Python Lesson 5

Today we'll be creating a times table tester game, and learning a few things along the way.

Data type conversion

Question 1: What will this Python code display? What do you need to change to make the program print 10?

```
five="5"
ten=five+five
print(ten)
```

Question 2: This times table tester function has a bug[1]! It always tells you that you have typed the wrong answer. What is the problem? Can you fix it?

```
def ask_question(num1, num2):
    print("What is ", num1, "x", num2, "?", sep="")
    answer=input()
    if answer==num1*num2:
        print("Correct!")
        return True
    else:
        print("Wrong, the answer is", num1*num2)
        return False

ask_question(6, 9)
```

Both questions are to do with incorrect data types. You need to convert between strings and integers.

To convert a values to different data types you can use the following functions:

str(value) – converts value (e.g. a number) to a string

int(value) – converts value (e.g. a string) to an integer, or raises a **ValueError** if the value cannot be converted. For example, int("5") returns the number 5 but int("X") raises a ValueError. This error is called called an **exception**, and indicates that something unexpected has happened in your program.

float(value) – converts a value to a floating point number, or raises a ValueError.

1 When using Python 3!

Exceptions and Handling Exceptions

As we've seen, if you use the int() function to convert something to an integer that doesn't look like a number, Python will raise a ValueError.

Python has a try/except feature which lets you catch exceptions and do something useful instead of crashing your program! Here is an example of catching an exception and printing a friendly error message.

If the lines of code in the "try:" block raise a ValueError, then the lines of code under the "except ValueError:" will be run immediately. This is called "catching an exception".

```
def input_number():
    while True:
        try:
            number=int(input("Enter a number: "))
            break
        except ValueError:
            print("Oops! That wasn't a valid number. Try again.")
    return number

# Call input_number
input_number()
```

Dates and times

Many computers internally store the current time as a number of seconds or milliseconds since midnight on January 1st 1970 (Greenwich Mean Time!). This is called the "epoch" – the beginning of time for computers. You can see this value with Python, using the **time** module.

```
import time

print(time.time())
1467206513.7132623

# wait a few seconds

print(time.time())
1467206569.2550714
```

You can use this to time how long something has taken. Here's an example program that calculates the 35th number in the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13, ...) and prints how long it took:

```
import time

def fibonacci(number):
    if number==0:
        return 0
    elif number==1:
        return 1
    else:
        return fibonacci(number-1) + fibonacci(number-2)

start=time.time()
print("The 35th number in the Fibonacci sequence is", fibonacci(35))
end=time.time()

print("This calculation took", end-start, "seconds")
```

The variable **start** is set to the current time before calling the function, and the variable **end** is set to the current time after calling the function. We can subtract start from end to see how long the program took to run.

You can use time.time() like this in a game, to tell the player how quickly they won!

Task: You can add this to the "Escape from Stamford Green" adventure game from last week to tell a player, when they have escaped from the school, how many seconds they took to complete the game.

More Reading:

The **time** module has lots of other functionality for printing the current time of day in English formats, etc. You can look at the Python documentation! Use "Help → Python Docs" from the IDLE menu, then search for "time".

Something else that's a little bit fun about this Fibonacci program is that it shows an example of **recursion,** which means that a function is calling itself!

When running fibonnaci(35), the function will calculate the answer by adding together the previous two numbers in the sequence by calling fibonacci(34)+fibonacci(33). These will in turn repeat, until eventually we reach fibonacci(0) and fibonacci(1).

Times Table Challenge!

We're going to write a program putting together what we've seen. The program will ask you questions for a times table, and give you a score based on speed and accuracy!

```
import random
import time

def input_number(prompt):
    while True:
        try:
            print(prompt)
            number=int(input("> "))
            break
        except ValueError:
            print("Oops! That wasn't a valid number. Try again.")
    return number

def ask_question(num1, num2):
    answer=input_number("What is " + str(num1) + " x " + str(num2) + "?")
    if answer==num1*num2:
        print("Correct!")
        return True
    else:
        print("Wrong, the answer is", num1*num2)
        return False

def do_times_table_test(table):
    questions=list(range(1, 13))
    random.shuffle(questions)

    correct=0
    start_time=time.time()

    for question in questions:
        print()
        if ask_question(table, question):
            correct=correct+1

    end_time=time.time()
    duration=end_time-start_time

    # Bonus score if you took less than 60 seconds
    if duration<60:
        score_multiplier=1+3*(60-duration)/60
    else:
        score_multiplier=1
    score=correct*score_multiplier

    print()
    print("You got", correct, "sums correct in", int(duration), "seconds.")
    print("This means you scored", int(score), "points!")
    print()

while True:
    table=input_number("Which times table do you want to try?")
    do_times_table_test(table)
```