Week03: Functions, Part II (Ch. 5)

CSC110: INTRO TO COMPUTER PROGRAMMING WITH PYTHON

INSTRUCTOR: BILL BARRY

1

This Week: More on Functions

Parameters and Returns, Turtle Practice

Creating and Using Your Own Module

Hackathon Intro

Hackathon

2

Roadmap

☞ More Functions

Selection

Iteration

Midterm Exam

Strings

File I/O, Exceptions

Lists

List Algorithms

Dictionaries & Sets

Final Exam

3

Parameters and Returns, Applied in Turtle

4

Pause & Play

LET’S REVIEW PARAMETERS AND USE THEM IN TURTLE WORLD

5

Turtle Window Program with Parameters

Start with the code from last week’s Turtle program. Change it so that users can specify the color, pen size, and pane size. Add a main()

```
def drawSquare():
    forward(100)
    right(90)
    forward(100)
    right(90)
    forward(100)
    right(90)
    forward(100)
    right(90)
```

```
def drawWindow():
    drawSquare()
    left(90)
    drawSquare()
    left(90)
    drawSquare()
    left(90)
    drawSquare()
    left(90)

def main():
    from turtle import *
    color('blue')
    pensize(5)
    drawWindow()
    done()

main()
```

6

Turtle Program w/Params: Code

```
def drawSquare(size):
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)

def drawWindow(squareSize):
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
```

```
def main():
    windowColor = input("What color window? ")
    penSize = int(input("What pen size would you like? "))
    paneSize = int(input("What size panes would you like? "))
    color(windowColor)
    pensize(penSize)
    drawWindow(paneSize)
    done()

from turtle import *
main()
```

7

Parameters and Return Values

Functions can have "inputs" (*parameters*) and can have "outputs" (*returns*)

These take the form:

```
def fnName(param1, param2):
    # calculations
    return returnvalue
```

Example:

```
def dogYears(humanYears):
    dYears = humanYears / 7
    return dYears
```

When there is a return value, you can use the function wherever that value is needed:

```
myAge = 29
myAgeInDogYears = dogYears(myAge)
```

– or –

```
print(dogYears(myAge))
```

Note that the name of the parameter (receiving end) and the name of the argument (sending end) don't need to match; the function is isolated and stands alone. If there are multiple arguments, the order of the arguments does need to match the order of the parameters, however

8

Pause & Play

LET'S REVISIT OUR TURTLE PROGRAM AND ADD SOME RETURN VALUES TO OUR CODE

9

"Return" to Turtle Window Program

Start with our recent Turtle code. Move each user input into its own function. Make main call that function and use its returns. Use no globals!

```
def drawSquare(size):
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)

def drawWindow(squareSize):
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
```

```
def main():
    windowColor = input("What color would you like the window to be? ")
    penSize = int(input("What pen size would you like? "))
    paneSize = int(input("What size panes would you like? "))
    color(windowColor)
    pensize(penSize)
    drawWindow(paneSize)
    done()

from turtle import *
main()
```

10

“Return” Turtle Window Program Code

```
def drawSquare(size):
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)

def drawWindow(squareSize):
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)

def getColorChoice():
    windowColor = \
        input("What color? ")
    return windowColor

def getPenSizeChoice():
    penSize = int(input("What pen size would you like? "))
    return penSize

def getPaneSizeChoice():
    paneSize = int(input("What size panes would you like? "))
    return paneSize

def main():
    whatColor = getColorChoice()
    whatPenSize = getPenSizeChoice()
    whatPaneSize = getPaneSizeChoice()

    color(whatColor)
    pensize(whatPenSize)
    drawWindow(whatPaneSize)
    done()

from turtle import *
main()
```

11

Python and Multiple Return Values

In Python, it’s easy to return more than one value from a function

Within the function, just **list one or more values to return**, separated by commas

On the calling side, just **list one or more variables before the equal sign**, separate by commas

Python packs up the returned values into a bundle (a *tuple*), then unpacks them and stores them into the specified variables

Note: don’t expect this to work in other languages; it doesn’t! It’s much more work elsewhere, e.g., you must do the packaging and unpackaging yourself, which isn’t trivial

12

“Return” Turtle Window Program Code

```
def drawSquare(size):
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)
    forward(size)
    right(90)

def drawWindow(squareSize):
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
    drawSquare(squareSize)
    left(90)
```

```
def getOptions():
    windowColor = input("What color would you like the window to be? ")
    penSize = int(input("What pen size would you like? "))
    paneSize = int(input("What size panes would you like? "))
    return windowColor, penSize, paneSize

def main():
    whatColor, whatPenSize, whatPaneSize = getOptions()
    color(whatColor)
    pensize(whatPenSize)
    drawWindow(whatPaneSize)
    done()

from turtle import *
main()
```

13

Creating and Using Your Own Module

14

Using Built-in Modules

Modules are files containing multiple related functions

Python provides several useful ones, including the random and math modules

To enable the use of the module’s functions, import it in your program:

```
import math
```

You can then use its functions using the module name, a dot, and the function name:

```
print math.sqrt(4)
```

Here’s a page with a brief listing of what math contains:
https://www.programiz.com/python-programming/modules/math

15

Ways to Use a Module

1. Import the module, use *module.function* notation:
```
import math
print math.sqrt(81)
```
2. Import individual functions/constants you need:
```
from math import sqrt
from math import pi
print(sqrt(81) * pi)
```
3. Import all functions:
```
from math import *
print(sqrt(81) * pi)
# be aware that conflicts can arise, if you name functions with
# the same name as any imported ones
```

16

Pause & Play

LET’S USE CREATE AND USE A MODULE

17

Creating Your Own Module

You can create your own module. For example, put several related functions in a separate .py file, e.g., **`payroll.py`**

You can then import the module the same way as built-in ones:

```
import payroll
```

You can access the functions with the *module-dot-function* notation you saw with the math module:

```
netPay = grossPay – payroll.withholding(grossPay)
```

18

Create and Use a Module

Write a module called **taxFunctions** (in a file named taxFunctions.py)

In it, define a “constant” for tax rate; set it to 9.5% (.095)

Write a function called **calcTax**
- It should accept one parameter representing the amount on which to calculate tax
- It should return the tax calculated, rounded to the nearest penny

Create a file called **order.py**
- In it, write a main program that asks for an order subtotal
- Use the calcTax function from the module to calculate the tax
- Display, in an attractive form, the subtotal, tax, and total

19

Code Solution #1

taxFunctions.py

```
TAX_RATE = 0.095

def calcTax(amount):
    tax = round(amount * TAX_RATE, 2)
    return tax
```

order.py

```
import taxFunctions

def main():
    orderSubtotal = float(input('Enter order subtotal: '))
    orderTax = taxFunctions.calcTax(orderSubtotal)
    orderTotal = orderSubtotal + orderTax
    print('Subtotal    $', format(orderSubtotal, '8,.2f'))
    print('Tax         $', format(orderTax,      '8,.2f'))
    print('----------   ', format('-' * 7,       '>8s'))
    print('Subtotal    $', format(orderTotal,    '8,.2f'))

main()
```

20

Code Solution #2

taxFunctions.py

```
TAX_RATE = 0.095

def calcTax(amount):
    tax = round(amount * TAX_RATE, 2)
    return tax
```

order.py

```
from taxFunctions import calcTax

def main():
    orderSubtotal = float(input('Enter order subtotal: '))
    orderTax = calcTax(orderSubtotal)
    orderTotal = orderSubtotal + orderTax
    print('Subtotal    $', format(orderSubtotal, '8,.2f'))
    print('Tax         $', format(orderTax,      '8,.2f'))
    print('----------   ', format('-' * 7,       '>8s'))
    print('Subtotal    $', format(orderTotal,    '8,.2f'))

main()
```

21

Code Solution #3

taxFunctions.py

```
TAX_RATE = 0.095

def calcTax(amount):
    tax = round(amount * TAX_RATE, 2)
    return tax
```

order.py

```
from taxFunctions import *

def main():
    orderSubtotal = float(input('Enter order subtotal: '))
    orderTax = calcTax(orderSubtotal)
    orderTotal = orderSubtotal + orderTax
    print('Subtotal    $', format(orderSubtotal, '8,.2f'))
    print('Tax         $', format(orderTax,      '8,.2f'))
    print('----------   ', format('-' * 7,       '>8s'))
    print('Subtotal    $', format(orderTotal,    '8,.2f'))

main()
```

22

Mini Hackathon Project Intro

23

Mini Hackathon Overview

Today
- We'll work in groups of about four students
- Groups should plan together
- *No one should code any of this work before tomorrow*; we want this to be coded *only* during the class period tomorrow

Tomorrow
- Groups work to get this coded, tested, and tidied up
- Success is measured by the ability to complete work during the class period, with full understanding
- No grades will be assigned, nor work submitted; this is just a fun way to practice

Notes
- No one should control or overpower; every student's input must be heard and considered. Think "we all rise together"
- There's no need for each student to type on their own

24

Payroll Project Overview

The code lets the user enter payroll information including employee name, hours worked, hourly wage, and the number of deductions claimed

It does calculations including gross pay (no overtime), social security tax (6.2% of gross pay), Medicare tax (1.45% of gross pay), and withholding tax (subtract $80.8 for each deduction claimed, multiply result by 20%), and net pay. Where fractional penny amounts might be generated, rounds to the nearest penny in each individual calculation

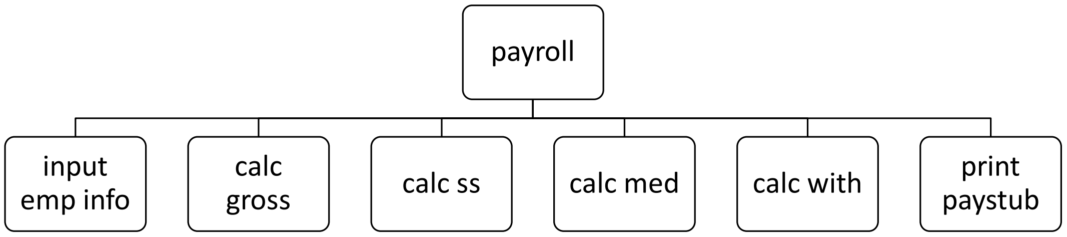
Finally, it prints a payroll check stub as shown

```
Employee name: Joe Schmoe
Hours worked:  45.25
Hourly wage:   53.67
# deductions:  2

Check for Joe Schmoe
-------------------------
Gross pay      $ 2,428.57
SS tax         $   150.57
Medicare tax   $    35.21
Withholding    $   453.39
-------------------------
Net pay        $ 1,789.40
```

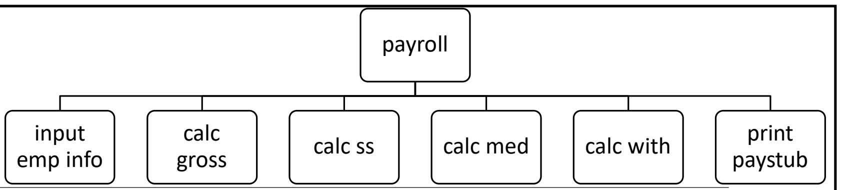
25

Call Hierarchy



26

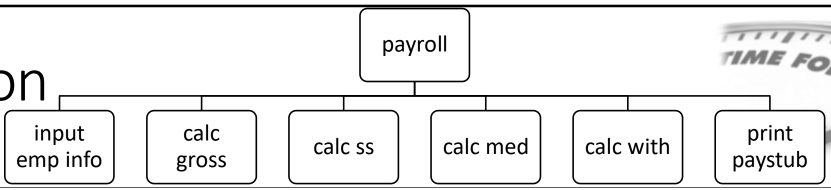
Function Data

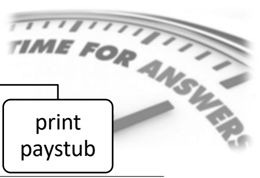


Function	Inputs (parameters)	Outputs (returns)
payroll		
input emp info		
calc gross		
calc ss		
calc med		
calc with		
print paystub		

27

Function Data





Function	Parameter(s)	Return(s)
payroll	(none)	(none)
input emp info	(none)	name, hours, wage, deductions
calc gross	hours, wage	gross pay
calc ss	gross pay	ss tax
calc med	gross pay	med tax
calc with	gross pay	with tax
print paystub	name, gross pay, ss tax, med tax, with tax, net pay	(none)

28

Mini Hackathon

READY, SET, GO!

29

Payroll Challenge: Bill's Solution #1

```
SS_TAX_RATE = 0.062
MED_TAX_RATE = 0.0145
WITH_TAX_RATE = 0.2
WITH_ALLOWANCE_PER_DEDUCT = 80.80
CHECK_MONEY_FMT = '8,.2f'

def main():
    payroll()

def payroll():
    name, hours, wage, deductions = inputEmpInfo()
    grossPay = calcGross(hours, wage)
    ssTax = calcSS(grossPay)
    medTax = calcMed(grossPay)
    withTax = calcWith(grossPay, deductions)
    netPay = grossPay - ssTax - medTax - withTax
    printCheck(name, grossPay, ssTax, medTax, withTax, netPay)

def inputEmpInfo():
    empName =           input('Employee name: ')
    empHours =    float(input('Hours worked:  '))
    empWage =     float(input('Hourly wage:   '))
    empDeductions = int(input('# deductions:  '))
    return empName, empHours, empWage, empDeductions
```

30

Payroll Challenge: Bill's Solution #2

```
def calcGross(hours, wage):
    return round(hours * wage, 2)

def calcSS(grossPay):
    return round(grossPay * SS_TAX_RATE, 2)

def calcMed(grossPay):
    return round(grossPay * MED_TAX_RATE, 2)

def calcWith(grossPay, deductions):
    taxable = grossPay - deductions * WITH_ALLOWANCE_PER_DEDUCT
    return round(taxable * WITH_TAX_RATE, 2)

def printPaystub(name, gross, ss, med, withholding, net):
    print()
    print('Check for ', name)
    print('-' * 25)
    print('Gross pay      $', format(gross,       CHECK_MONEY_FMT))
    print('SS tax         $', format(ss,          CHECK_MONEY_FMT))
    print('Medicare tax   $', format(med,         CHECK_MONEY_FMT))
    print('Withholding    $', format(withholding, CHECK_MONEY_FMT))
    print('-' * 25)
    print('Net pay        $', format(net,         CHECK_MONEY_FMT))

main()
```

31

The End

32