Week 3 Video Programs

**What is a while loop?**

**x = 0**

**while x < 5:**

**print("Not there yet, x=" + str(x))**

**x = x + 1**

**print("x=" + str(x))**

How many times will "Not there yet" be printed?

Ans: 5

**Correct Response**

You got it! The variable x starts at 0 and gets incremented once per iteration, so there are 5 iterations for which x is smaller than 5.

#### More while Loop Examples

Can you work out what this function does? Try passing different parameters to the attempts function to see what it does.



**def attempts(n):**

**x = 1**

**while x <= n:**

**print("Attempt " + str(x))**

**x += 1**

**print("Done")**

**attempts(5)**

Output:

**Attempt 1**

**Attempt 2**

**Attempt 3**

**Attempt 4**

**Attempt 5**

**Done**

#### Why Initializing Variables Matters

In this code, there's an initialization problem that's causing our function to behave incorrectly. Can you find the problem and fix it?



**def count\_down(start\_number):**

**current=3**

**while (current > 0):**

**print(current)**

**current -= 1**

**print("Zero!")**

**count\_down(3)**

**Here is your output:**

**3**

**2**

**1**

**Zero!**

**You nailed it! By initializing the current variable you got**

**the function to behave correctly.**

#### Infinite Loops and How to Break Them

The following code causes an infinite loop. Can you figure out what’s missing and how to fix it?



**def print\_range(start, end):**

**n = start**

**while n <= end:**

**print(n)**

**n=n+1**

**print\_range(1, 5)**

**Here is your output:**

**1**

**2**

**3**

**4**

**5**

**Great work! You've managed to fix the error in the code that**

**was causing an infinite loop!**

#### What is a for loop?

Want to try this out? Let's give it a go!

Fill in the gaps of the sum\_squares function, so that it returns the sum of all the squares of numbers between 0 and x (not included). Remember that you can use the range(x) function to generate a sequence of numbers from 0 to x (not included).



**def square(n):**

**return n\*n**

**def sum\_squares(x):**

**sum = 0**

**for n in range(x):**

**sum += n\*n**

**return sum**

**print(sum\_squares(10)) # Should be 285**

**RunReset**

**Here is your output:**

**285**

**Nice job! You've got Python to do some complex operations**

**for you!**

#### More for Loop Examples

In math, the factorial of a number is defined as the product of an integer and all the integers below it. For example, the factorial of four (4!) is equal to 1\*2\*3\*4=24. Fill in the blanks to make the factorial function return the right number.



**def factorial(n):**

**result = 1**

**for i in range(1,n+1):**

**result=result \* i**

**return result**

**print(factorial(4)) # should return 24**

**print(factorial(5)) # should return 120**

**Here is your output:**

**24**

**120**

**Well done, you! The pieces of code you're tackling keep**

**getting more complex, but you're doing a great job!**

#### Nested for Loops

Given the following code:



**teams = [ 'Dragons', 'Wolves', 'Pandas', 'Unicorns']**

**for home\_team in teams:**

**for away\_team in teams:**

What should the next line be to avoid both variables being printed with the same value?



while home\_team != away\_team:



for home\_team == away\_team:



away\_team = home\_team



if home\_team != away\_team:

**Correct**

**You got it! We want to print all possible team pairings but exclude those where a team would play against itself. To do that, we need a conditional that skips the case where both teams are the same.**

#### Common Errors in for Loops

The validate\_users function is used by the system to check if a list of users is valid or invalid. A valid user is one that is at least 3 characters long. For example, ['taylor', 'luisa', 'jamaal'] are all valid users. When calling it like in this example, something is not right. Can you figure out what to fix?

****

**def validate\_users(users):**

**for user in users:**

**if is\_valid(user):**

**print(user + " is valid")**

**else:**

**print(user + " is invalid")**

**validate\_users(["purplecat"])**

**RunReset**

**Here is your output:**

**purplecat is valid**

**Nice job! You figured out that you needed to call the**

**function with a list instead of just a string!**

#### What is recursion? (Optional)

The function sum\_positive\_numbers should return the sum of all positive numbers between the number n received and 1. For example, when n is 3 it should return 1+2+3=6, and when n is 5 it should return 1+2+3+4+5=15. Fill in the gaps to make this work:



**def sum\_positive\_numbers(n):**

**# The base case is n being smaller than 1**

**if n <= 1:**

**return n**

**else:**

**return n + sum\_positive\_numbers(n-1)**

**# The recursive case is adding this number to**

**# the sum of the numbers smaller than this one.**

**print(sum\_positive\_numbers(3)) # Should be 6**

**print(sum\_positive\_numbers(5)) # Should be 15**

**Here is your output:**

**6**

**15**

**Whoohoo! You've just** written your first recursive function.

**Well done!**

#### Recursion in Action in the IT Context

Which of the following scenarios would benefit the most from using a recursive function to solve the problem?



You need to print out a list of the employees in your company.



You need to know how many files are present in a single directory of your computer.



**You need to create a family tree, showing several generations of your ancestors, with all of their children.**

**Correct**

**Great job! You're getting the concept of recursion and when it's a better solution than the traditional looping techniques.**

**You need to create a family tree, showing several generations of your ancestors, with all of their children.**

**is selected.This is correct.**

**Great job! You're getting the concept of recursion and when it's a better solution than the traditional looping techniques.**



You need to calculate the square root of numbers 1 through 10.