**Video Programs**

**Data Types**

Why does this code raise an error:



**print("1234"+5678)**



**Because Python doesn't know how to add a number to a string.**



Because in Python it's only possible to add numbers, not strings.



Because in Python it's not possible to print integers.



Because numbers shouldn't be written between quotes.

#### Variables

Now, it's your turn to give it a try!

Fill in the blanks to calculate the area of a triangle of base 5, height 3 and output the result. **Reminder:** the area of a triangle is (base\*height)/2.



**base = 5**

**height = 3**

**area = (base\*height)/2**

**print(area)**

#### Expressions, Numbers, and Type Conversions

Practice writing some expressions and conversions yourself.

In this scenario, we have a directory with 5 files in it. Each file has a different size: 2048, 4357, 97658, 125, and 8. Fill in the blanks to calculate the average file size by having Python add all the values for you, and then set the files variable to the number of files. Finally, output a message saying "The average size is: " followed by the resulting number. Remember to use the str() function to convert the number into a string.



**total = 2048 + 4357 + 97658 + 125 + 8**

**files = 5**

**average = total / files**

**print("The average size is:" +str(average))**

#### Defining Functions

Do you think you can flesh out your own function? I think you can! Let’s give it a go.

Flesh out the body of the print\_seconds function so that it prints the total amount of seconds given the hours, minutes, and seconds function parameters. Remember that there are 3600 seconds in an hour and 60 seconds in a minute.

****

**def print\_seconds(hours, minutes, seconds):**

**print(hours\*3600+minutes\*60+seconds)**

**print\_seconds(1,2,3)**

#### Returning Values

Use the get\_seconds function to work out the amount of seconds in 2 hours and 30 minutes, then add this number to the amount of seconds in 45 minutes and 15 seconds. Then print the result.



**def get\_seconds(hours, minutes, seconds):**

**return 3600\*hours + 60\*minutes + seconds**

**amount\_a = get\_seconds(2,30,0)**

**amount\_b = get\_seconds(0,45,15)**

**result = amount\_a+amount\_b**

**print(result)**

#### The Principles of Code Reuse

Ready to try it yourself? See if you can reduce the code duplication in this script.

In this code, identify the repeated pattern and replace it with a function called month\_days, that receives the name of the *month*and the number of *days* in that month as parameters. Adapt the rest of the code so that the result is the same. Confirm your results by making a function call with the correct parameters for both months listed.

****

**def month\_days(month,days) :**

**print(str(month)+ "has" + str(days) + " days.")**

**month\_days('june',30)**

**month\_days('july',31)**

#### Code Style

This function to calculate the area of a rectangle is not very readable. Can you refactor it, and then call the function to calculate the area with base of 5 and height of 6? **Tip:** a function that calculates the area of a rectangle should probably be called rectangle\_area, and if it's receiving base and height, that's what the parameters should be called.



**def rectangle\_area(base, height):**

**area = base \* height # the area is base\*height**

**print("The area is " + str(area))**

**rectangle\_area(5,6)**

#### Comparing Things

Figure out what's the relationship between the strings "cat" and "Cat" by replacing the plus sign with comparison operators.



**print("cat" < "Cat")**



"cat" equals "Cat"



"cat" is smaller than "Cat"



**"cat" is larger than "Cat"**

#### Branching with if Statements

The is\_positive function should return True if the number received is positive, otherwise it returns None. Can you fill in the gaps to make that happen?



**def is\_positive(number):**

**if number>0:**

**return True**

**else:**

**return None**

#### else Statements

The is\_positive function should return True if the number received is positive and False if it isn't. Can you fill in the gaps to make that happen?



def is\_positive(number):

if number > 0:

return True

else: return False

#### elif Statements

The number\_group function should return "Positive" if the number received is positive, "Negative" if it's negative, and "Zero" if it's 0. Can you fill in the gaps to make that happen?



**def number\_group(number):**

**if number>0:**

**return "Positive"**

**elif number<0:**

**return "Negative"**

**else:**

**return "Zero"**

**print(number\_group(10)) #Should be Positive**

**print(number\_group(0)) #Should be Zero**

**print(number\_group(-5)) #Should be Negative**

#### 