Fundamentals of Programming 1

Lecture 4

Aurelia Power, ITB, 2018

Control Structures

- **Control flow** is the order that the instructions are executed in a program/script.
- **Control structure** is a set of instructions and control statements that control execution.
- **Control statement** is a statement that determines the control flow of a set of instructions.
- All programs are written in terms of **three control structures**:
    - Sequence structure
    - Selection structure
    - Repetition/Iterative structure.

- Every program is formed by using one, two or all control structures: the sequence, selection and repetition, depending on the problem the program implements.

Sequence Control Structure

- All our programs up until now have been executed in **sequence,** i.e. **one instruction after the next**, from top to bottom.

- **Sequential control is built into Python** → unless directed otherwise, the computer executes python statements one after the other in the order in which they're written.

- We can have as many instructions as needed in a sequence.

EXAMPLE:

```
age = input ("Enter your age:  ")      ←——————  first executed
print('you are', age, 'years old')     ←——————  second executed
print(" See you later !!");            ←——————  last executed
```

Transfer of Control and the Selection Structure

- There are numerous instances when the sequence order is **too restrictive** and we want **more control** over the order in which the instructions are executed.

EXAMPLE: we want to print different messages, depending on the age the user entered; in that case we must transfer the control flow from one action/statement to another → This is called **transfer of control**.

- So, programs often need to deal with alternative situations and make ***choices.***

ANOTHER EXAMPLE: a program processing requests for airline tickets could have the **following choices** to make:

- display the price of the seats requested;
- display a list of alternative flights;
- display a message saying that no flights are available to that destination.

Selection structure allows such choices to be made in programs.

Let's consider again the age example…

- How can we make the program print an additional message, depending on the age entered?
- **For instance**, if the age entered is over or equal to18, it should also print "you can get a driving licence…"
- So there should be 2 possible outputs …

```
Enter you age: 10
You are 10 years old
… (the rest of the instructions being executed)
```

first possible output

```
How old are you? 25
You are 25 years old
You can get a driving licence…
… (the rest of the instructions being executed)
```

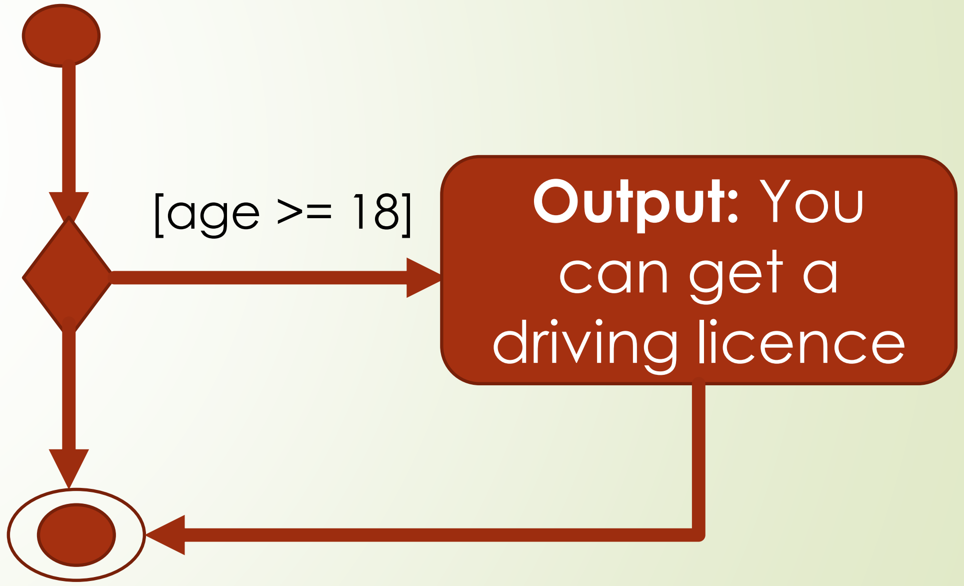
second possible output

Let's write the pseudocode and represent visually the choice that must be made for the age portion of the program...

```
PROMPT user to enter age and store it
in variable age ( so we can access it
in the program whenever we need)

OUTPUT message with the value of age

IF age is greater or equal to 18
    BEGIN
      OUTPUT "You can get a driving
                licence…"
    END
```



The Python syntax for single selection – the if statement

- When the program has only one selection/choice to make it is called **single selection**;
- The selection/ is made based on whether a certain **boolean condition – the guard** is satisfied:
  - If the boolean condition is **true** then the program branches out to execute the instructions that correspond to that course of action;
  - If the boolean condition is **false**, then the program ignores the instructions associated with that course of action (it skips them), and moves to the next statement.

```
if (condition):
   statement(s) to be executed
```

The Python syntax for single selection – the if statement

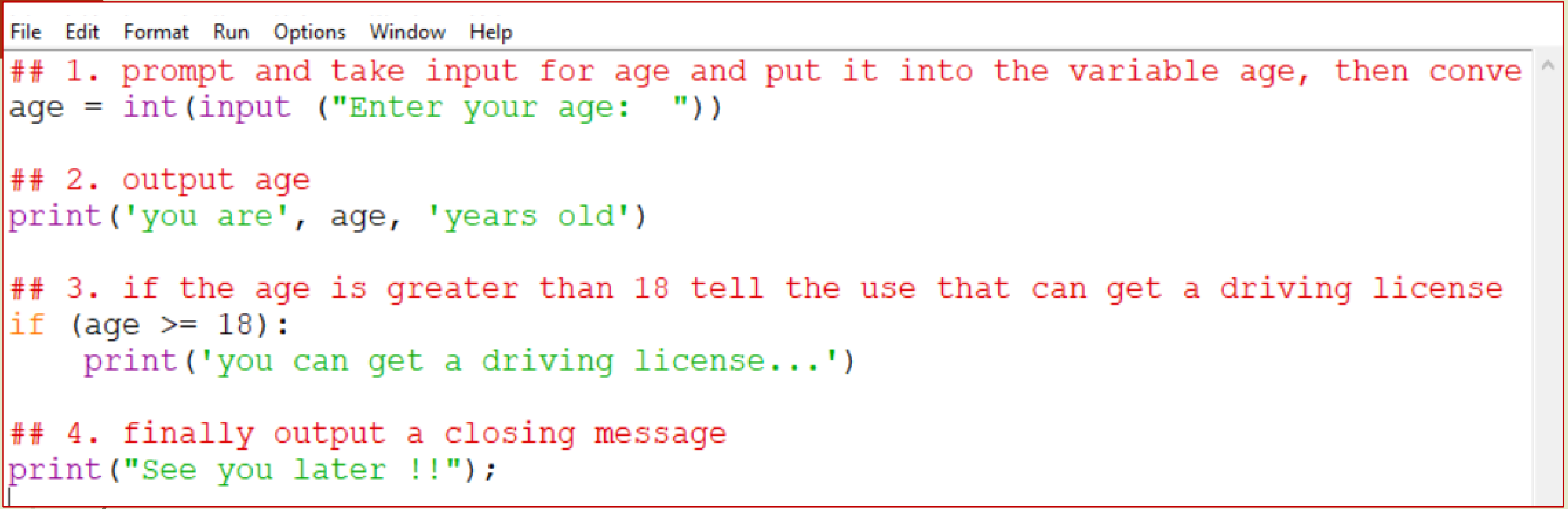
```
if (age >= 18):
   print("you can get a driving license…")
```

NOTES:

- The parentheses that enclose the Boolean condition are optional.
- The condition must be followed by a colon :
- The statements/instructions that are to be selectively executed must be indented one level… so indentation is the way the interpreter groups statements.

```
if party == 'Yes':
   print("Count me in ☺ !")
```

Putting it all together in code



```
## 1. prompt and take input for age and put it into the variable age, then conve
age = int(input ("Enter your age:  "))

## 2. output age
print('you are', age, 'years old')

## 3. if the age is greater than 18 tell the use that can get a driving license
if (age >= 18):
    print('you can get a driving license...')

## 4. finally output a closing message
print("See you later !!");
```

2 possible interactions:

1. The user enters an age that's greater than 18, say 25
2. The user enters an age that's less than 18, say 16

2 possible outputs:

1. Enter your age: 25
   you are 25 years old
   you can get a driving license
   See you later
2. Enter your age: 16
   you are 16 years old
   See you later

Let's go back to the age issue…

- So far we are able to display a selective message only when the age is greater or equal than 18;
- What if we also want a different message for users under 18?
- **For instance**, if the age entered is over or equal to18, it should print "you can get a driving license…", but if the age is smaller than 18, it should print "you are underage…"
- Again, there are 2 ways the program can interact…

```
Enter you age: 10
You are 10 years old
You are underage…
See you later!!
```

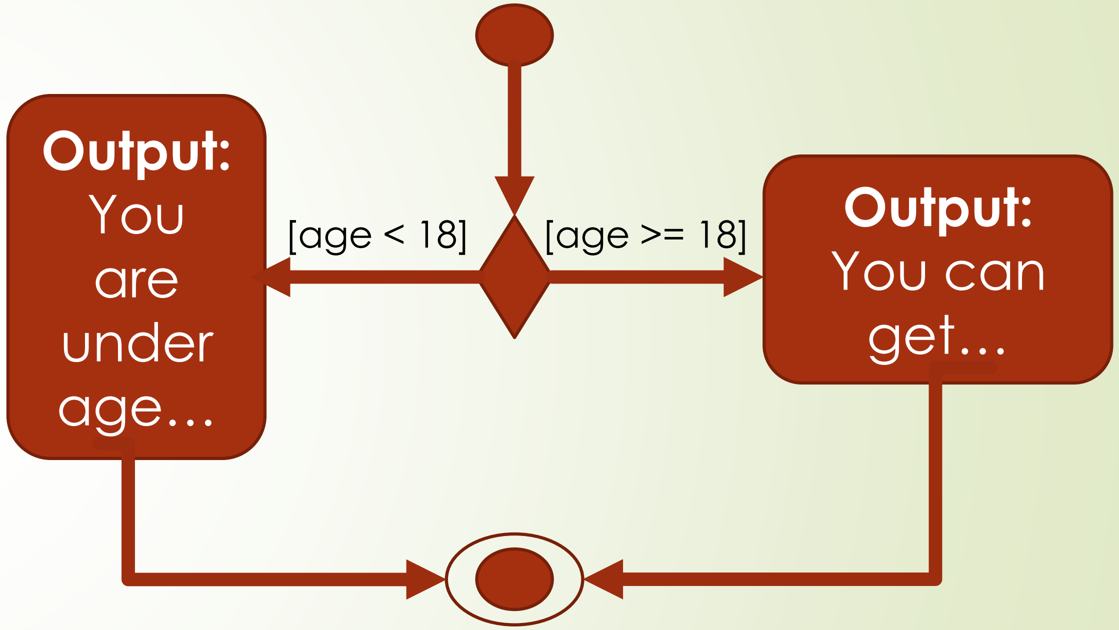
first way of program interaction

```
How old are you? 25
You are 25 years old
You can get a driving license
See you later!!
```

second way of program interaction

Let's write the pseudocode and represent visually the choice that must be made…

```
PROMPT user to enter age and STORE
user input in variable age
OUTPUT message with the value of age
IF age is greater or equal to 18
    BEGIN
      OUTPUT "You can get a driving
               license."
    END
ELSE
   BEGIN
      OUTPUT "You are an under age…"
    END
OUTPUT "See you later!!"
```

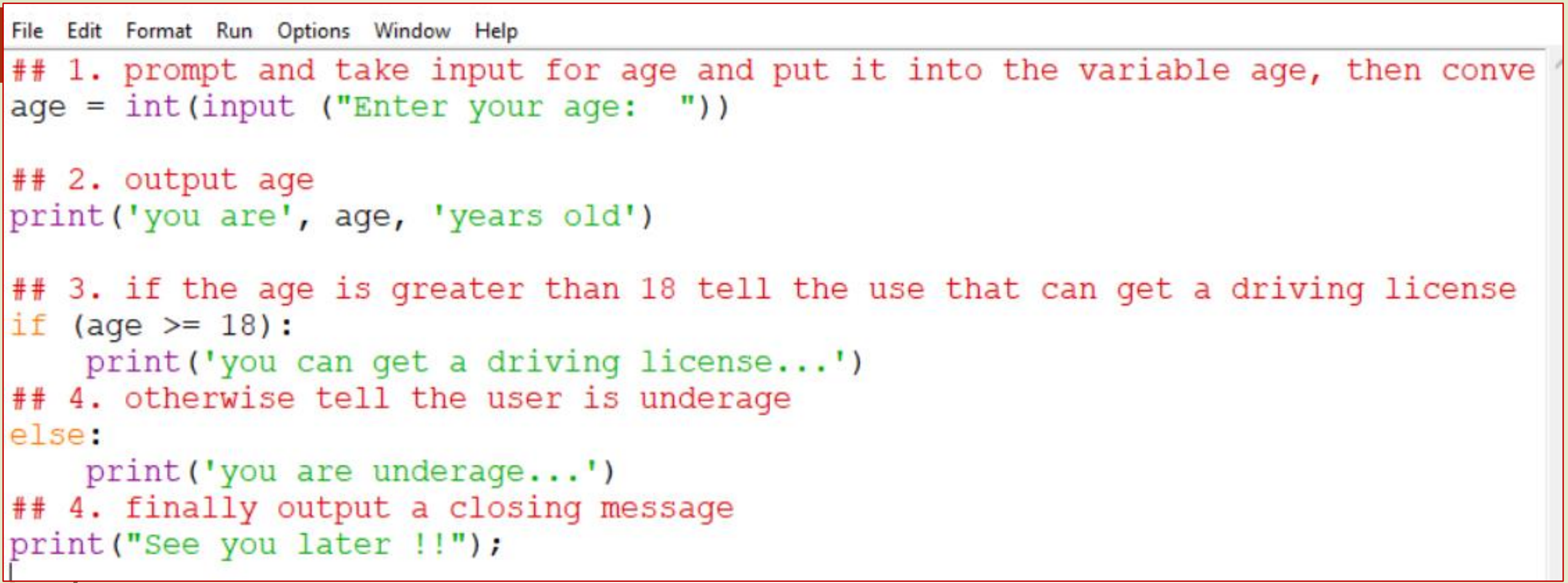


The Python syntax for double selection – the if…else statement

- This form of selection is called **double selection** because it involves two courses of action: one for when the age >= 18, and one for when the age < 18;
- It also makes use of a **Boolean condition – the guard**.
- If the Boolean condition is **true** then the program branches out to execute the instructions in the if block, associated with the first course of action.
- If the Boolean condition is **false**, then the program branches out to execute the instructions in the else block, associated with the other course of action.
- It will never execute both branches, only one.

```
if (condition):
   conditional instruction(s)/statement(s) go here
else:
   conditional instruction(s)/statement(s) go here
```

Putting it all together in code



```
## 1. prompt and take input for age and put it into the variable age, then conve
age = int(input ("Enter your age:  "))

## 2. output age
print('you are', age, 'years old')

## 3. if the age is greater than 18 tell the use that can get a driving license
if (age >= 18):
    print('you can get a driving license...')
## 4. otherwise tell the user is underage
else:
    print('you are underage...')
## 4. finally output a closing message
print("See you later !!");
```

Possible output for when age is greater or equal to18

```
Enter your age:  28
you are 28 years old
you can get a driving license...
 See you later !!
```

Possible output for when age is smaller than 18

```
Enter your age:  12
you are 12 years old
you are underage...
See you later!!
```

Your turn …

- What will the following blocks of code output if the angle is 160 ?
- What will the following blocks of code output if the angle is 82 ?

```
if (angle == 90):
   print("This is a right angle")
```

```
if (angle == 90):
   print("This is a right angle")
else:
   print("This is not a right angle")
```

Multiple if statements – multiple alternatives

- We can have multiple if statements one after another working with the same value to specify multiple alternative courses of action;
- For instance, we want to print different messages for different grades:

```
if (grade >= 80):
        print(“it’s  an A”)
if (grade < 80):
        print(“it’s  a B”)
if (grade < 60):
       print(“it’s  a C”)
if (grade < 40)
       print(“it’s  a D”)
if (grade < 35)
        print(“it’s  an F”)
```

Assume I got 25 in my FOP1 exam and that the **grade** variable will point to that; what will the script output??

```
it’s  a B
it’s  a C
it’s  a D
it’s  an F
```

It will output almost all grades… so how can we fix it????

- When using multiple **if** statements we must be careful of how we formulate our Boolean conditions.

How to address the issue of multiple if statements – solution 1

- We can formulate our Boolean conditions so that they specify both ends of a range, using **relational operators** and the **AND** or **OR conditional operators**;
  - **A** is between 80 and 100 inclusive; $\rightarrow$
  - **B** is between 60 and 79 inclusive; $\rightarrow$
  - **C** is between 40 and 59 inclusive; $\rightarrow$
  - **D** is between 35 and 39 inclusive; $\rightarrow$
  - **F** is anything below 35; $\rightarrow$

```
if (grade >= 80 and grade <= 100):
   print("it's  an A")

if (grade >= 60 and grade < 80):
   print("it's  a B")

if(grade >= 40 and grade < 60) :
   print("it's  a C")

if (grade >= 35 and grade < 40 ):
   print ("it's  a D")

if (grade > =0  and grade < 35 ):
   print("it's  an F")
```

Now if my grade is 25, it will output "it's an F"

Your turn…

1. Consider the following if statement to compute a discounted price:

```
if (originalPrice > 100) :
    discountedPrice = originalPrice - 20
else:
    discountedPrice = originalPrice - 10
```

What is the discounted price if the original price is **95**? **100**? **105**?

2. Assume **a = 3;** what will the next block of code output?

```
if(a == 3):
    print (“a equals to 3”)
if(a == 2+1):
    print (“a equals to 2+1”)
if(a+1 == 2+1):
    print (“a  + 1 equals to 2 + 1”)
if(a+1 >= 3):
    print (“a + 1 greater or equal 3”)
```

3. Write a selection structure with two branches that sets **n** to 1 if x is positive, and to –1 if x is negative (assume x has already been given a value).

Your turn…

1. What is the value of each variable after the if statement?

a.
```
n = 1; k = 2; r = n
if (k < n):
    r = k
```

b.
```
n = 1; k = 2; r = ‘hello’
if (n < k):
    r = k
else:
    r = k + n
```

c.
```
n = 1; k = 2; r = k
if (r < k):
    n = r
else:
    k = n
```

d.
```
n = 1; k = 2; r = 3
if (r < n + k):
    r = 2 * n
else:
    k = 2 * r
```

Your turn…

2. What do these code fragments print?

a.
```
n = 1; m = -1
    if (n < -m):
        print(n)
    else:
        print(m)
```

b.
```
n = 1;  m = -1
    if (-n >= m):
        print(n)
    else :
        print(m)
```

c.
```
n = 0;  m = 3
    if (n + 3 != m):
        print(n)
    else :
        print(m)
```

2. What do these code fragments print?

a.
```
n = '3'; m = -1
    if (n == m):
        print(n)
    else:
        print(m)
```

b.
```
n = 'e';  m = 'tree'
    if (n not in m):
        print(n)
    else :
        print(m)
```

c.
```
n = 2;  m = 'a'
    if (n * m == 'aa'):
        print(n)
    else :
        print(m)
```