From Snap! to Python - Number guessing game

UBMS Snap! Programming Summer 2023

July 10, 2023

Short program: Guess the Number

Colab solution link: tinyurl.com/guessTheNumberSolution.

- This example also demonstrates an exemplary solution path:
  1. Understand what's asked from you (requirements)
  2. Understand what the program needs from you (input)
  3. Understand what's the result supposed to look like (output)
  4. Write the process as pseudocode (without syntax)
  5. Create a process diagram (with commands)
  6. Code the Python program (source code)
  7. Run, test and debug the source code
  8. Fix pseudocode/diagram accordingly.
  9. Identify extensions.
  10. Implement extensions (repeat steps 4-8).
- Write a 'Guess the number' game. When you run the program, the output should look like this:
- The program should generate a random number between 1 and 20.
- Enter the source code into the IDLE file editor, or into Colab, and save as `guessTheNumber.py`.
- Solution path/pseudocode (code highlighted)
  1. `import random` module.

```
Enter number between 1 and 20:
Take a guess: 10
Your guess is too high.
Take a guess: 2
Your guess is too low.
Take a guess: 8
Your guess is too high.
Take a guess: 3
Your guess is too low.
Take a guess: 7
Good job! You guessed my number in 5 guesses!
```

Figure 1: Desired output of guessTheNumber.py

2. Generate a `random` number.
3. Store number in `num`.
4. Set `attempt` (number of guesses) to 0.
5. Get `input` number `guess` from user.
6. Increase `attempt` by 1
7. Check if `guess` is the same as `num`
8. Print success message and `attempt` value
9. End program
10. Otherwise, check if `guess` is smaller than `num`
11. Print information
12. Otherwise, check if `guess` is larger than `num`
13. Print information
14. Return to step 3

- BPMN Process diagram:

Create a Snap! solution

- Turn this pseudocode into a Snap! game:

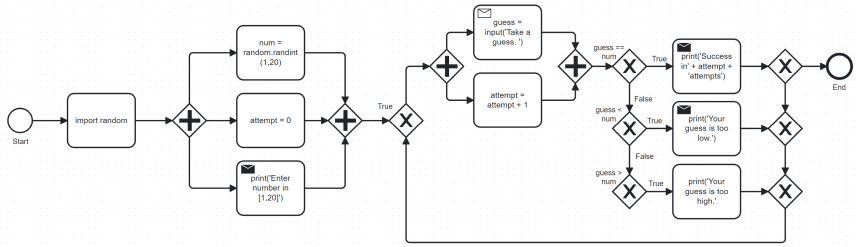
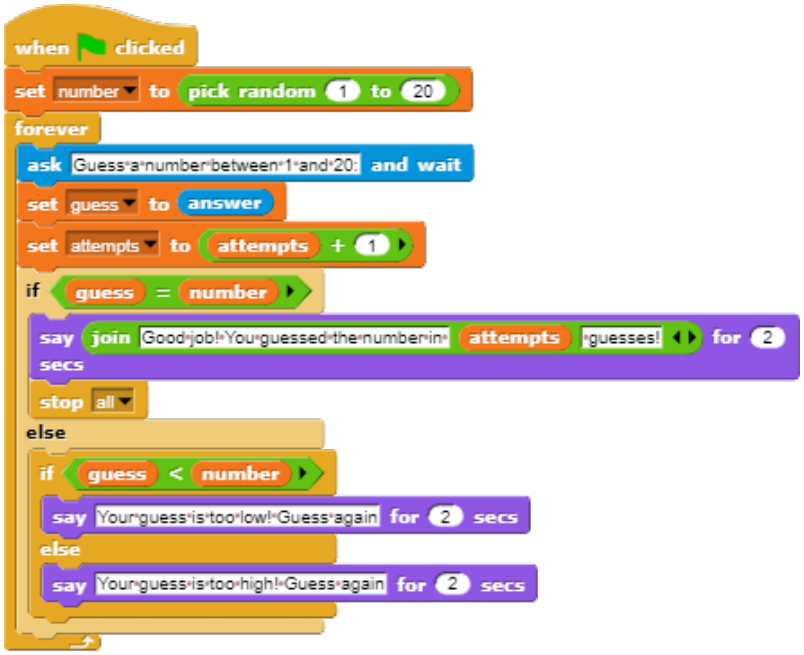


Figure 2: Flow diagram for guessTheNumber.py

```
// Create a variable to store the random number
set number to pick random(1, 100)

// Create a forever loop to keep the game running
forever:

   // Ask the user to guess the number
   ask("Guess a number between 1 and 100:")

    // Get the user's guess
    set guess to answer

    // Compare the user's guess to the random number
    if guess = number:
        // The user guessed correctly
        say("Your guess is correct!")
        stop
    else:
        if (guess < number):
            // The user's guess is too low
            say("Your guess is too low!")
        else:
            // The user's guess is too high
            say("Your guess is too high!")
```

- Snap! solution:



Python prerequisites: `random`, `forever`, `elif`, `int`

- The `random` module is a built-in library with functions that create pseudo-random numbers, like `random.randint`:

```
import random
# create a random integer between 1 and 20
print(random.randint(1,20))
```

```
10
```

- The `forever` loop from Snap! becomes a `while True` loop in Python:

```
while True:
   print('It is true!')
```

- The following loop breaks after 5 iterations because we made it:

```
i = 1
while True:
```

```
    print('looping: ',i)
    i+=1
    if i > 5:
        break

looping:  1
looping:  2
looping:  3
looping:  4
looping:  5
```

- In Python, there's not only `if` and `else` but also `elif` to test a series of conditions - with any number of `elif` following one another. Example:

```
i = 9
if i < 5:
    print('i is smaller than 5')
elif i > 5:
    print('i is greater than 5')
else:
    print('i is 5')

i is greater than 5
```

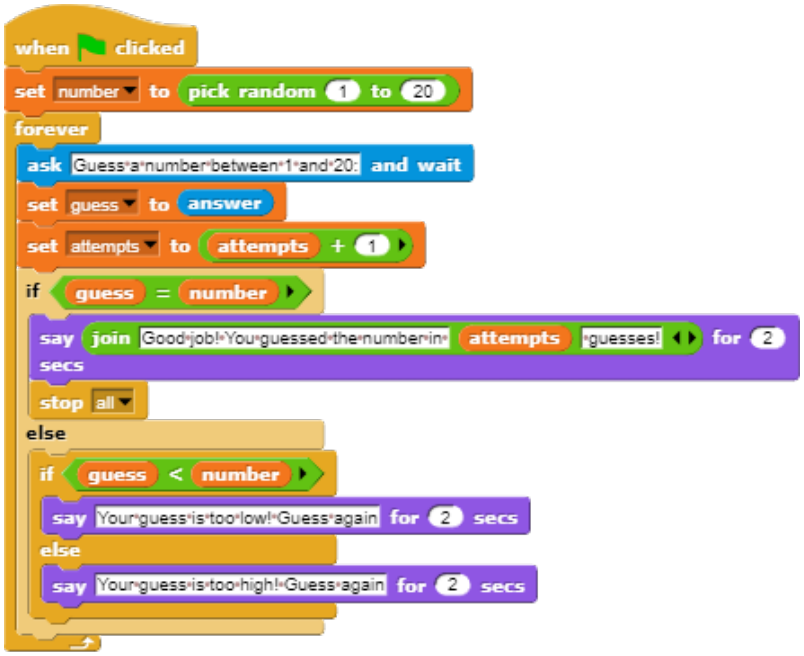
- Input is always a string. If you want to work with it as a number, it must be convertible and converted:

```
number = input('Enter a number: ')
print('Input data type: ', type(number))
try:
    print('Integer from input:', int(number))
except ValueError:
    print('Wrong input value - need integer.')

Enter a number:
```

- You know two conversion functions now: `int` and `str`: `int` converts its argument to an integer (if possible), and `str` converts its arguments to a string (which is always possible).

Solution code

- Snap! solution:



- Python solution (GitHub):

```
# import random module
import random
# pick random number between 1 and 20
num = random.randint(1,20)
# set attempts counter to 0
attempt = 0
# ask user for number guess
print('Enter number between 1 and 20: ')
# infinite loop until number is guessed
while True:
    guess = int(input('Take a guess: '))
    attempt = attempt + 1
    if guess < num:
        print('Your guess is too low.')
        continue
```

```
    elif guess > num:
        print('Your guess is too high.')
        continue
    else:
        print('Good job! You guessed my number in ' + str(attempt) + ' guesses!')
        break

Enter number between 1 and 20:
Take a guess:
```

- Program extensions:
  1. Make program safe against no/wrong input (exception handling): currently, it terminates with an error if a floating-point number or a letter or nothing is entered by mistake.
  2. Exchange the infinite `while` loop by a `for` loop with a set number of allowed guesses (most games don't go on forever).
- What did you learn?
  1. For best productivity and learning, follow a solution path - don't just "code away"
  2. For best learning effects find different solutions to the same problem.
  3. For best results, handle exceptions. Balance exception handling with usability and performance.
  4. There is always more than one solution, usually many. There is no best solution to a programming problem that satisfies all requirements, even the unspoken ones, equally well.