PAYS 2023

INTRODUCTION TO PROGRAMMING USING PYTHON

5: Booleans and random

Alexandra Papoutsaki

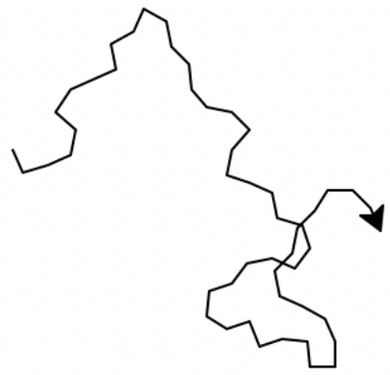
she/her/hers

Lecture 5: Booleans and random

- `random` module
- booleans
- conditionals

walk function

```
def walk(num_steps, step_size):
    for i in range(num_steps):
        angle = randint(-90, 90)
        right(angle)
        forward(step_size)
```



random module

- http://docs.python.org/library/random.html
- It generates *pseudo-random* numbers
  - the numbers are not technically random, they're generated based on an algorithm (for most purposes, this is pretty good!)
- If you want truly random numbers, check out http://www.random.org/

Useful functions

- `random` - returns a random float between 0 and 1.
- `uniform(a, b)` - returns a random float between `a` and `b`.
- `randint(a, b)` - returns a random integer between `a` and `b`.
- samples from many other distributions
  - beta
  - exponential
  - gamma
  - normal

Importing only one function

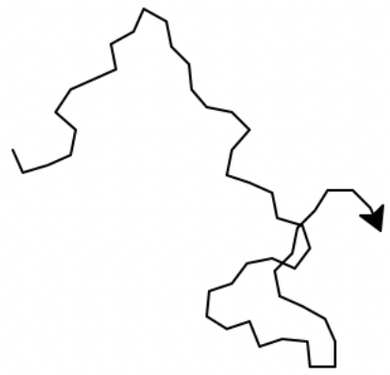
- For now, we will only use the `randint` function.
- Rather than importing everything (*) we will be specific:

```
from random import randint
```

```
>>> for i in range(100):
...     print(randint(0,10))
...
8
9
5
0
1
7
```

walk function

```
def walk(num_steps, step_size):
    for i in range(num_steps):
        angle = randint(-90, 90)
        right(angle)
        forward(step_size)
```

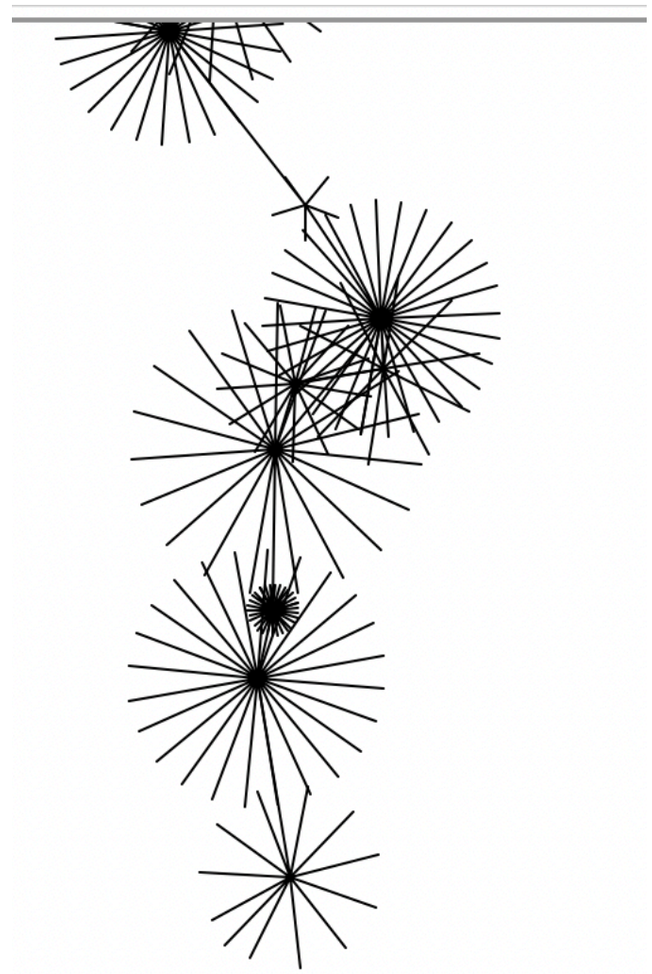


`pretty_picture` function

```
def pretty_picture():
    for i in range(10):
        # get some random values
        spokes = randint(5, 30)
        length = randint(10, 60)
        angle = randint(-90, 90)
        move = randint(20, 100)

        # move randomly somewhere else
        right(angle)
        forward(move)

        # draw a random star there
        asterisk_star(length, spokes)
```



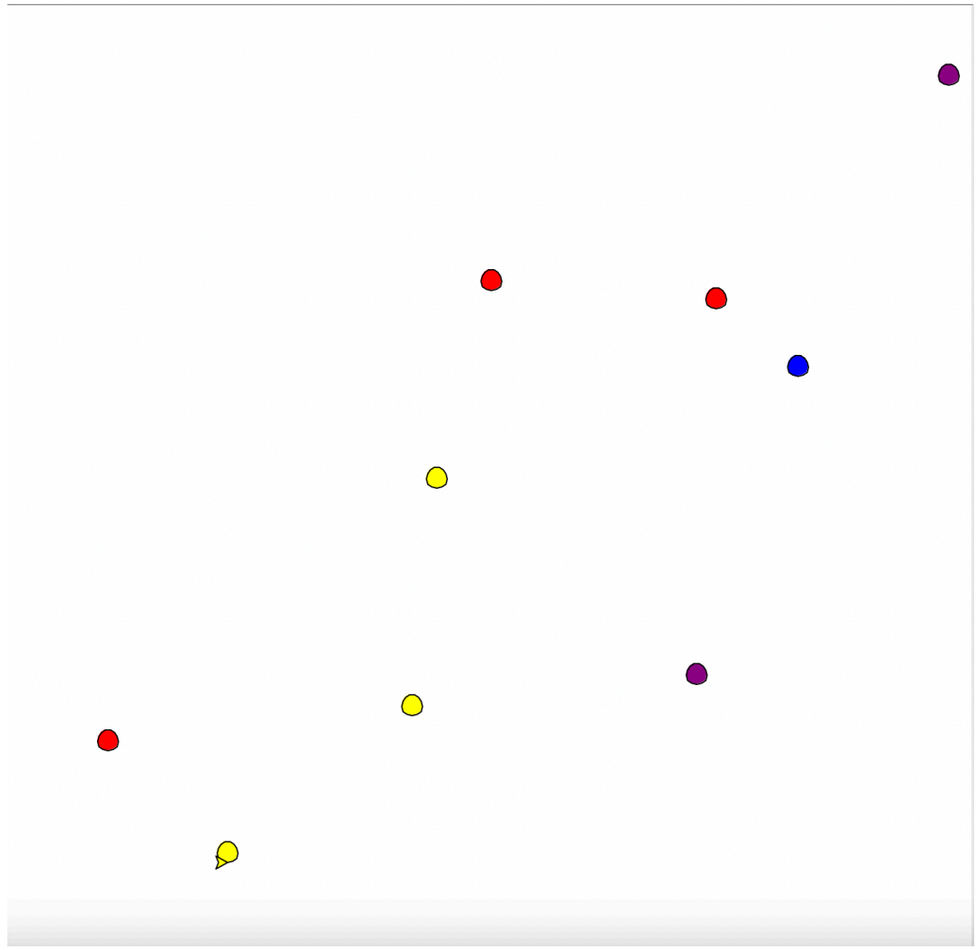
add_circles function

```
def add_circles(number):
    """ Add number colored circles of radius 4 randomly through the screen """
    x_range = int(window_width() / 2)
    y_range = int(window_height() / 2)

    for i in range(number):
        x = randint(-x_range, x_range)
        y = randint(-y_range, y_range)

        # set the fill color of the circles
        # setcolor_xy(x, y)
        setcolor_random()

        pu()
        goto(x, y)
        pd()
        begin_fill()
        circle(8)
        end_fill()
```



Lecture 5: Booleans and random

- `random` module
- booleans
- conditionals

Booleans

- So far, we have seen three types: `int`, `float`, `string`
- Python contains one more type, `bool` (stands for boolean)
- `bool` can only take the value True or False
- They generally result from asking T/F questions

T/F questions we can ask

- == (equal)
  - notice that '=' is the assignment operator while '==' asks whether two things are equal
- != (not equal)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Examples

```
>>> 10 < 0
False
>>> 11 >= 11
True
>>> 11 > 11.0
False
>>> 11 > 10.9
True
>>> 10 == 10.1
False
>>> "test" == "test"
True
>>> "test" == "TEST"
False
>>> 10 != 10
False
>>> 10 != 11
True
>>> "banana" < "apple"
False
>>> type(True)
<class 'bool'>
>>> type(0 < 10)
<class 'bool'>
```

Combining booleans

- We can also combine boolean expressions to make more complicated expressions
- What kind of connectors might we want?

and

- `<bool expression> and <bool expression>`
- only returns `True` if both expressions are `True`
- otherwise, it returns `False`

A	B	A and B
T	T	T
T	F	F
F	T	F
F	F	F

```
>>> x = 5
>>> x < 10 and x > 0
True
>>> x = -1
>>> x < 10 and x > 0
False
```

or

- `<bool expression> or <bool expression>`
- returns `True` if either expression is `True`
- returns `False` only if both expressions are `False`

A	B	A or B
T	T	T
T	F	T
F	T	T
F	F	F

```
>>> x = 5
>>> x < 10 or x > 0
True
>>> x = -1
>>> x < 10 or x > 0
True
```

not

- `not <bool expression>`
- Negates the expression:
  - if the expression evaluates to `True` returns `False`
  - if the expression evaluates to `False` returns `True`

A	not A
T	F
F	T

```
>>> not 5==5
False
```

Lecture 5: Booleans and random

- `random` module
- booleans
- conditionals

`if` statement

- the key use of `bool` is to make decisions based on the answers
- the `if` statement allows us to control the flow of the program based on the result of a boolean expression

```
if bool_expression:
    # do these statements if the bool_expression is True
    statement1
    statement2
statement3
```

`if` statement

- the `if` statement is called a "control" statement in that it changes how the program flows
- As the program runs, it evaluates the boolean expression. If it evaluates to `True`, it executes all of the statements under the `if` block and then continues on:
  - It will execute `statement1`, `statement2` and then `statement3`
- Otherwise, (i.e. the boolean expression evaluates to `False`), it will skip these statements and continue on (i.e. just execute `statement3`).

simple_if function

```
def simple_if(num):
    """
    Given a number, prints out some comments based on
    the size of the number
    """
    if num > 10:
        print("That's a big number")

    print("I'm done")
```

`input` function

- Built-in function to read input from the keyboard
- It takes a string as a parameter and displays the string to the user
- Then waits for the user to enter some text. The program doesn't continue until the user hits enter/return
  - whatever the user typed will be returned by the input function as a string
- Note: if you want to convert the user input to a number, you need to use the `int(...)` or `float(...)` functions

If-else statement

- Sometimes we'd also like to do something if the bool expression evaluates to `False`. In this case, we can include an `else` statement.
- `if <bool expression>:`

```
    # execute these statements if the bool expression evaluates to True
    statement1
    statement2
 else:
    # do these statements if the bool is False
    statement3
    statement4
statement5
```

`If-else` statement

- if the boolean expression evaluates to True,
  - execute `statement1`, `statement2`, then `statement5`
- else (i.e. the boolean expression evaluates to False)
  - execute `statement3`, `statement4`, then `statement5`.

name_analysis function

```
def name_analysis():
    """
    Prompts the user for their name and gives a subjective
    analysis of the name
    """
    name = input("Enter your name: ")

    if name == "Alexandra" or name == "Zilong":
        print(name + ", that's a great name!")
    else:
        print(name + ", that name is ok!")

    print("Nice to meet you, " + name)
```

elif statement

- ```
  if <bool expression>:
      statement1
  elif <bool expression>:
      statement2
      ... # we can have as many elif blocks as we want
  else:
      statement3


  statement4
  ```
  ```

`elif` statement

- The program starts with the first `if` statement.
- If it is True, it executes the statements in the `if` block (here, only `statement1`) then goes to the end (here, `statement4`) and continues
- If it is false, it goes to the first `elif` and checks if it is true. If it is true, it executes the statements in the elif block (here, `statement2`) then goes to the end (here, `statement4`) and continues
- The program will keep going down the list of `elif` statements as long as none of them are true
- If they are all false, then it will execute the statements under `else`
- `elif` avoids redundant calculations: if we know things are mutually exclusive, then once we find one that is true, we don't check the others (jump directly outside the `if-elif-else` block)

setcolor_xy function

```
def setcolor_xy(x, y):
    """ Set the fill color based on x, y coordinates """
    if x < 0 and y < 0:
        fillcolor("blue")
    elif x < 0 and y > 0:
        fillcolor("purple")
elif x > 0 and y < 0:
    fillcolor("red")
else:
    fillcolor("yellow")
```

setcolor_random function

```
def setcolor_random():
    """ Set the fill color randomly from: blue, purple, red and yellow """
    color = randint(1, 4)

    if color == 1:
        fillcolor("blue")
    elif color == 2:
        fillcolor("purple")
    elif color == 3:
        fillcolor("red")
    else:  # color == 4
        fillcolor("yellow")
```

temperature function

```
def temperature_report(temperature):
    """ Converts a numerical temperature to one of: hot, warm, cool or cold """
    if temperature > 80:
        temp = "hot"
    elif temperature > 70:
        temp = "warm"
    elif temperature > 50:
        temp = "cool"
    else:
        temp = "cold"

    return temp
```

Resources

- Textbook: Chapter 7 and Chapter 8.
- conditional-turtle.txt
- conditionals.txt

Practice Problems

- Practice 2 (solution)