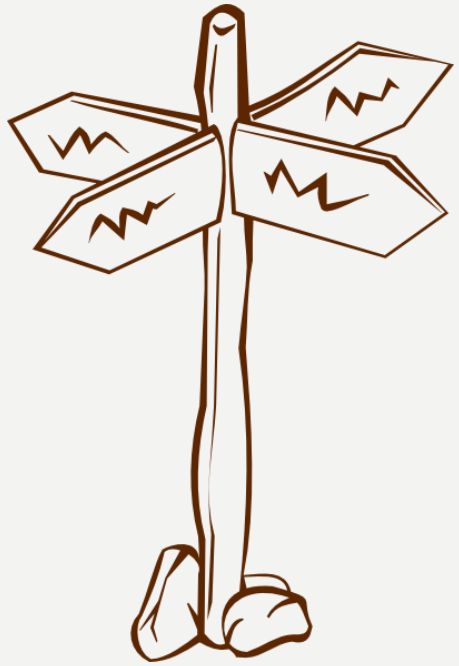
CONTROL FLOW



WHAT DOES FLOW OF CONTROL MEAN?

- Flow of control denotes in which order instructions and statements are executed or evaluated when a program is running
- Lets take an example of **control flow** from the "real" world
  - Dexter has just taken his final exam in programming and his future depends on the grade he will get from that exam
  - **If** he passes the test **then** he can graduate and go work for Google as a data scientist
  - However, if he fails, he won´t be able to graduate and has to take the programming course again!

WHAT DOES FLOW OF CONTROL MEAN?

- Conclusion
  - Whether Dexter will graduate and go work for Google, or have to take the course again depends on his grade. In other words, the grade **controls** the **flow** of Dexter‘s future
- You can think of it like this:
  - **If** Dexter´s grade is greater than 4.5 **then** he will graduate and go work for Google

    Else he won´t graduate and has to take the course again

IF – ELSE STATEMENTS

- **If** statements can help us as programmers to **control the flow** of a program
- An **if statement** will always need to evaluate a condition to decide which statement to execute (Like in Dexter‘s case, passing the exam was the condition he needed to satisfy in order to graduate)
- This condition is called a **boolean expression,** and just like the **bool** datatype, boolean expressions can only be **True** or **False**

– Lets try and express Dexter´s situation in code on the next slide, but to do that we must introduce **if – else statements**!

IF – ELSE STATEMENTS

A variable to hold Dexter‘s grade

A variable to hold Dexter‘s grade as a float

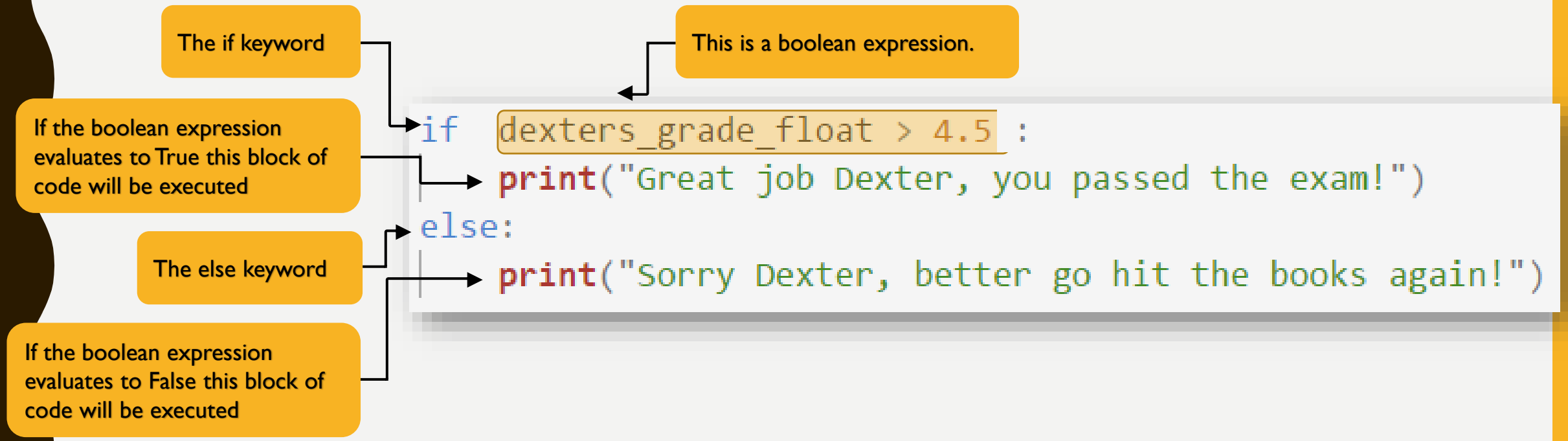
If Dexter´s grade is greater than 4.5 this code will be executed

Otherwise, this code will be executed

```
1  dexters_grade_str = input("Please enter the grade: ")
2  dexters_grade_float = float(dexters_grade_str)
3
4  if  dexters_grade_float > 4.5 :
5      print("Great job Dexter, you passed the exam!")
6  else:
7      print("Sorry Dexter, better go hit the books again!")
```

IF – ELSE STATEMENTS

- Lets take a closer look at the if-else statement from the previous slide



IF – ELSE STATEMENTS

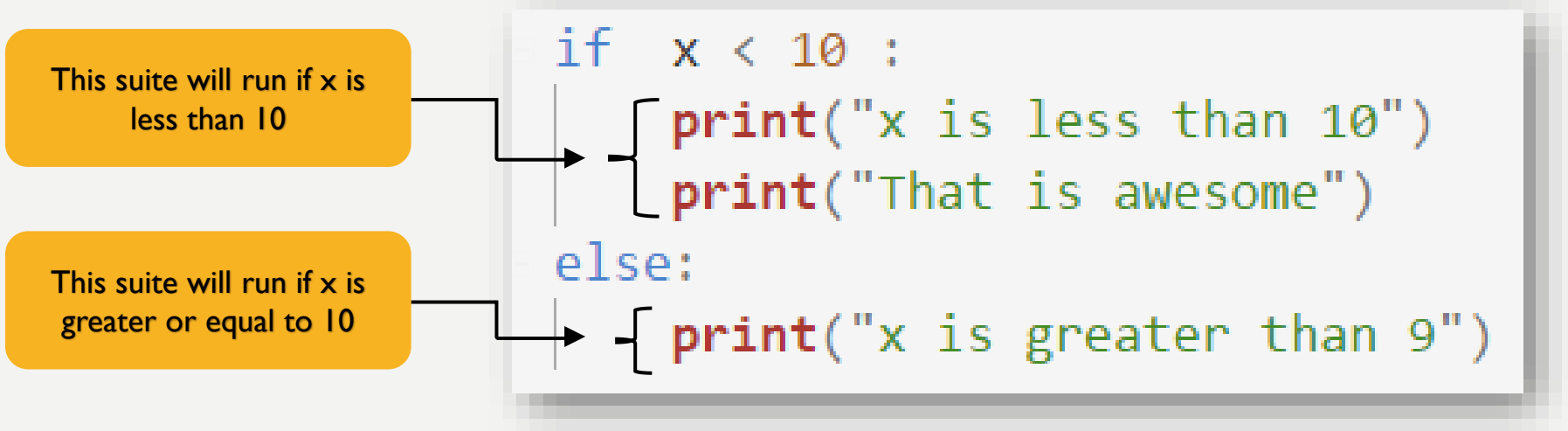
Notice the colon, if statements and else statements must end with colon. In the following line the code that should be executed follows

Notice that the code within the if statement is indented. This is mandatory for the code to run

```
if  dexters_grade_float > 4.5 :
    print("Great job Dexter, you passed the exam!")
else:
    print("Sorry Dexter, better go hit the books again!")
```

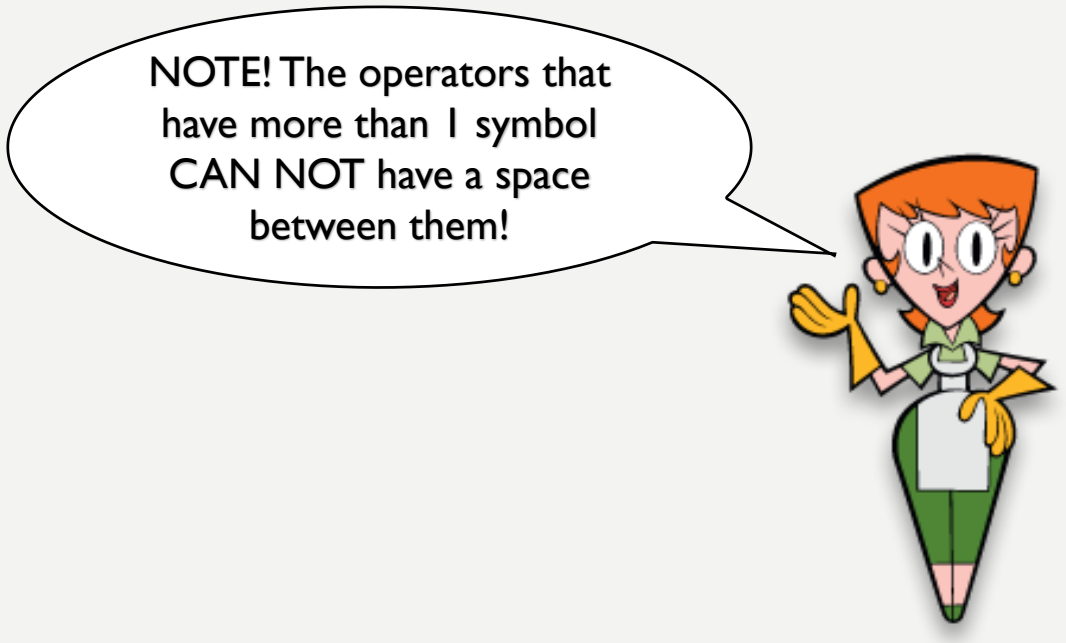
INDENTATION

- The code within an if block is often called a **suite**
- Elements of the suite must all be indented the same number of spaces/tabs
- Python only recognizes suites when they are indented the same distance (**the *standard is 4 spaces***)
- You must be careful to get the indentation right to get suites right.



BOOLEAN OPERATORS

- Here are some of the boolean operators:
  - **>** Greater than
  - **<** Less than
  - **<=** Less than or equal to
  - **>=** Greater than or equal to
  - **==** equal to
  - **!=** not equal to



BOOLEAN EXPRESSIONS AND BOOLEAN OPERATORS

- Examples of boolean expressions using the boolean operators from the previous slide
  - 4 > 65 evaluates to **False**
  - 1 < 20 evaluates to **True**
  - 3 == 3 evaluates to **True**
  - 3 != 3 evaluates to **False**
  - 45 >= 45 evaluates to **True**
  - -4 <= 10 evaluates to **True**

PITFALL
THE '==' OPERATOR VS. THE '=' OPERATOR

- It is very common to confuse the ' = ' operator with the ' == ' operator so it deserves a whole slide!
- Remember that ' = ' is the **assignment operator (gildisveitingavirkinn)**
  - we use it when we assign values to variables
- ' == ' is the equality operator
  - We use it to compare values, like this:

```
if  x == 4.5 :
```

- This is however a very common error :

```
if  x = 4.5 :
```

  - But thankfully Python will produce an error which is not the case in many other programming languages

MORE IF ELSE EXAMPLES

- If statements do not **necessarily** need to be followed by an else statement
- Example
  - Joey is getting himself some ice cream.
    he inputs how many scoops he wants but if it's his birthday he'll get an extra scoop for free!

```
1  is_joeys_birthday = True
2  number_of_scoops = input("How many scoops of ice cream? ")
3  number_of_scoops_int = int(number_of_scoops)
4
5  if  is_joeys_birthday == True :
6      print("Happy Bday, you get an extra birthday scoop!")
7      number_of_scoops_int += 1
8
9  print("Joey got", number_of_scoops_int, "scoops!")
```

THE 'AND' OPERATOR

- Boolean expressions can be combined into more complex expressions with the **'and'** operator
- The **'and'** operator takes two operands. One on the right hand side and the other on the left hand side. These operands are often boolean expressions themselves.
- The **'and'** operator only evaluates to **True** if both operands are **True**

```
1   age_str = input("How old are you? ")
2   age_int = int(age_str)
3
4   if  age_int > 12 and age_int < 20 :
5       print("So you are only a teenager")
```

This Boolean statement is made up of two Boolean statements.
The first one is (age > 12) and the other one is (age < 20). They must both evaluate to True for the body of the if statement to execute.

THE 'OR' OPERATOR

- Boolean expressions can also be combined into more complex expressions with the 'or' operator
- The **'or'** operator takes two operands. One on the right hand side and the other on the left hand side. These operands are often boolean expressions themselves.
- The **'or'** operator evaluates to **True** if both or one of the operands evaluates as **True**

```
1  user_pick_str = input("Pick a number between 10 and 20 : ")
2  user_pick_int = int(user_pick_str)
3
4  if  user_pick_int < 10 or user_pick_int > 20 :
5      print("Invalid input")
6  else:
7      print("Great job! You picked a number between 10 and 20")
```

THE 'OR' OPERATOR

- Lets look at an example of the or operator

Lets say that I run this program and input the number 3

First 3 < 10 is evaluated as True

Next 3 > 20 is evaluated as False

Then finally True or False evaluates as True so the program print Invalid input

```
1  user_pick_str = input("Pick a number between 10 and 20 : ")
2  user_pick_int = int(user_pick_str)
3
4  if  user_pick_int < 10 or user_pick_int > 20 :
5      print("Invalid input")
6  else:
7      print("Great job! You picked a number between 10 and 20")
```

THE 'OR' OPERATOR

- Lets take a look at what happens with a different input

Lets say that I run this program and input the number 12

First 12 < 10 is evaluated as false

Next 12 > 20 is evaluated as False

Then finally False or False evaluates as False so the program executes the else statement

```
1  user_pick_str = input("Pick a number between 10 and 20 : ")
2  user_pick_int = int(user_pick_str)
3
4  if  user_pick_int < 10 or user_pick_int > 20 :
5      print("Invalid input")
6  else:
7      print("Great job! You picked a number between 10 and 20")
```

BOOLEAN PRECEDENCE RULES

- Expressions inside parentheses are always evaluated first
- Then the default precedence of operations is:
  - Perform relational operations such as <, >, <=, >= !=, == next
  - Perform **'and'** operations next
  - Perform 'or' operations last

MORE FLOW OF CONTROL

- We‘ve seen if-else statements and they can come in very handy when we only have 2 options.
- For example if you´re at a theme park and the roller coaster has a height limit. There are only two options
  - Either you‘re tall enough and can ride the roller coaster
  - Or you‘re not...
- But what if we need more options ?
  - Lets examine that on the next slide

This is a classic real world example in code of a situation that has only two options

```
1  height_str = input("Please enter your height: ")
2  height_float = float(height_str)
3
4  if  height_float >= 1.5 :
5      print("Yes, you may ride the roller coaster!")
6  else:
7      print("Sorry...maybe next year you'll be tall enough.")
```

THE ELIF STATEMENT

- If you have more than 2 options in a program that depends on a validation of some variable or variables you can use the **elif** statement which is short for else if
- This sample program here makes use of **elif**
- This program converts a numeric grade to a alphabetic grade
- Lets analyze the flow of this program on the next slide

```
1  numeric_grade_str = input("Enter numeric grade here: ")
2  numeric_grade_float = float(numeric_grade_str)
3  alphabetic_grade = ""
4
5  if numeric_grade_float >= 8 and numeric_grade_float <= 10:
6      alphabetic_grade = "A"
7
8  elif numeric_grade_float >= 6 and numeric_grade_float < 8:
9      alphabetic_grade = "B"
10
11 elif numeric_grade_float >= 4 and numeric_grade_float < 6:
12     alphabetic_grade = "C"
13
14 else:
15     alphabetic_grade = "F"
16
17 print("The alphabetic grade is:", alphabetic_grade)
```

THE ELIF STATEMENT

- The flow of this program is as follows

First this Boolean expression is evaluated. If and only if it evaluates as false will the next Boolean expression be evaluated.

This expression will only be evaluated if the previous Boolean expression was **False.**

This expression will only be evaluated if all previous Boolean expressions were **False.**

If all Boolean expression evaluated as false the code in the body of the else statement will be executed.

```
1   numeric_grade_str = input("Enter numeric grade here: ")
2   numeric_grade_float = float(numeric_grade_str)
3   alphabetic_grade = ""
4
5   if numeric_grade_float >= 8 and numeric_grade_float <= 10:
6       alphabetic_grade = "A"
7
8   elif numeric_grade_float >= 6 and numeric_grade_float < 8:
9       alphabetic_grade = "B"
10
11  elif numeric_grade_float >= 4 and numeric_grade_float < 6:
12      alphabetic_grade = "C"
13
14  else:
15      alphabetic_grade = "F"
16
17  print("The alphabetic grade is:", alphabetic_grade)
```