Assignment 4 – User Defined Functions

Functions

A function is a section of code that performs a specific task.

Syntax

def function_name([*arguments*]):
    statements

Defining (no arguments)

1. Create a python file named **future_value_2.py**.
2. Code the following

```
#! /usr/bin/env python3


def print_welcome():
    print("Welcome to the Future Value Calculator")
    print()

```

Calling (no arguments)

3. Code the following

```
#! /usr/bin/env python3


def print_welcome():
    print("Welcome to the Future Value Calculator")
    print()


print_welcome()

```

```
Run: test  test  blackjack
C:\Users\Saddleback\AppData\Local\Programs\Python\Pytho
Welcome to the Future Value Calculator


Process finished with exit code 0
```

Defining and calling (with arguments)

4. Add the argument to the call and method

Screen Capture #1 (1 point)

```
#! /usr/bin/env python3


def print_welcome(message):
    print(message)
    print()


message = "Welcome to the Future Value Calculator"
print_welcome(message)
```

```
Run: test  test  blackjack
C:\Users\Saddleback\AppData\Local\Programs\Python\Pyth
Welcome to the Future Value Calculator


Process finished with exit code 0
```

main() Function

When using functions, it's best practice is to use a main() function to control the flow and the main() function will be called to start operations of the program.

4. First, we'll create the function to calculate the future value

```python
#! /usr/bin/env python3


def print_welcome(message):
    print(message)
    print()


def calculate_future_value(monthly_investment,
                           yearly_interest,
                           years):

    # convert yearly values to monthly values
    monthly_interest_rate = yearly_interest / 12 / 100
    months = years * 12

    # convert future value
    future_value = 0.0
    for i in range(0, months):
        future_value += monthly_investment
        monthly_interest = future_value * monthly_interest_rate
        future_value += monthly_interest

    return future_value


message = "Welcome to the Future Value Calculator"
print_welcome(message)
```

5. Then, we'll create the main function which will control the flow

```python

    return future_value


def main():
    # display the header
    message = "Welcome to the Future Value Calculator"
    print_welcome(message)
    choice = "y"
    while choice.lower() == "y":
        # get input from the user
        monthly_investment = float(input("Enter monthly investment:   "))
        yearly_interest = float(input("Enter yearly interest rate: "))
        years = int(input("Enter number of years:      "))

        # get the future value
        future_value = calculate_future_value(monthly_investment,
                                              yearly_interest, years)

        # display future value
        print("Future value:\t\t\t\t" + str(round(future_value, 2)))
        print()

        # see if the user wants to calculate another
        choice = input("Continue? (y/n): ")
        print()

    print("Exiting the Future Value Calculator")

```

6. Finally, we need to call the main function (if it exists…)
   a. Note: there are two (2) underlines in each side of *name* and *main*.

```python

    print("Exiting the Future Value Calculator")


if __name__ == "__main__":
    main()

```

7. Now run it…
    a. Use the same values for your image.

Screen Capture #2 (2 points)

```
"C:\Users\Saddleback\PycharmProjects\As
Welcome to the Future Value Calculator

Enter monthly investment:    100
Enter yearly interest rate:  12
Enter number of years:       10
Future value:                23233.91

Continue? (y/n): n

Exiting the Future Value Calculator

Process finished with exit code 0
```

Default Argument Values

It is possible to assign default values to one or more arguments in a function.

8. Modify the future_value_2.py file, setting default values for the calculate_future_values function

```
def calculate_future_value(monthly_investment,
                           yearly_interest,
                           years=20)

    # convert yearly values to monthly values
    monthly_interest_rate = yearly_interest / 12 / 100
```

9. To test we'll need to remove the years parameter when calling the calculate_future_value function

```
    # get the future value
    future_value = calculate_future_value(monthly_investme
                                          yearly_interest)
```

10. Now test it…
    a. Additionally, the number of years input will have no impact (you can remove it if you want)
    b. Use the same values for your image.

Screen Capture #3 (1 points)

```
"C:\Users\Saddleback\PycharmProjects\
Welcome to the Future Value Calculato

Enter monthly investment:   10
Enter yearly interest rate: 12
Enter number of years:      0
Future value:               9991.48

Continue? (y/n): n

Exiting the Future Value Calculator

Process finished with exit code 0
```

Named Arguments

By default, when you passed parameters to a function, the function arguments must be in the same order as the parameters being passed. When a function has many arguments, this can get messy. A solution to this is to use named arguments.

11. Continue to modify the future_value_2.py file, using the names of the arguments and setting them to the input variables names

```
        # get the future value
        future_value = calculate_future_value(monthly_investment=monthly_investment,
                                              yearly_interest=yearly_interest,
                                              years=years)
```

12. To test we'll need to change the order around

```
        # get the future value
        future_value = calculate_future_value(monthly_investment=monthly_investment,
                                              years=years,
                                              yearly_interest=yearly_interest)
```

13. Run to test:
    a. Use the same values for your image.

Screen Capture #4 (2 points)

Code Validation – future_value_2.py (2 points)

```
"C:\Users\Saddleback\PycharmProjects\A
Welcome to the Future Value Calculator

Enter monthly investment:   100
Enter yearly interest rate: 10
Enter number of years:      5
Future value:               7808.24

Continue? (y/n): n

Exiting the Future Value Calculator

Process finished with exit code 0
```

Local vs Global Variable

Scope

Scope refers to the visibility of variables:

Variables created in a function will not be *visible* in other functions and is considered *local* to the function it was created in. To use a local variable in another function, it must be passed to the function.

Variables created outside all functions are visible to all functions and do not need to be passed to be used.

Local Variables

We've already been using local variables. We get these three values and since the variables are created in a function, they are local variables.

```
    choice = "y"
    while choice.lower() == "y":
        # get input from the user
        monthly_investment = float(input("Enter monthly investment:   "))
        yearly_interest = float(input("Enter yearly interest rate: "))
        years = int(input("Enter number of years:      "))

        # get the future value
```

But we use the values in the calculate_future_value function which is why they must be passed into that function to be used.

```
        # get the future value
        future_value = calculate_future_value(monthly_investment=monthly_investment,
                                              yearly_interest=yearly_interest,
                                              years=years)
```

Global Variables

Global variables are read-only in functions.

```
years = 20


def print_welcome(message):
    print(message)
    print()


def calculate_future_value(mo
                           yea

    # convert yearly values t
    years = 10
    months = years * 12
```

Shadows name 'years' from outer scope more... (Ctrl+F1)

Because of this, they are more used to hold constant values which adds to the readability of the code.

```
months_per_year = 12


def print_welcome(message):
    print(message)
    print()


def calculate_future_value(monthly_investment,
                           yearly_interest, years):

    # convert yearly values to monthly values
    monthly_interest_rate = yearly_interest / months_per_year / 100
    months = years * months_per_year
```

User Modules

A module, like functions is reusable code only instead of a single function, it's a file.

Creating a Module

14. Create a new file named **temperature.py.**
15. Code as follows:

temperature.py

```python
#! /usr/bin/env python3


def to_celsius(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9
    return celsius


def to_fahrenheit(celsius):
    fahrenheit = celsius * 9/5 + 32
    return fahrenheit


# the main() function is used to test the conversion functions
# this code isn't run if this isn't the main module
def main():
    for temp in range(0, 212, 40):
        print(temp, "Fahrenheit =", round(to_celsius(temp)), "Celsius")

    for temp in range(0, 100, 20):
        print(temp, "Celsius = ", round(to_fahrenheit(temp)), "Fahrenheit")


# if this module is the main module, call the main()
#  function to test the conversion functions
if __name__ == "__main__":
    main()
```

16. Run to test…

Screen Capture #5 (2 points)

```
C:\Users\Saddleback\AppData\Local\Prog
0 Fahrenheit = -18 Celsius
40 Fahrenheit = 4 Celsius
80 Fahrenheit = 27 Celsius
120 Fahrenheit = 49 Celsius
160 Fahrenheit = 71 Celsius
200 Fahrenheit = 93 Celsius
0 Celsius = 32 Fahrenheit
20 Celsius = 68 Fahrenheit
40 Celsius = 104 Fahrenheit
60 Celsius = 140 Fahrenheit
80 Celsius = 176 Fahrenheit

Process finished with exit code 0
```

Documenting a Module

To document a module, just add docstrings (three “”” double quotes).

17. Add docstrings as follows:

temperature.py

```python
#! /usr/bin/env python3
"""
This module contains functions for converting
temperature between degrees Fahrenheit
and degrees Celsius
"""


def to_celsius(fahrenheit):
    """
    This converts degrees Fahrenheit to Celsius
    :param fahrenheit: This is the degrees Fahrenheit to convert
    :return: The converted Celsius value
    """
    celsius = (fahrenheit - 32) * 5/9
    return celsius


def to_fahrenheit(celsius):
    """
    This converts degrees Celsius to Fahrenheit
    :param celsius: This is the degrees Celsius to convert
    :return: The converted Fahrenheit value
    """
    fahrenheit = celsius * 9/5 + 32
    return fahrenheit
```

Importing a Module

import *module_name* [as *namespace*]

18. Create a new python **convert_temp.py** file. (Attach this file when submitting your assignment.)
19. Start by importing the temperature module
    a. This will import into the default namespace

```
#! /usr/bin/env python3

import temperature
```

20. Add a function to print the menu

```
import temperature


def display_menu():
    print("MENU")
    print("1. Fahrenheit to Celsius")
    print("2. Celsius to Fahrenheit")
    print()
```

21. Followed by the function to get the temperature and convert to either Fahrenheit or Celsius

```

def convert_temp():
    option = int(input("Enter a menu option: "))
    print()
    if option == 1:
        f = int(input("Enter degrees Fahrenheit: "))
        c = round(temperature.to_celsius(f), 2)
        print("Degree Celsius:", c)
    elif option == 2:
        c = int(input("Enter degrees Celsius: "))
        f = round(temperature.to_fahrenheit(c), 2)
        print("Degree Fahrenheit:", f)
    else:
        print("You must enter a valid menu number.")

```

22. Finally add the main() function and the main() calling

```
def main():
    display_menu()
    again = "y"
    while again.lower() == "y":
        convert_temp()
        print()
        again = input("Convert another temperature? (y/n): ")
        print()

    print("Cya")


if __name__ == "__main__":
    main()
```

23. Run to test…

Screen Capture #6 (2 points) Show both calculations

Code Validation – temperature.py & convert_temp.py (2 points)

```
"C:\Users\Saddleback\PycharmProjects\Assi
MENU
1. Fahrenheit to Celsius
2. Celsius to Fahrenheit

Enter a menu option: 1

Enter degrees Fahrenheit: 86
Degree Celsius: 30.0

Convert another temperature? (y/n): y

Enter a menu option: 2

Enter degrees Celsius: 30
Degree Fahrenheit: 86.0

Convert another temperature? (y/n): n

Cya

Process finished with exit code 0
```

24. To import into a specified namespace
    b. This will import into the default namespace

```
#! /usr/bin/env python3

import temperature as temp

```

25. Then update the calls to the module

```

def convert_temp():
    option = int(input("Enter a menu option: "))
    print()
    if option == 1:
        f = int(input("Enter degrees Fahrenheit: "))
        c = round(temp.to_celsius(f), 2)
        print("Degree Celsius:", c)
    elif option == 2:
        c = int(input("Enter degrees Celsius: "))
        f = round(temp.to_fahrenheit(c), 2)
        print("Degree Fahrenheit:", f)
    else:
        print("You must enter a valid menu number.")

```

Standard Modules

There are many standard built-in modules that you can import into your python projects. By importing modules like datetime, math, random, sqlite3, etc., you can greatly add functionality with little work. In the next section, we'll work with the random module.

random

As you would expect, the random module provides functions that allows you to generate "random" numbers or a range of random numbers.

26. Start by adding the *random* module

```
#! /usr/bin/env python3

import random
```

27. Then code as follows and run:

Screen Capture #7 (1 point)

```
#! /usr/bin/env python3

import random

number = random.random()
print("The number is: " + str(number))
```

```
Run: test | test_temp | blackjack | test
C:\Users\Saddleback\AppData\Local\Programs
The number is: 0.9964275753954317

Process finished with exit code 0
```

randint

The randint returns a number between the min and max (inclusive).

randint(min, max)

28. Modify as follows and run:

Screen Capture #8 (1 point)

```
#! /usr/bin/env python3

import random

number = random.randint(1, 10)
print("The number is: " + str(number))
```

```
Run test
C:\Users\Saddleback\AppData\Local\Programs\
The number is: 9

Process finished with exit code 0
```

Guess the Number Game Application

You should try to code this on your own first but code is provided on the next page if you get stuck.

Requirements:

- Create a new python file named **guess_number.py**. (Make sure you attach this file when submitting your assignment.
- In this file, code to generate a random number between 1 and MAX.
  - Use a global variable to hold the MAX
- Prompt the user to input a guess
- Respond to the user if they got the number correct, their guess was too low or too high
  - If they guessed correct, display the number of guesses it took
- Break the code into functions (suggestions)
  - main to control the flow
  - display_title to display the title
  - play_game to generate the random number, get the user guess, display the guess outcome

The output should look like this:

Screen Capture #9 (2 points) Running screen print

Code Validation – guess_number.py (2 points)

```
C:\Users\Saddleback\venv\Scripts\python.ex
Guess the number!

I'm thinking of a number from 1 to 10

Your guess: 5
Too low.
Your guess: 7
Too high.
Your guess: 6
You guessed it in 3 tries.

Would you like to play again? (y/n): n

Bye!

Process finished with exit code 0
```

```python
#!/usr/bin/env python3

import random

LIMIT = 10


def display_title():
    print("Guess the number!")
    print()


def play_game():
    number = random.randint(1, LIMIT)
    print("I'm thinking of a number from 1 to " + str(LIMIT) + "\n")
    count = 1
    while True:
        guess = int(input("Your guess: "))
        if guess < number:
            print("Too low.")
            count += 1
        elif guess > number:
            print("Too high.")
            count += 1
        elif guess == number:
            print("You guessed it in " + str(count) + " tries.\n")
            return

def main():
    display_title()
    again = "y"
    while again.lower() == "y":
        play_game()
        again = input("Would you like to play again? (y/n): ")
        print()
    print("Bye!")


# if started as the main module, call the main function
if __name__ == "__main__":
    main()
```

Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

Extra Credit #1 – Even or Odd Checker (+1 Extra Credit)

Create a program that checks whether a number is even or odd.

```
Even or Odd Checker

Enter an integer:  33
This is an odd number.
```

Specifications:

- Store the code that gets user input and displays output in the main function.
- Store the code that checks whether the number is even or odd in a separate function.
- Assume that the user will enter a valid integer.

Extra Credit #2 – Dice Roller (+1 Extra Credit)

Create a program that uses a function to simulate the roll of a dice.

```
Dice Roller

Roll the dice? (y/n): y

Die 1: 3
Die 2: 6
Total: 9

Roll again? (y/n): y

Die 1: 1
Die 2: 1
Total: 2
Snake eyes!

Roll again? (y/n): y

Die 1: 6
Die 2: 6
Total: 12
Boxcars!

Roll again? (y/n): n
```

Specifications:

- The program should roll two six-sided dice.
- Store code that rolls a single die in a function
- Store the code that gets the input and displays output in the main function. Whenever it's helpful, use helper functions to split this code into other functions.
- The program should display a special message for two ones (snake eyes) and two sixes (boxcars).