

## ***ASSIGNMENT OVERVIEW***

In this assignment you'll be creating a program called **craps.py**, which will allow the user to play the game of Craps on the computer.

This assignment is worth 50 points and is due on the crashwhite.polytechnic.org server at 23:59:59 on the date given in class.

## ***BACKGROUND***

The basic rules for playing craps are relatively simple:

1. A player rolls two six-sided dice and adds the numbers rolled together.
2. On this first roll, a 7 or an 11 automatically wins, and a 2, 3, or 12 automatically loses, and play is over. If a 4, 5, 6, 8, 9, or 10 are rolled on this first roll, that number becomes the "point."
3. The player continues to roll the two dice again until one of two things happens: either they roll the "point" again, in which case they win; or they roll a 7, in which case they lose.

Playing craps can include a number of variations on this game, and also typically involves betting on various outcomes; those aspects of the game are not covered in this assignment.

## ***PROGRAM SPECIFICATION***

Create a Python program that:

- a. asks the user if they'd like to play craps
- b. asks the user if they need instructions, and provide them if desired
- c. requires the user to press the <Enter> key each time they want to roll the dice
- d. provides status information on the user's roll, the point, and whether the user has won or lost
- e. asks the user if they'd like to play again
- f. has thorough comments, written while coding the program, and supplemented with additional comments before submitting.

## ***DELIVERABLES***

**craps.py**

You should keep a working copy of this file in your home folder on the server, and a backup copy of the file elsewhere. To submit your assignment for grading, copy your **craps.py** file to your directory in **/home/studentID/forInstructor/** at **crashwhite.polytechnic.org** before the deadline.

## ASSIGNMENT NOTES

- This program will require the use of a number of **while** loops, which will repeat rolling the dice until the game is over, and repeat playing the game until the user doesn't want to play anymore.
- To simulate rolling the dice, the program will need to import the **random** module.
- Because you'll be rolling dice a lot, your program should define a function **die\_roll()** that returns the value of a single six-sided die roll. You may define other functions as well.
- Although the program could just repeatedly roll dice for the user without any prompting, part of the fun of playing craps is rolling the dice. The closest we can come to that in this program is asking the user to hit the <Enter> key to roll the dice. This can be accomplished by inserting the following statement into your program at the appropriate location:

```
pause = input("Press <Enter> to roll the dice!")
```

We don't actually care about the value of anything the user enters, so the contents of the variable **pause** aren't important to us.

- One of the main challenges of getting user input is making it appropriately easy: you want your program to have usability. So when the program asks the user if they'd like to play the game, what kind of answer might they give? "Yes"? "yes"? Or just a simple "y"? The statements below go a long way towards solving this problem:

```
play_game = input("Would you like to play the game of Craps? (Y/n)")
while (play_game[0].lower() != "n"):
    # Start the game!
    pause = input("Press <Enter> to roll the dice!")
    .
    .
    .
```

- These statements include aspects of Python that we haven't yet covered. **play\_game[0]** refers to the first character in the string **play\_game**, so whether the user enters "Yes" or "Y", the program will just look at the first character. The method **lower()** converts that first character to a lowercase letter so that we don't have to check for both upper and lowercase situations. Combined, these statements effectively say, "Get the user's input, and if the lowercase version of the initial of their response is NOT an "n", the program will go on to let them play craps."
- One additional convention will make this program even more usable. You're probably familiar with the concept of a default, something that is assumed to be the case unless otherwise specified. In this program, although we want to ask the user if they want to play, it's highly likely that they DO want to play—they are running the program, after all. So let's make playing the default value: unless they specifically say "No", we're going to assume they want to play. This is subtly indicated in the user prompt—the (Y/n) part—by capitalizing the value that will be assumed if the user just hits the <Enter> key.

For our program to be able to recognize this entered value of "" (a pair of quotes with nothing

between them), we have to edit the **while** condition in the program:

```
play_game = input("Would you like to play the game of Craps? (Y/n)")
while (play_game == "" or play_game[0].lower() != "n"):
    pause = input("Press <Enter> to roll the dice!")
    .
    .
```

There's a very subtle point here that you should understand. The new statement condition, **play\_game == "" or play\_game[0].lower() != "n"**, only works when we enter the empty string because we've placed the **play\_game == ""** part of the condition first in that statement. With an **or** statement, once the first condition is **True**, the entire condition is **True**, and Python doesn't bother looking any further in that condition. And that works for us because the second part of that condition would contain an error. Why?

If we had reversed the order of that condition so that it read **play\_game[0].lower() != "n" or play\_game == ""** we'd have an error if someone enters the empty string **""**: the check for the first character of **play\_game** would try to check the empty string, which doesn't have *any* characters in it.

## ***GETTING STARTED***

1. With paper and pencil, and perhaps in collaboration with a partner, identify what the main components are that you'll need to include in your program.
2. Sketch out the basic flow of your program using a flowchart, and write some pseudocode that you can use to begin implementing those main components.
3. Either on your personal computer or on the *crashwhite.polytechnic.org* server, open up two windows: a text editor to write the program (on the left), and a shell to perform test runs of your program on the right.
4. Enter pseudocode in text editor, and then fill in more details for various parts of the code.
5. Save your program with the required name.
6. Copy this initial version of the program to the *crashwhite.polytechnic.org* server (to make sure that the upload/copy process works).

```
$ scp -P 1030 craps.py studentID@crashwhite.polytechnic.org:
/home/studentID/forInstructor
```

7. Switching to the shell, run your program as new features are added, making sure to fix old problems before adding new components. You'll repeatedly run through this edit-run, edit-run process to find bugs and fix them.
8. When your program is completed (but before the deadline), copy it to the server as indicated above. The newer version of your program will replace the old one there.

## ***QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)***

1. Did you use nested **if-else** statements or **if-elif-else** in determining whether the user wins, loses, or has a point to make? Which technique seems more intuitive to you?
2. Is your program “robust,” i.e. will it continue to function without any runtime errors, regardless of what the user enters? We’ll be looking at more sophisticated ways to prevent different types of errors in your programs in the future.

## ***SAMPLE INTERACTIONS***

```
PhileasFogg:Desktop rwhite$ python craps.py
```

```
Would you like to play the game of Craps (Y/n)? y
```

```
Do you need instructions (y/N)? y
```

1. A player rolls two six-sided dice and adds the numbers rolled together.
2. On this first roll, a 7 or an 11 automatically wins, and a 2, 3, or 12 automatically loses, and play is over. If a 4, 5, 6, 8, 9, or 10 are rolled on this first roll, that number becomes the 'point.'
3. The player continues to roll the two dice again until one of two things happens: either they roll the 'point' again, in which case they win; or they roll a 7, in which case they lose.

```
Press <Enter> to roll the dice!
```

```
You rolled a 7 on your first roll...
```

```
You WON!!!
```

```
Wanna play again (Y/n)?
```

```
Press <Enter> to roll the dice!
```

```
You rolled a 6 on your first roll...
```

```
That's your point. Try to roll it again before you roll 7 and lose!
```

```
Press <Enter> to roll the dice!
```

```
You rolled a 5
```

```
Keep on rolling...
```

```
Press <Enter> to roll the dice!
```

```
You rolled a 7
```

```
You rolled a 7 and lost!!!
```

```
Wanna play again (Y/n)? n
```

```
Thanks for visiting! See you next time!
```

```
PhileasFogg:Desktop rwhite$
```