \Rightarrow C
- Cond_1 **false** and cond_3 **false** \Rightarrow D



Conditionals

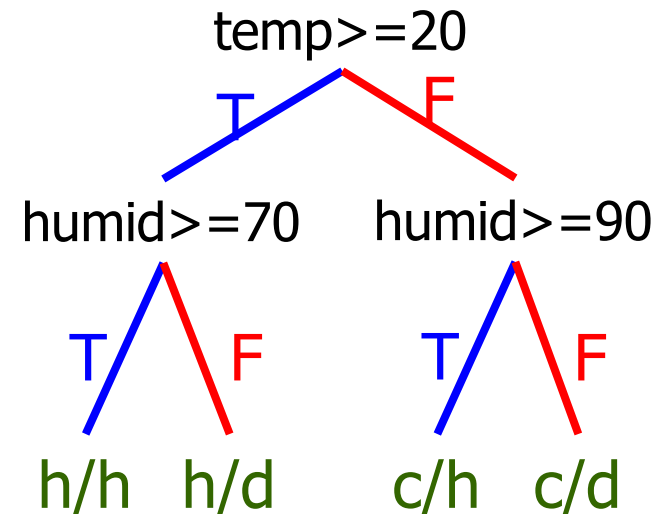
- Example (Pseudo-code)

```
if temperature >= 20 then
  if humidity >= 70 then print
    "hot and humid"
  else
    print "hot and dry" else #
temperature < 20
  if humidity >= 90 then print
    "cold and humid"
  else
    print "cold and dry"
```

- How is today?

Hot Dry	Hot Humid
Cold Dry	Cold Humid

Hot Dry	Hot Humid
Cold Dry	Cold Humid





Conditionals

- Given three unequal numbers, a , b , c , how can the **correct ordering** be printed using conditionals?
- Try writing Pseudo code for this



Conditionals

- Key reminder: Indentation does matter both in Pseudocode and Python

Example

- All employees get a base bonus of \$100.
- If an employee's sales exceed \$10,000, they get an *additional* \$500 bonus.
- If their sales also exceed \$20,000, they get a *further* \$1000 bonus on top of everything else.

```
base_bonus = 100
sales = 15000
total_bonus = base_bonus
if sales > 10000:
    total_bonus = total_bonus + 500
    if sales > 20000:
        total_bonus = total_bonus + 1000
print("Total Bonus:", total_bonus)
```



Conditionals

- List of conditions
 - Count the number of times for each face of a die.

```
reset count1 to count6 value to zero
for i in [1..1000] do
  face = throw a die
  if face = 1 then count1 = count1 + 1
  if face = 2 then count2 = count2 + 1
  if face = 3 then count3 = count3 + 1
  if face = 4 then count4 = count4 + 1
  if face = 5 then count5 = count5 + 1
  if face = 6 then count6 = count6 + 1
```

- Each condition is **exclusive** from each other.



Conditionals

- List of conditions
 - Count the number of grades in a class.

```
reset countA to countF value to zero
for each student s in class list L do
  g = score of student s
  if g >= 85 then countA = countA + 1
  if g >= 70 then countB = countB + 1
  if g >= 55 then countC = countC + 1
  if g >= 40 then countD = countD + 1
  if g < 40 then countF = countF + 1
```
 - Each condition is **not exclusive** from each other!

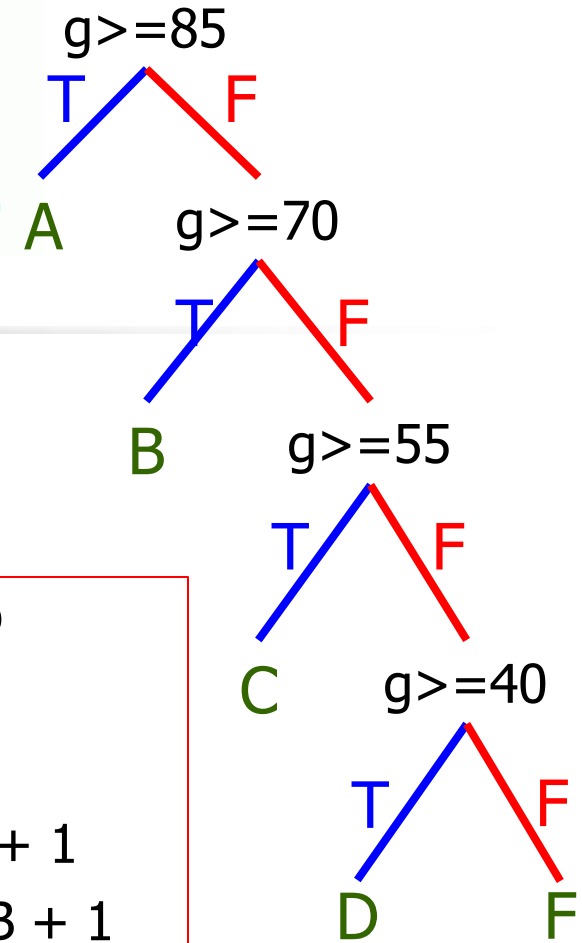


Conditionals

- List of conditions
 - Change the pseudo code in previous slide so conditions becomes **exclusives**

Conditionals

```
if g >= 85:  
    print("A")  
elif g >= 70:  
    print("B")  
else:  
    print("others")
```



- Chained conditions

- Shorter syntax in Python-style.

reset countA to countF value to zero

for each student s in class list L do

g = score of student s

if g >= 85 then countA = countA + 1

elif g >= 70 then countB = countB + 1

elif g >= 55 then countC = countC + 1

elif g >= 40 then countD = countD + 1

else countF = countF + 1

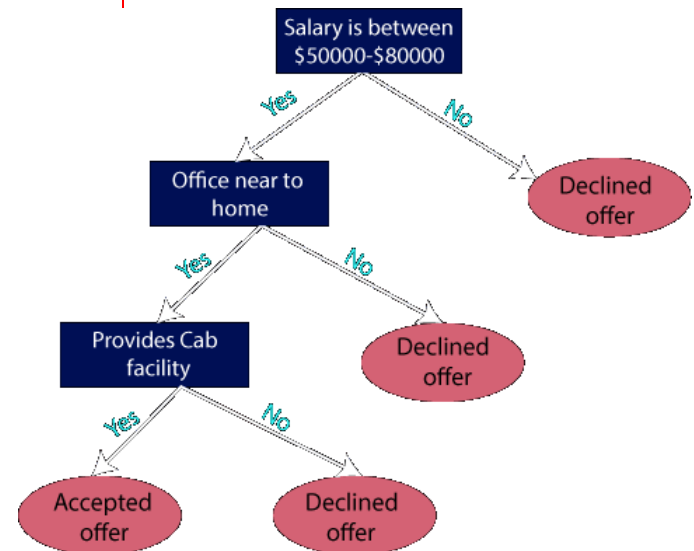
- **else if** can be replaced by **elif**.
 - All branches on right side (false).

Making **decision** or doing **classification**
Straightforward, no guessing!

Complex Conditionals

- **Nested conditions** could best represent complex **multi-way decisions**.
- One very common decision structure is called **decision tree**, often obtained through **data mining**.

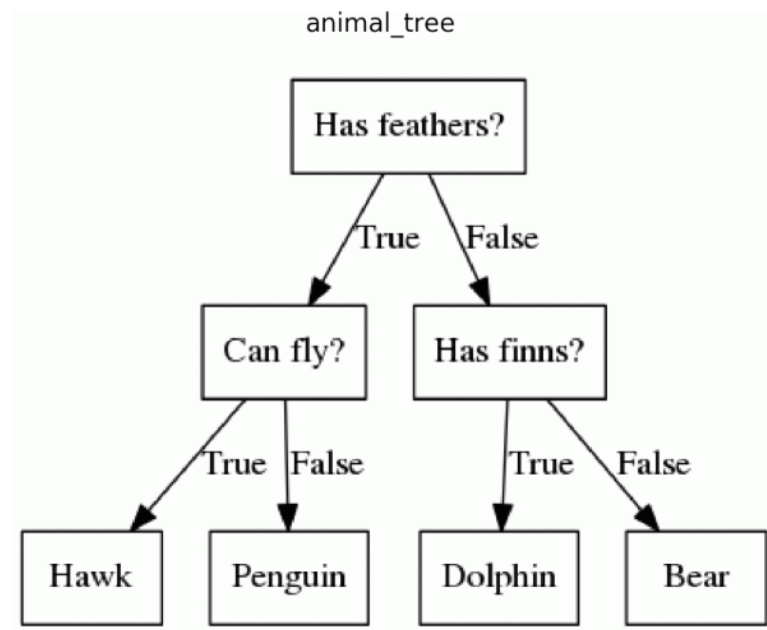
```
get salary S, distance D, transportation T
if 50000 <= S <= 80000 then
  if D is near home then
    if T is provided then
      accept offer
    else
      decline offer
  else
    decline offer
else
  decline offer
```

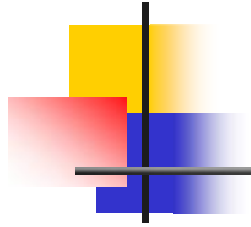


Complex Conditionals

- Another **decision tree**

```
if animal has feathers then
  if animal can fly then
    return hawk
  else
    return penguin
else
  if animal has fins then
    return dolphin
  else
    return bear
```





End of Lecture 4