

Adding structure to your program: functions



Example

- Program to read in the area of a circle, calculate and print out the radius:

```
import math

# Calculate radius of circle, given area
area = float(input("Area of the circle: "))
rsquared = area/math.pi
radius = math.sqrt(rsquared)
print("The area of the circle is : " + str(area))
print("The corresponding radius is : " + str(radius))
```

- This works, but is really using Python for “scripting”
- We want to add some structure

Make a function and run it

```
import math

def main():
    # Calculate radius of circle, given area
    area = float(input("Area of the circle: "))
    rsquared = area/math.pi
    radius = math.sqrt(rsquared)
    print("The area of the circle is : " + str(area))
    print("The corresponding radius is : " + str(radius))

main()
```

- This does exactly the same thing, BUT...
 - `def main():` is the start of a function called `main`
 - The contents of `main` are all indented by 4 spaces (or 1 tab)
 - The `main()` at the end *executes* (runs) the function `main()`
- This adds “structure”, we create a function, then run it, but no real advantage yet. . .

Temperature Converter

```
def fahrenheit(tc):  
    tf = 1.8*tc + 32  
    return tf  
  
def centigrade(tf):  
    tc = (tf - 32.0)/1.8  
    return tc  
  
# Main function  
def main():  
    boiling = 100.0 # boiling point of water (C)  
    dv = 134.0 # max temp in Death Valley (F)  
  
    boilingf = fahrenheit(boiling)  
    dvc = centigrade(dv)  
  
    print("Water boils at " + str(boilingf) + " F")  
    print("Death Valley is at " + str(dvc) + " C")
```

main()

Temperature Converter

- What this does:
 - `def fahrenheit(tc)`: defines a function that takes a variable `tc` in centigrade and returns its value in fahrenheit
 - `def centigrade(tf)`: defines a function that takes a variable `tf` in fahrenheit and returns its value in centigrade
 - then in `main()` you can simply use these functions (as you would use e.g. `math.cos()`)
 - Note that `main()` is simply a function with no arguments
 - The final code `main()` runs `main()`, which calls the two functions `fahrenheit()` and `centigrade()` as required
- This allows you to break up your program, and makes it much more readable

Radius from area revisited: refining the first program

```
import math

def getRadius(area):
    rsquared = area/math.pi
    radius = math.sqrt(rsquared)
    return radius

def main():
    area = float( input("Area of the circle: "))
    radius = getRadius(area)
    print( "The area of the circle is : " + str(area))
    print( "The corresponding radius is : " + str(radius))

main()
```

Temperature Converter

- Points to note:
 - `def getRadius(area):` is a function that takes area and returns radius, so can be called from in `main()` (or from any other function)
 - Note there is a variable called radius in both `getRadius()` and in `main()`, but these are *different* variables
 - Variables only exist inside the functions where they are defined
- Provided you use this type of structure you do not need to worry about variable name clashes in different functions
- You *can* define variables outwith functions (these are called global variables) but, at this stage of programming, don't do it!