

Making decisions with code

if statements

Susan Ibach | Technical Evangelist
Christopher Harrison | Content Developer

Every day we are faced with decisions

- Should I drive or take the bus?
- Should I cook at home or go out for dinner?
- Which laptop should I buy?

The choice we make depends on different conditions

- Should I drive or take the bus?
 - Am I late? What's the price of gas?
- Should I cook at home or go out for dinner?
 - Do I have any food at home? Do I have enough money to go out?
- Which laptop should I buy?
 - How much RAM do I need? How much money do I have?

If your code is going to solve problems, it has to make decisions as well

- If the user maintained a bank account balance over \$1000 waive the transaction fees
- If a user cancels their appointment less than 24 hours before the appointment time, charge a cancellation fee
- If the hockey player gets the puck in the net, add one to the score

If statements allow you to specify code that only executes if a specific condition is true

```
answer=input("Would you like express shipping?")  
if answer == "yes" :  
    print("That will be an extra $10")
```

What do you think the == symbol means?

You can use different symbols to check for different conditions

<code>==</code>	is equal to	<code>if answer == "yes" :</code>
<code>!=</code>	is not equal to	<code>if answer != "no" :</code>
<code><</code>	is less than	<code>if total < 100 :</code>
<code>></code>	is greater than	<code>if total > 100 :</code>
<code><=</code>	is less than or equal to	<code>if total <= 100 :</code>
<code>>=</code>	is greater than or equal to	<code>if total >= 100 :</code>

If statements allow you to specify code that only executes if a specific condition is true

```
answer=input("Would you like express shipping? ")  
if answer == "yes" :  
    print("That will be an extra $10")  
    print("Have a nice day")
```

Does it matter if that print statement is indented?

YES – the indented code is only executed if the condition is true

DEMO

if statements

Real world if statements

Almost every if statement can be written two ways

```
if answer == "yes" :  
if not answer == "no" :
```

```
if total < 100 :  
if not total >= 100 :
```

Which do you prefer?

Write it the way you would say it

- If course is completed – send certificate to student
– `if courseCompleted == "yes" :`
- If order total under \$50 – add shipping
– `if total < 50 :`
- If cat has not been vaccinated – call owner to set appointment
– `if not vaccinated == "yes" :`

What do you think will happen if we type "YES" instead of "yes"

```
answer=input("Would you like express shipping? ")  
if answer == "yes" :  
    print("That will be an extra $10")  
print("Have a nice day")
```

One of the challenges of working with strings of characters, is that the computer considers "y" and "Y" to be two different letters.

Is there a way we could change a string from uppercase to lowercase?

```
answer=input("Would you like express shipping?")
if answer.lower() == "yes" :
    print("That will be an extra $10")
print("Have a nice day")
```

Hint: There were functions we could call for string variables

Hint: `lower()`

What if we try an if statement with numbers instead of strings

```
deposit = 150
if deposit > 100 :
    print("You get a free toaster!")
print("Have a nice day")
```

What will appear on the screen if deposit is 150?

What will appear on the screen if deposit is 50?

What will appear on the screen if deposit is exactly 100?

DEMO

Working with numeric values and if statements

Always test $>$, $<$ and boundary conditions

```
deposit = 150
if deposit > 100 :
    print("You get a free toaster!")
print("Have a nice day")
```

So when you test this code, try:

- a value less than 100

- a value greater than 100

- exactly 100

Handling user input

DEMO

Asking the user for a numeric value to use in an if statement

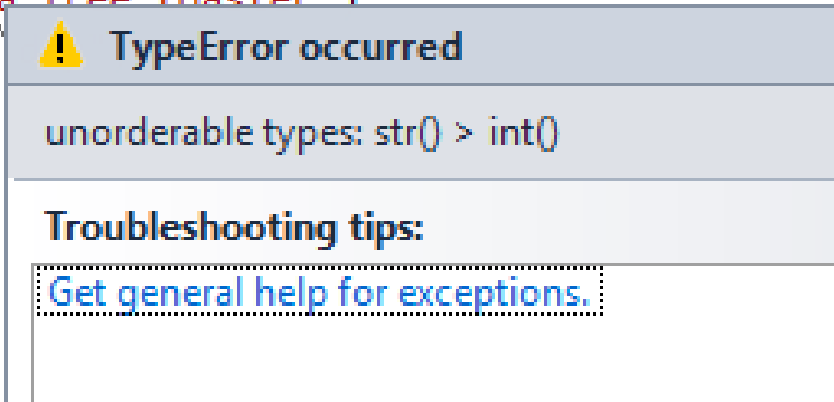
How could we let the user enter the amount to deposit?

```
deposit=input("How much would you like to deposit? ")
if deposit > 100 :
    print("You get a free toaster!")
print("Have a nice day")
```

Why did our code crash?

How can we fix it?

```
deposit = input("how much would you like to deposit? ")
if deposit > 100 :
    print("You get a free toaster")
print ("Have a nice day")
```



! TypeError occurred

unorderable types: str() > int()

Troubleshooting tips:

[Get general help for exceptions.](#)

We have to convert the string value returned by the input function to a number

```
deposit=input("How much would you like to deposit? ")
if int(deposit) > 100 :
    print("You get a free toaster!")
print("Have a nice day")
```

Here is another way to do the same thing

```
deposit=int(input("How much would you like to deposit? "))
if deposit > 100 :
    print("You get a free toaster!")
print("Have a nice day")
```

Branching

What if you get a free toaster for over \$100 and a free mug for under \$100

```
deposit=input("How much would you like to deposit? ")
if float(deposit) > 100 :
    print("You get a free toaster!")
else:
    print("Enjoy your mug!")
print("Have a nice day")
```

The code in the *else* statement is only executed if the condition is NOT true

What will appear on the screen if we enter 50? 150? 100?

DEMO

Adding an else clause

You can use boolean variables to remember if a condition is true or false

```
deposit= input("how much would you like to deposit? ")
if float(deposit) > 100 :
    #Set the boolean variable freeToaster to True
    freeToaster=True

#if the variable freeToaster is True
#the print statement will execute
if freeToaster :
    print("enjoy your toaster")
```

Make sure you test what happens when your if statement is true and what happens when your if statement is false.

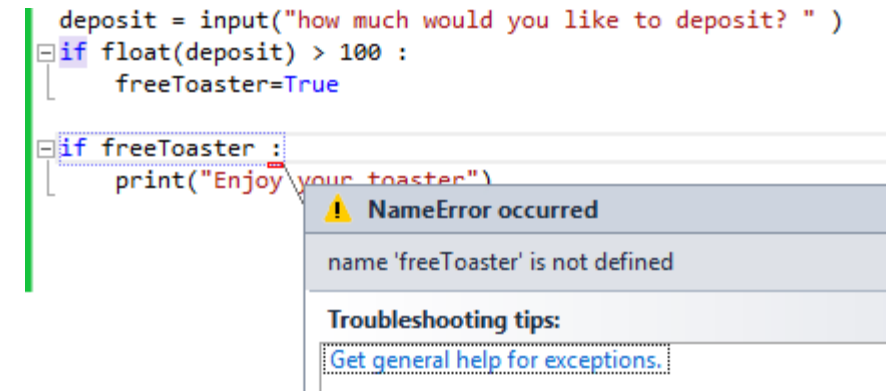
DEMO

Using a Boolean variable and testing all paths

Why does our code crash when we enter a value of 50 for a deposit?

```
deposit= input("how much would you like to deposit? ")  
if float(deposit) > 100 :  
    #Set the boolean variable freeToaster to True  
    freeToaster=True
```

```
#if the variable freeToaster is True  
#the print statement will execute  
if freeToaster :  
    print("enjoy your toaster")
```



The screenshot shows a code editor with the following Python code:

```
deposit = input("how much would you like to deposit? ")  
if float(deposit) > 100 :  
    freeToaster=True  
if freeToaster :  
    print("Enjoy your toaster")
```

An error message box is displayed on the right side of the editor, indicating a **NameError occurred**. The message states: **name 'freeToaster' is not defined**. Below the error message, there are **Troubleshooting tips:** and a link to [Get general help for exceptions.](#)

Look at the error message: Name 'freeToaster' is not defined.

It's always a good idea to initialize your variables!

```
#Initialize the variable to fix the error  
freeToaster=False
```

```
deposit= input("how much would you like to deposit? ")  
if float(deposit) > 100 :  
    #Set the boolean variable freeToaster to True  
    freeToaster=True  
  
#if the variable freeToaster is True  
#the print statement will execute  
if freeToaster :  
    print("enjoy your toaster")
```

Aren't you just making the code more complicated by using the Boolean variable?

- That depends...
- What if you are writing a program, and there is more than one place you have to check that condition? You could check the condition once and remember the result in the Boolean variable
- What if the condition is very complicated to figure out? It might be easier to read your code if you just use a Boolean variable (often called a flag) in your if statement

And now we have more ways to make typing mistakes! Can you find three?

```
deposit=input("How much would you like to deposit? ")
if float(deposit) > 100:
    print("You get a free toaster!")
    freeToaster=true
else:
    print("Enjoy your mug!")
print("Have a nice day")
```

```
deposit=input("How much would you like to deposit? ")
if float(deposit) > 100:
    print("You get a free toaster!")
    freeToaster=True
else:
    print("Enjoy your mug!")
print("Have a nice day")
```

Your challenge

- Calculate shipping charges for a shopper
- Ask the user to enter the amount for their total purchase
- If their total is under \$50 add \$10, otherwise shipping is free
- Tell the user their final total including shipping costs and format the number so it looks like a monetary value
- Don't forget to test your solution with
 - a value > 50
 - a value < 50
 - a value of exactly 50

Congratulations!

- Your code can now react to different conditions!
- You can now solve problems that require decision making





Microsoft

©2013 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.