

Unit 4: Loops

Activity 3: The Do While Loop

In this lesson, you will learn about the all-purpose **do while** loop structure.

Objectives:


- Use the **do while** loop structure
- Use the **break** statement to get out of a loop

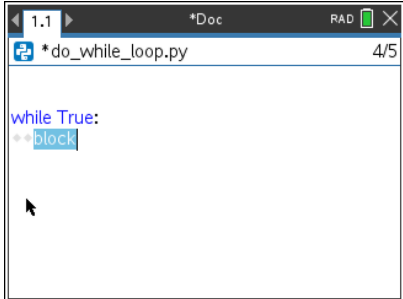
Teacher Tip: Many programming teachers do not like using the **do while** structure because it promotes bad programming habits that can lead to 'spaghetti code' (programs that jump all around). In fact, Python does not have an explicit **do while** structure. The **do while** structure is covered here with the intention that it is addressed appropriately: Use only one **break** statement.

However, one advantage of the **do while** structure is the *possibility* of multiple exit points; but this would be rare, and there's always a workaround to avoid this situation. The **break** statement forces program control to process the first statement after the **do while** loop, so there's no confusion about where the program flow is headed.

What is this thing called *do while* loop?

The **do while** structure creates a more flexible (but possibly hard-to-follow) loop. It repeatedly executes the statements in the loop body. Note that this loop will be executed endlessly, unless a **break** statement is executed somewhere inside the loop body.

If no **break** statement is encountered, then the loop will be an 'infinite' loop. If you accidentally run into an infinite loop on the handheld, press and hold  until the program 'breaks.'



```
1.1 *Doc RAD X  
*do_while_loop.py 4/5  
while True:  
    block
```

The **do while** structure is processed at least once since there's no condition to be met for its entry.

break forces program flow to the statement immediately after the loop.

Python doesn't include an explicit **do while** loop function, so we will model this loop structure using a **while** loop with a condition that is always true.

Example: Random Numbers

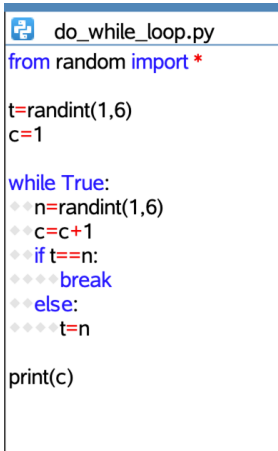
How many random numbers from 1 to 6 can be generated before two consecutive values are the same?

In the program shown to the right, observe the use of **while** loop with a conditional that is always **True** and with a *nested conditional* **break** statement.

Predict how this program works.

When the random integer generator **randint(1,6)** produces two consecutive identical values, the loop will **exit** to process the **print** statement at the end of the program.

The reserved word **break** can be typed using the keyboard.



```
do_while_loop.py  
from random import *  
  
t=randint(1,6)  
c=1  
  
while True:  
    n=randint(1,6)  
    c=c+1  
    if t==n:  
        break  
    else:  
        t=n  
  
print(c)
```

Teacher Tip: One advantage of a **do while** structure is the possibility of multiple **break** points, but this topic should be avoided with students until they have done more coding because it can lead to bad programming habits.

Program: Guess My Number

To demonstrate the use of the **do while** loop, we'll develop a two-player game to guess a random number from 1 through 10. When a player guesses the number, then the program ends with a 'winner' message.

Sample output of such a program is shown to the right.

1. Start a new program called **guess**.
2. Start by setting up (initializing) the game. Identify three variables, the player number (*player*), the computer's or handheld's number (*number*), and the player's guess (*guessnum*).

The computer picks a random number from 1 to 10. We start with the player numbered 0 and the player's guess set to 0. This makes it easy to change the player number as we'll soon see.

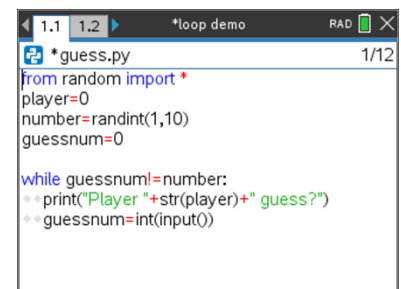
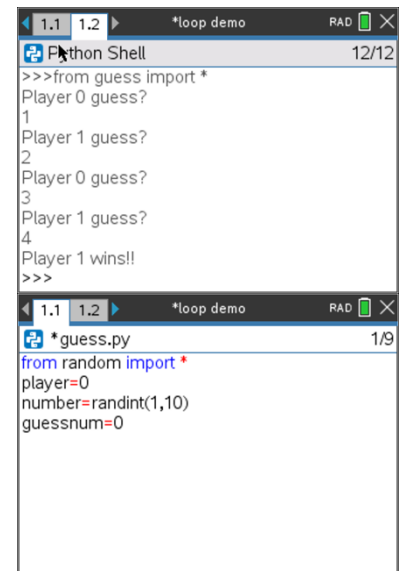
3. Next, build the loop where we first ask the player to enter a guess. The complete statement is:

```
print("Player "+str(player)+" guess?")
guessnum=int(input())
```

The '+' symbol is for *concatenation* of strings. The prompt portion of a **print** statement must be a string so the *player* variable (a number) is converted to a string with the **str()** function found in the **Menu > Built-ins > Type** submenu.

Teacher Tip: Concatenation is the process of combining two strings into one string. The characters of the second string are appended to the end of the characters of the first string. You must use the **str()** function to convert a numeric variable's value into a string value.

The value of **guessnum** variable initialized at the beginning of the program is replaced with the user input value.



4. Next, build the **break** condition. When a player guesses the number, then the loop will **break**. We can use the primitive **if** statement here:

```
if guessnum == number:
    break
```

5. To switch players, use the statement:

```
player = 1 - player
```


Be sure this statement is included inside the **while** loop but not within the **if** conditional. Why?


This unique statement switches the value of **player** between 0 and 1. That is, when **player** is 0 the statement changes it to 1 and when **player** is 1 the statement changes it to 0. Try it!

6. Finally, add in a **print** statement after the loop to congratulate the winner:

```
print("Player "+str(player)+" wins!!")
```

Tip: If you'd rather not see 'player 0' and 'player 1,' just add 1 to the player variable in this statement and in the **print** statement. The user sees a 1 or a 2 even though the computer uses 0 and 1 for the players.

7. Before running the program, save the document (**CTRL S**) in case the program becomes stuck in an infinite loop. If you accidentally run into an infinite loop on the handheld, press and hold  until the program 'breaks.'



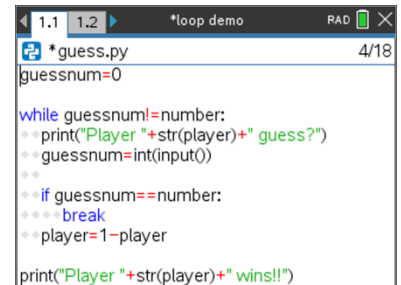
```
*guess.py 1/16
from random import *
player=0
number=randint(1,10)
guessnum=0

while guessnum!=number:
    print("Player "+str(player)+" guess?")
    guessnum=int(input())
    *
    * if guessnum==number:
    * * break
```



```
*guess.py 2/16
player=0
number=randint(1,10)
guessnum=0

while guessnum!=number:
    print("Player "+str(player)+" guess?")
    guessnum=int(input())
    *
    * if guessnum==number:
    * * break
    * * player=1-player
```



```
*guess.py 4/18
guessnum=0

while guessnum!=number:
    print("Player "+str(player)+" guess?")
    guessnum=int(input())
    *
    * if guessnum==number:
    * * break
    * * player=1-player

print("Player "+str(player)+" wins!!")
```

Teacher Tip: When writing programs, as with any document, it is important to save the document regularly so that changes are not lost.