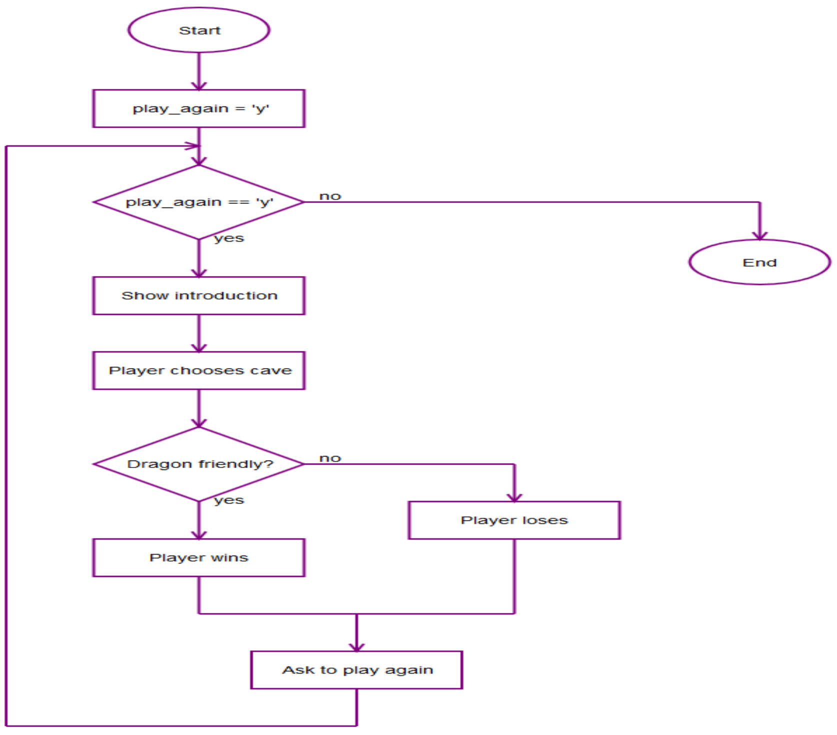
Tutorial: Functions (Dragon Kingdom Game)

The aim is to create a simple game that demonstrates the use of user-defined functions (with arguments, parameters, return values and local variables).

Dragon Kingdom Game - The player is in the Kingdom of Dragons. The dragons live in caves with their large piles of collected treasure. Some dragons are friendly and share their treasure. Other dragons are hungry and eat anyone who enters their cave. The player approaches two caves, one with a friendly dragon and the other with a hungry dragon, but doesn't know which dragon is in which cave (this is generated by the program). The user must choose between the two caves.

The following is an example view of the game running. The player's input is in bold.

```
You are in the Kingdom of Dragons. In front of you,
you see two caves. In one cave, the dragon is friendly
and will share his treasure with you. The other dragon
is hungry and will eat you on sight.
Which cave will you go into? (1 or 2)
1
You approach the cave...
A large dragon jumps out in front of you!
He opens his jaws and...
Gobbles you down!
Do you want to play again? (y or n)
n
```



Dragon Kingdom Game Flowchart. **Change the wording of the game if you wish but use the shown flowchart structure.**

You may need to check the lecture notes!

1. Create a new program called *dragon.py* and save to your H drive.
2. Your program will need to import two modules, so place these at the top of the program. Random provides *randint()* and time provides the *sleep()* function that we will use.

```
import random
import time
```

3. **Show Introduction.** Define a function called *display_intro* that has no parameters and prints out the following when called.

```
You are in the Kingdom of Dragons.  In front of you,
you see two caves. In one cave, the dragon is friendly
and will share his treasure with you. The other dragon
is hungry and will eat you on sight.
```

- Note: Use three quotes around a string for *multiline strings*.
- Remember, a function's definition must come *before* you call the function.
- Add a call to the function in your program to check that it works as expected.

4. **Player Choses Cave**. Define a function called *choose_cave* that has no parameters.
   a) Basic function. This should:
   - define a **local** variable *cave* as an empty string;
   - ask the player if they want to 'Enter cave 1 or 2';
   - check if the input is '1' or '2';
   - **return** the response to the main part of the program.

   Hint: keep the cave number in string format.

   b) Intermediate function You can add input validation to the *choose_cave* function. Add a loop that prompts for cave 1 or 2 until a valid response is entered by the user.

   c) In the main part of the program create a variable called *cave_number* that will hold the value returned by the *choose_cave()* function. Check what is returned from the function for testing purposes by adding a temporary print().

```
print(cave_number)        #Check for Friendly Dragon
```

5. **Check_cave.** Create a function called *check_cave* that has a parameter called *chosen_cave.*
   - Add the following code to the *check_cave* function:

```
print('You approach the cave...')
time.sleep(2)
print('A large dragon jumps out in front of you! ')
print('He opens his jaws and...')
time.sleep(2)
friendlyCave = random.randint(1, 2)
```

- Notes:
  - The time module function sleep() is used to pause the program for 2 seconds (to build some suspense in the game).
  - The randint() function will return either 1 or 2 and store this in friendlyCave to indicate the cave with the friendly dragon.

- Now, create a condition to check if the player wins or loses.
  - If the player's chosen cave (stored in chosen_cave) is equal to the randomly generated value (stored in friendlyCave). Print 'Gives you his treasure!'
  - Otherwise, print 'Gobbles you down!'
  - The value in friendlyCave is an integer because randint() returns integers. The value in chosen_cave is a string. Convert friendlyCave to compare two strings.

- Remember to add a call to the check_cave function passing the cave_number as an argument.

6. **Calling the Functions.** The main part of your program should now look like this.

```
display_intro()
cave_number = choose_cave()
check_cave(cave_number)
```

7. **Challenge 1 - Ask to Play Again.**
   Add the feature to check if the user wants to play again (see flowchart).

8. **Challenge 2 – Multiple caves**.
   Extend the above program. Create another program *dragon_multi.py* that has 4 caves (instead of 2) and include appropriate messages for the player selecting cave 1, 2, 3 or 4. Draw the amended flowchart first, then create the program.

Additional Question

9. Access one of your tutorial solutions from week 5 – **'Guess the Word'** and rewrite to include your own user-defined functions.