Introduction to Computer Programming Lecture 2.3:

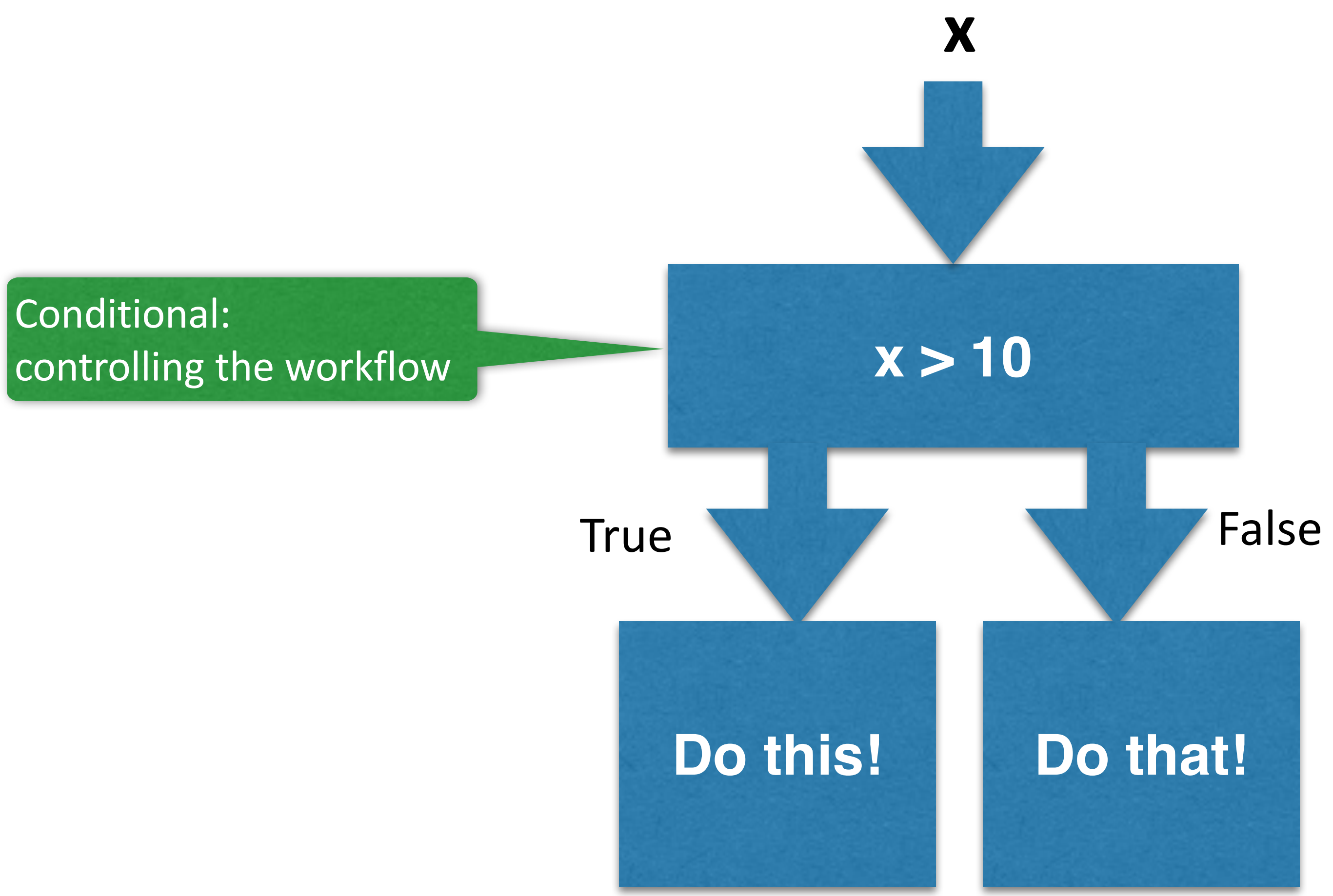
Control Flow

Hemma Philamore

Department of Engineering Mathematics

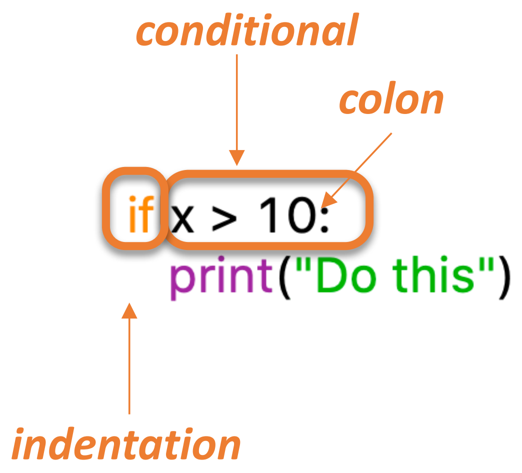
Controlling the flow

Conditional statements run different blocks of code depending on whether a Boolean condition evaluates to **true** or **false**.

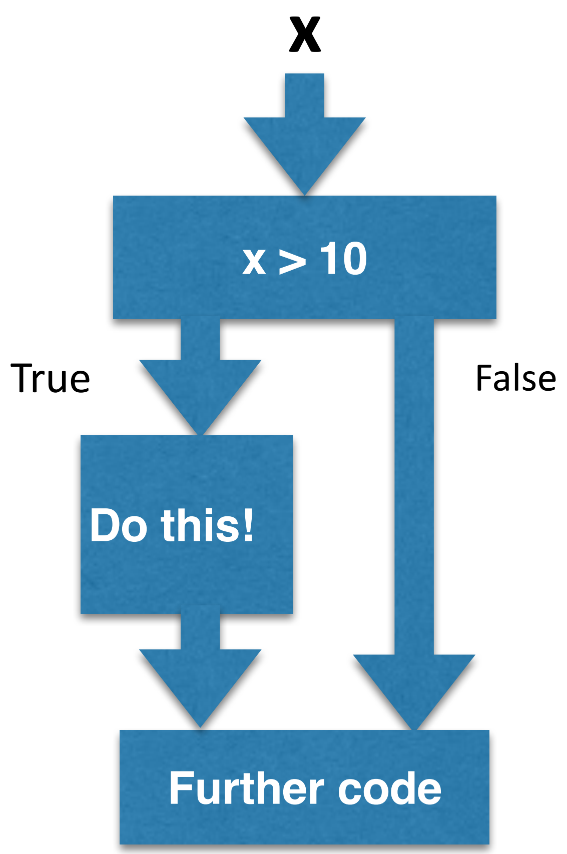


Conditional Statements

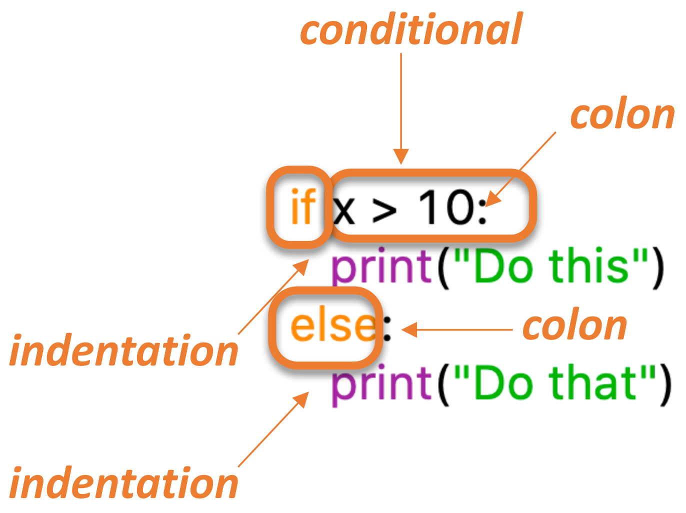
If… then…: Runs the block of code only if the condition is true.



```
if x > 10:
    print("Do this")
```

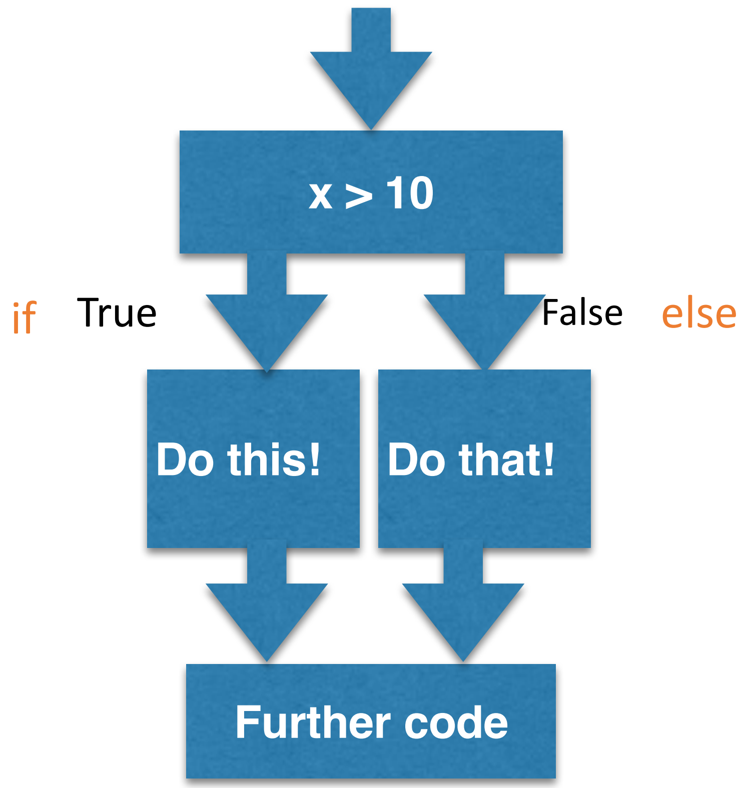


Conditional Statements



```
if x > 10:
    print("Do this")
else:
    print("Do that")
```

If… then… else…: Runs the block of code under "then" only if the condition is true. Runs the "else" code otherwise.



Conditional Statements

```
a = 4
b = 10
c = 15
```

indentation defines blocks

```
if a < b:
  print("a is smaller than b")
```

inside the block

outside the block

```
if b > a and b < c:
  print("b is right in the middle")
print("This I will do after the conditional block")
```

nested conditions

```
if a == b:
  print("They are the same")
else:
  print("They are NOT the same")
  if a > b:
    print("a is bigger")
  else:
    print("a is smaller")
```

We can check if an alternative to the `if` statement is true using an `else if` statement.

```
if A is true
    Perform task X (only)

else if B is true
    Perform task Y (only)
```

Often it is useful to include an `else` statement.

If none of the `if` and `else if` statements are satisfied, the code following the `else` statement will be executed.

```
if A is true
    Perform task X (only)

else if B is true
    Perform task Y (only)

else
    Perform task Z (only)
```

Chain of Conditional Statements

```
Sally = "Happy"
Ben = "Sad"

if Sally == "Happy" and Ben == "Happy":
  print("What a wonderful world")
elif Sally == "Happy" or Ben == "Happy":
  print("At least one is happy")
else:
  print("Meh....")
```

Conditional statements allow you to execute a block of code based on a condition.

Use boolean operators to compare two values: **==, !=, <=, >=, >, <**.

Boolean logic is especially useful here: **and, or**.

Time-telling program

Based on the current time of day, the program answers two questions:

> **Is it lunchtime?**

> `True`

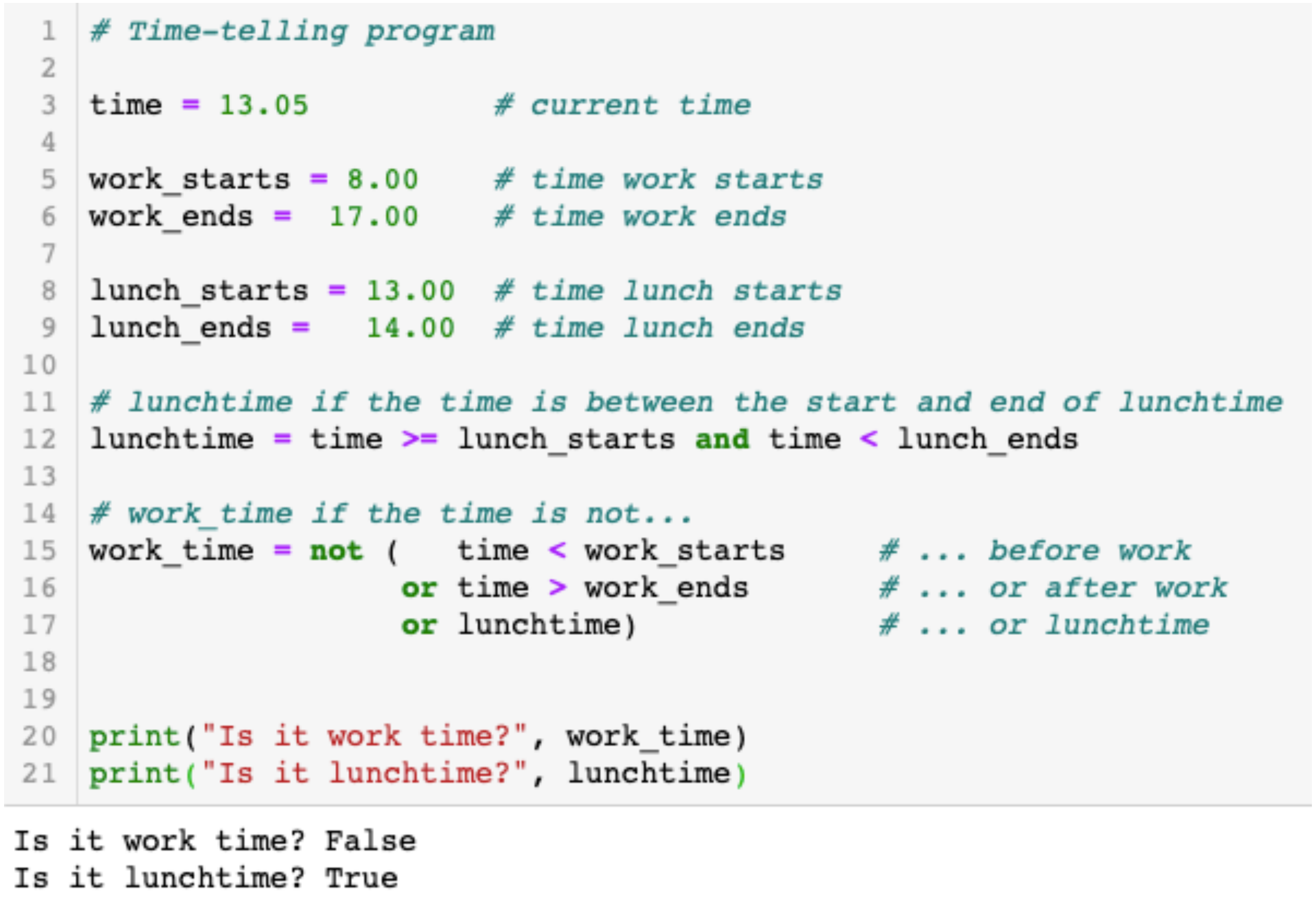
if it is lunch time.

> **Is it time for work?**

> `True`

if it is `not`:

- before work (`time < work_starts`)
- after work (`time > work_ends`)
- lunchtime (the previous question assigns the value `True` or `False` to variable `lunchtime`).



```python
# Time-telling program

time = 13.05          # current time

work_starts = 8.00    # time work starts
work_ends =  17.00    # time work ends

lunch_starts = 13.00  # time lunch starts
lunch_ends =   14.00  # time lunch ends

# lunchtime if the time is between the start and end of lunchtime
lunchtime = time >= lunch_starts and time < lunch_ends

# work_time if the time is not...
work_time = not (   time < work_starts     # ... before work
                 or time > work_ends       # ... or after work
                 or lunchtime)             # ... or lunchtime


print("Is it work time?", work_time)
print("Is it lunchtime?", lunchtime)
```

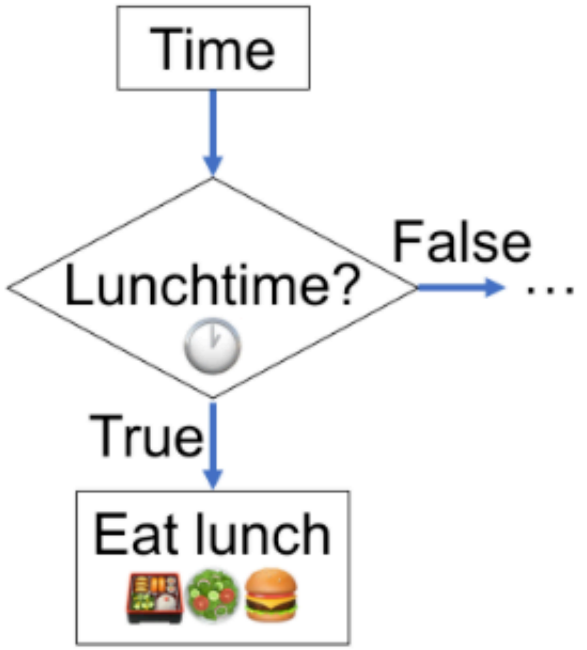
```
Is it work time? False
Is it lunchtime? True
```

What if we now want our computer program to do something based on these answers?

To do this, we need to use *control statements*.

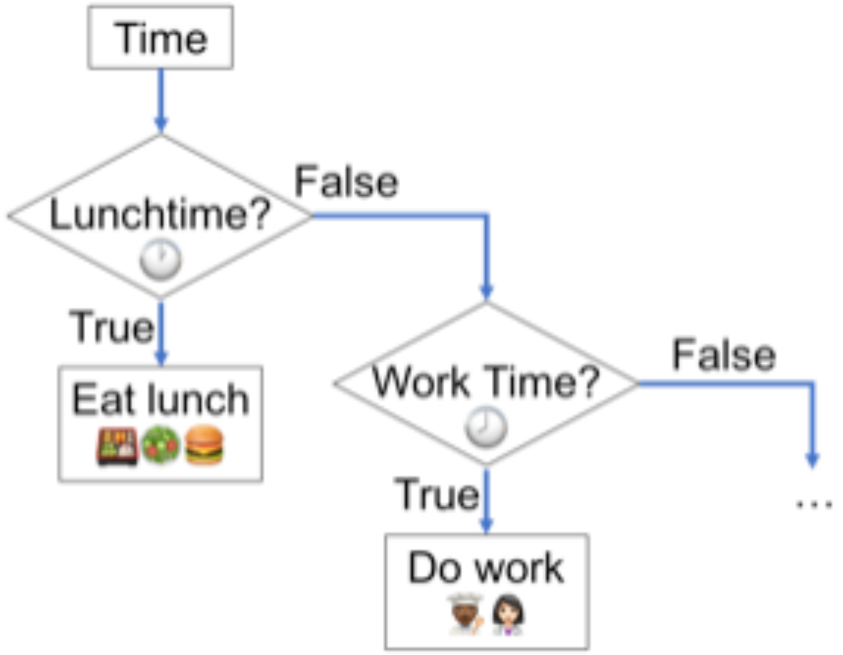
Control statements allow us to make decisions in a program.

```
if lunchtime is true
    Eat lunch
```



We can check if an alternative to the `if` statement is true using an `else if` statement.

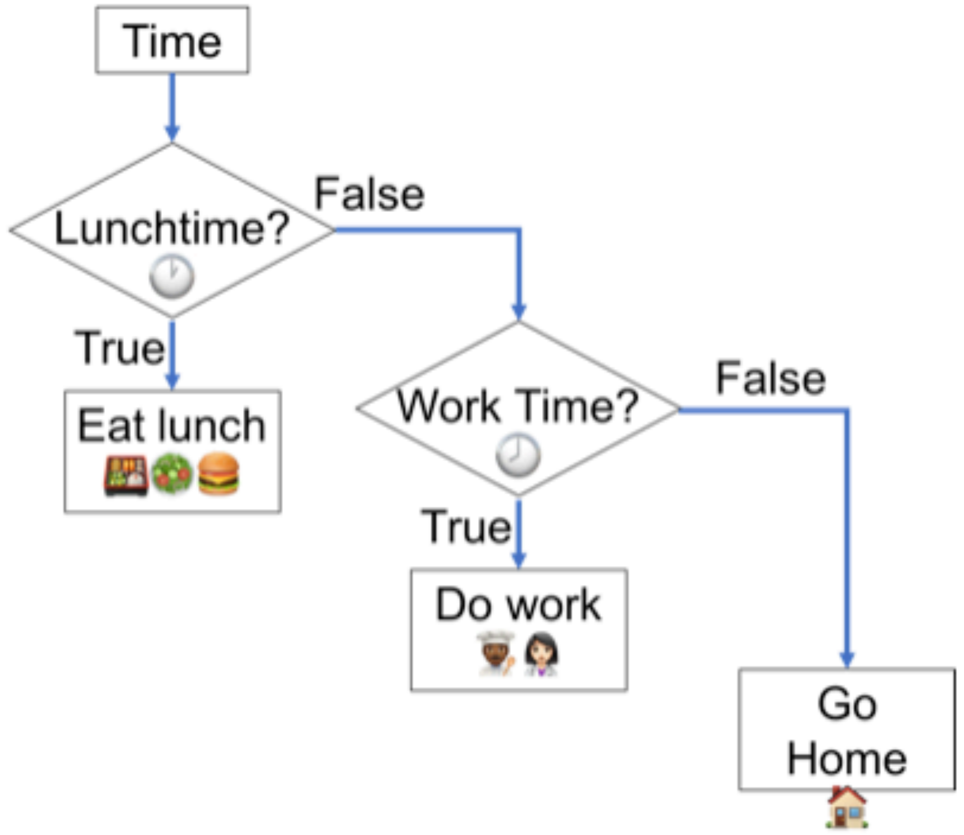
```
if lunchtime is true
    Eat lunch

else if work_time is true
    Do work
```

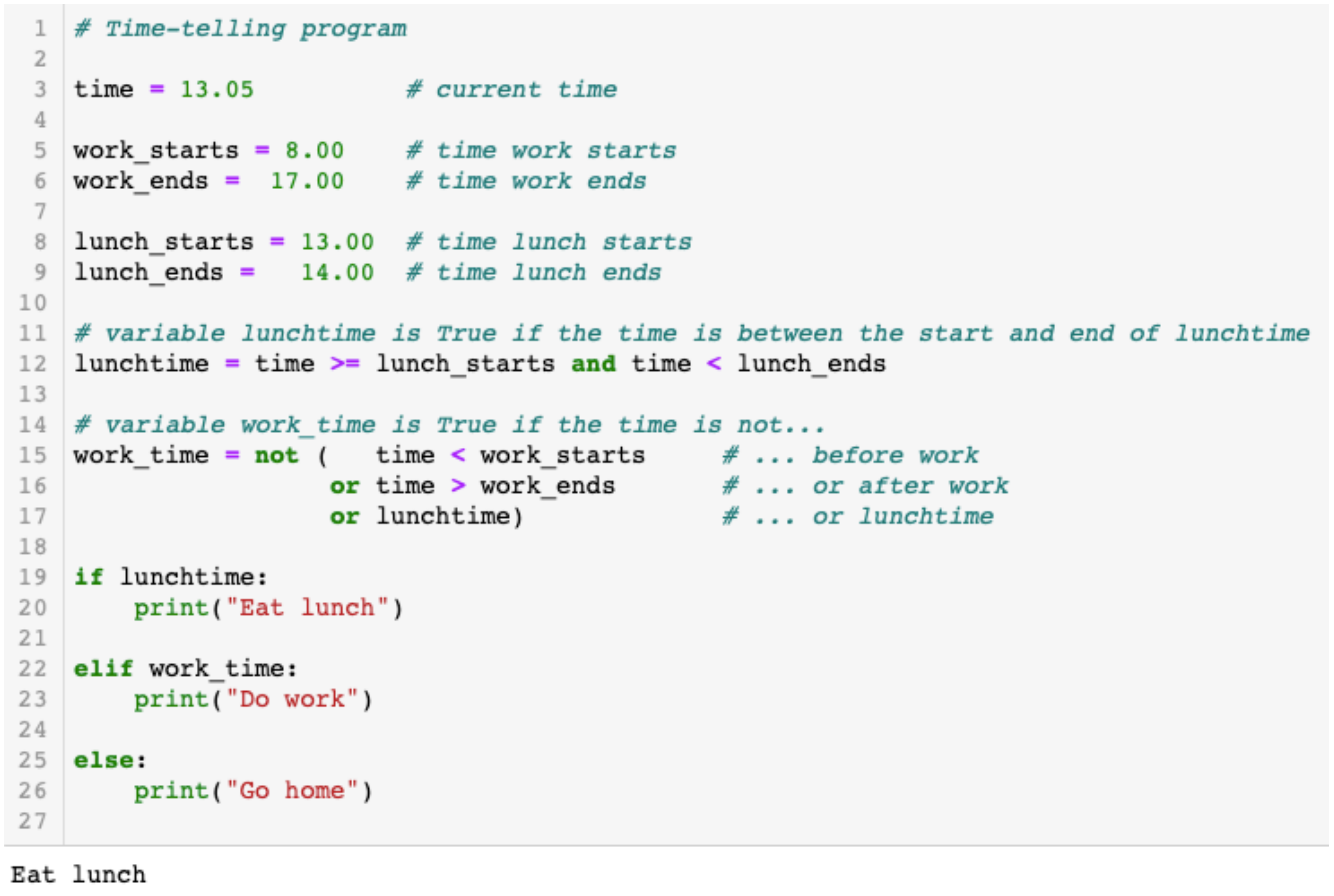


Often it is useful to include an `else` statement.

If none of the `if` and `else if` statements are satisfied, the code following the `else` statement will be executed.

```
if lunchtime is true
    Eat lunch

else if work_time is true
    Do work

else
    Go home
```



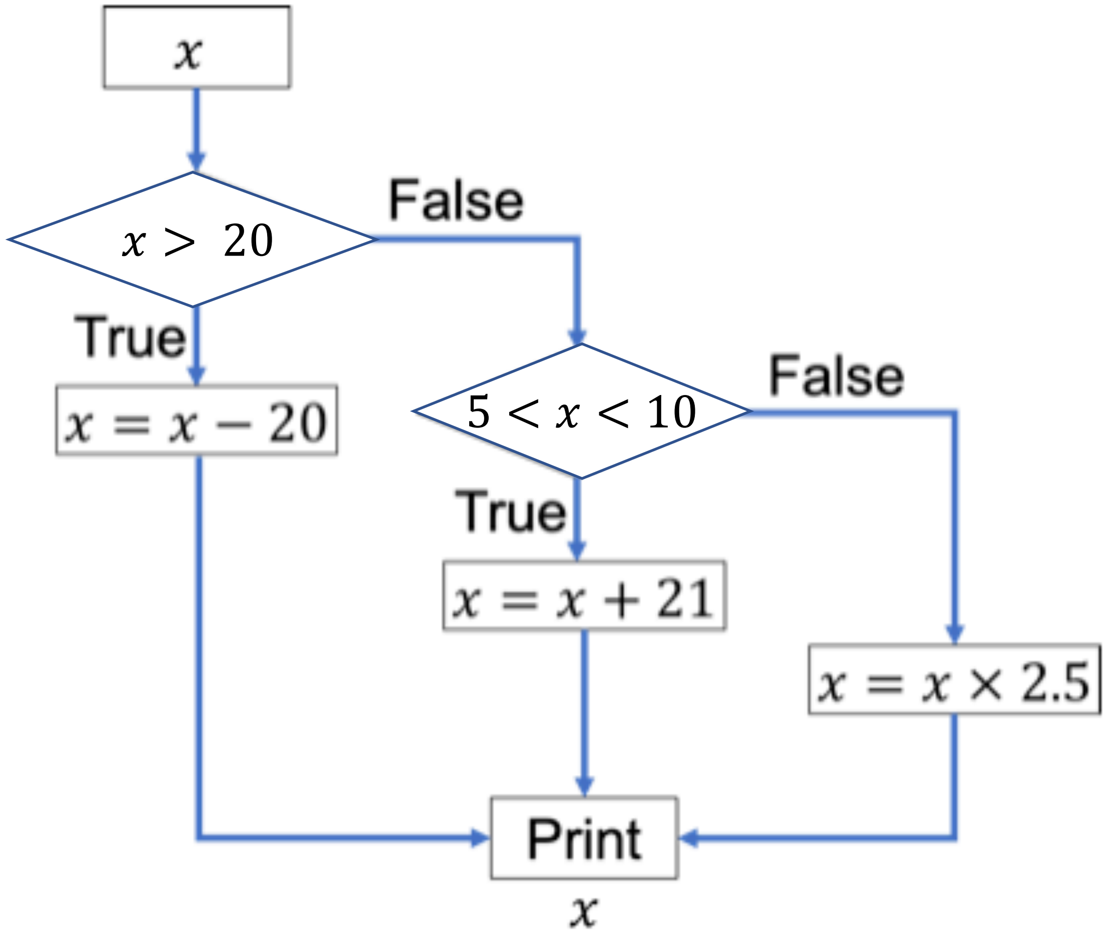


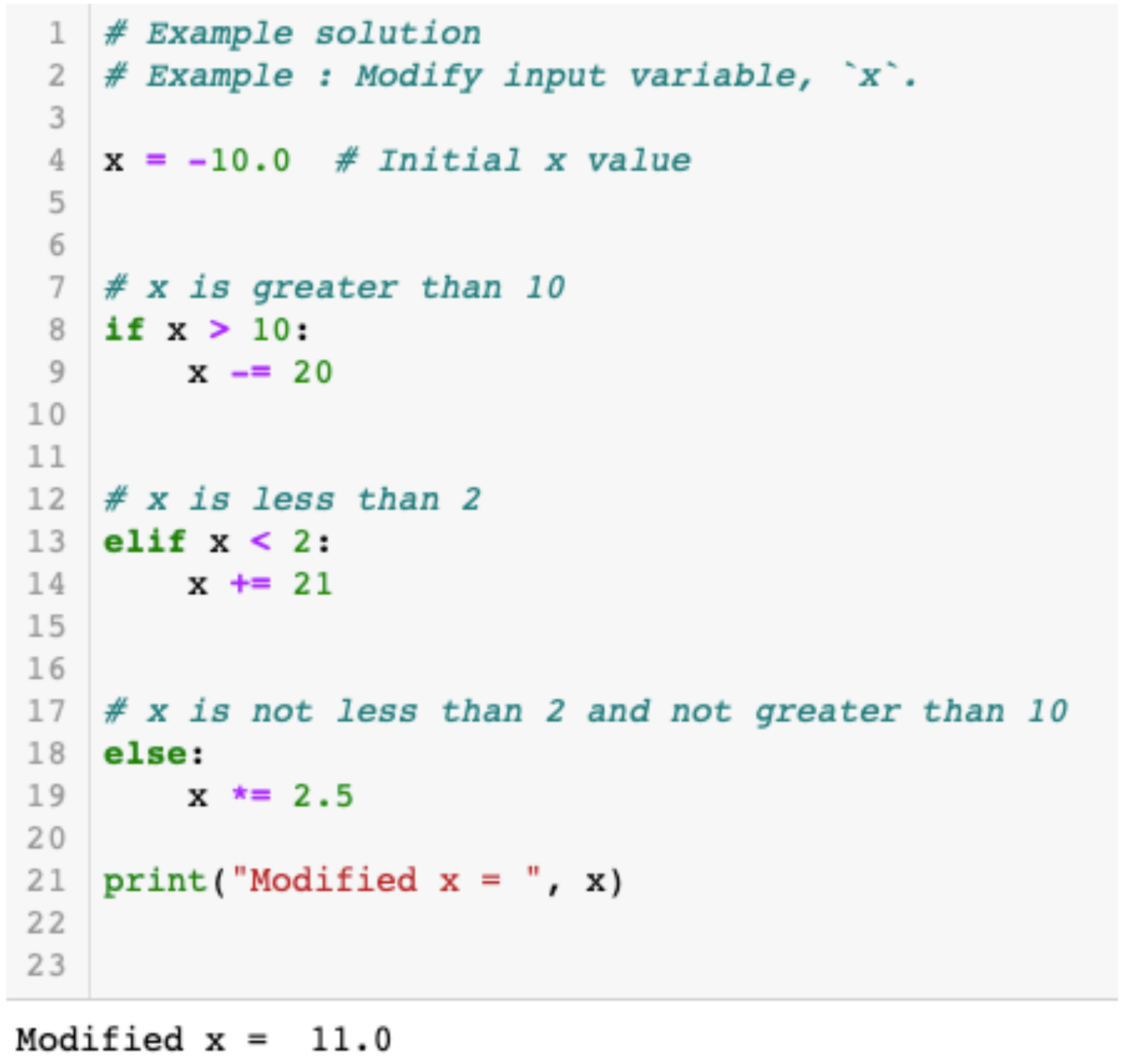
```python
# Time-telling program

time = 13.05          # current time

work_starts = 8.00    # time work starts
work_ends =  17.00    # time work ends

lunch_starts = 13.00  # time lunch starts
lunch_ends =   14.00  # time lunch ends

# variable lunchtime is True if the time is between the start and end of lunchtime
lunchtime = time >= lunch_starts and time < lunch_ends

# variable work_time is True if the time is not...
work_time = not (   time < work_starts     # ... before work
                 or time > work_ends       # ... or after work
                 or lunchtime)             # ... or lunchtime

if lunchtime:
    print("Eat lunch")

elif work_time:
    print("Do work")

else:
    print("Go home")

```

```
Eat lunch
```

Here is another example, using algebraic operators to modify the value of an initial variable, `x`.

The **modification of** `x` and the **message printed** depend on the initial value of `x`.





```python
# Example solution
# Example : Modify input variable, `x`.

x = -10.0  # Initial x value


# x is greater than 10
if x > 10:
    x -= 20


# x is less than 2
elif x < 2:
    x += 21


# x is not less than 2 and not greater than 10
else:
    x *= 2.5

print("Modified x = ", x)


```

```
Modified x =  11.0
```

Summary

Control Flow

- The Python `if` keyword performs a conditional test on an expression for a Boolean value of True or False.
- Alternatives to an `if` test are provided using `elif` and `else` tests.