INTRODUCTION TO PROGRAMMING IN PYTHON

Variable names, Assignment, Casting & Input.

REVIEW – DATA TYPES

3.14

"pelican"

True

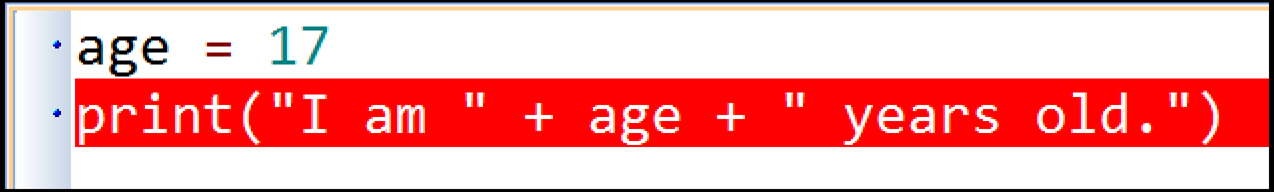
Last lesson you learned about 4 data types. What were they?

42

(01525)217567

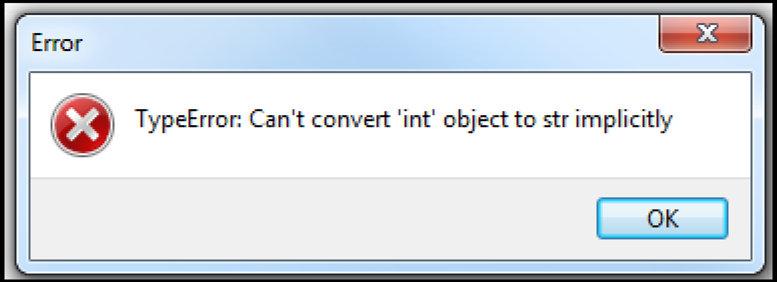
CONCATENATION

- Last lesson we looked at concatenation i.e. joining two strings together.
- Look at the code below:



```
age = 17
print("I am " + age + " years old.")
```

- Trying to run this code has generated an error because age is not a string:



TypeError: Can't convert 'int' object to str implicitly

CASTING

- Casting is when we convert a variable from one data type to another.
- To cast a variable as a string, integer or float respectively you use the following code:
  - str (var_name)
  - int (var_name)
  - float (var_name)
- If you have data stored as a string and want to do some maths with it you must first convert it to an integer or a float.
- If you have data that is an integer or float then you need to cast it as string if you want to concatenate it with other string data.
- There is no reason to cast a variable if it is already the correct data type.

CASTING

- Careful consideration of how a variable will be used can help you choose the right data type for a variable and minimise the need for casting.
- Some important things to remember:
  - A variables data type is that of the first value it holds e.g. name = “Fred” would make name a string data type because that is the type of the first value stored in the variable name.
  - A variables data type only changes permanently if the newly cast value is assigned back to the same name e.g. age = int(age)

CASTING - EXAMPLES

- Consider the program outlined below:
  - price = “1.50”
  - total_price = float(price) * 3
  - total_price = str(total_price)
  - print(“The price for one is “ + price + “ and the total price is “ + total_price)

- What data type is price on line 1?
- What is the data type of total_price after executing line 2?
- What is the data type of total_price on the final line of the program?
- Complete the casting quiz on Moodle and then open Casting.py and follow the instructions.

GETTING INPUT: VARIABLE NAMES

- There are a number of conventions to be considered when selecting variable names, also known as variable identifiers:
  - The names should be meaningful i.e. you should be able to tell what data is stored in the variable from its name.
  - The names should start with a letter or underscore and after that can also contain numbers but should not contain spaces or special characters.
  - Names cannot be reserved words that have a meaning in the language e.g. print. In PyScripter these words will change to a different colour to show they have another use.
  - Two common methods for variable names with multiple words are underscores (multi_word_var) and what is known as camel caps (multiWordVar) where the first letter of each word after the first is capitalised.
- Complete the variable names quiz on Moodle.

VARIABLE ASSIGNMENT

- Assigning a variable means giving it a new value.
- In Python the assignment operator is the equals sign.
- Assignment must always take place with the variable on the left i.e. name = “Bob” is fine but “Bob” = name does not work.
- To assign a string value the data must be in speech marks as above.
- To assign an integer or float value speech marks are not used. E.g. age = 17 or height = 1.86.
- If a number, whole or with fraction, were assigned in speech marks it would be treated as a string.

INPUT

- Getting input from a user or other source into our program is an important step in creating useful programs.
- The input command is used in conjunction with assignment to assign a variable the value input by the user.
- The syntax is variable_name = input(“Question to prompt user”)
- Inside the speech marks you can put a question that will appear to the user prompting them what to enter and you choose where to store that e.g.
  - forename = input(“What is your first name?”)
- The program will wait for the user to enter some data and when they press return it will store their response and move on with the program.

INPUT

- It is important to remember that all data received through input will be string data until you cast it. This can be done on the same line as gathering the input to save space.

  - Age = int(input(“How many years old are you?”)) is the same as
  - Age = input(“How many years old are you?”)
  - Age = int(Age)

  - Height_in_m = float(input(“What is your height in metres?”)) is the same as
  - Height_in_m = input(“What is your height in metres?”)
  - Height_in_m = float(Height_in_m)

  - Complete the Receiving input quiz on Moodle and open and complete Input.py

INPUT

- You can even use existing variables in your questions that you prompt the user with e.g.
  - name = input(“What is your name?”)
  - fav_colour = input(“What’s your favourite colour “ + name + “?”)
- The code above would include the user’s name they entered in the subsequent question about their favourite colour.

ANOTHER WAY TO JOIN STRINGS

- We have already learned how to concatenate strings.
- In a print command it is also possible to join strings using a comma.
- Using a comma has a few differences to using a + to join strings:
    1. It automatically inserts a space between strings where a comma is placed
    2. It will print a string version of the variable without the need to cast it first
    3. You can't use the comma method inside the brackets of an input command

```
name = “Bob”
age = 42
print(name + “is “ + str(age) + “ years old.”)
```

```
name = “Bob”
age = 42
print(name, “is”, age, “years old.”)
```

Both of these print statements would print the same thing:
Bob is 42 years old.