NIHR | Applied Research Collaboration South West Peninsula

@penCHORD_UoE
@peninsula_ARC

HSMA 5 Coding Club
Coding Club 2
Dr Daniel Chalk

"Video Olympics"

HSMA 5

#hsma5isalive

Coding Club

HSMA Coding Clubs are sessions to allow you to practice your coding skills together in fun ways! Each club will get you to practice some of the skills that you've recently learned on the course.

In each coding club, you'll either split into groups or work as one group, depending on numbers.

Coding Club

In each Coding Club session, you'll be given a task (or choice of tasks) to work on in your group.

You'll have about 1 hour 30 minutes to undertake the task, and then the group(s) will share what they've done at the end of the session.

Work produced in the coding clubs will be uploaded to the hsma5-student GitHub organisation.

Player 1

For today's task, you are going to write a simple game in Python.

You can write any game you like. The only conditions are that :
- the game must be written using Object Oriented code
- the game must include the use of a generator function

Let's remind ourselves about generator functions...

Generator Functions

Conventional functions in Python are called, and then run with some (optional) inputs, and then finish (usually by returning some output).

Generator functions remember where they were and what they did when control is passed back (they retain their “local state”), so that they can continue where they left off, and can be used as powerful *iterators* (for and while loops are other examples of *iterators*).

This is *very* useful where we want state to be maintained (e.g. during a simulation run)

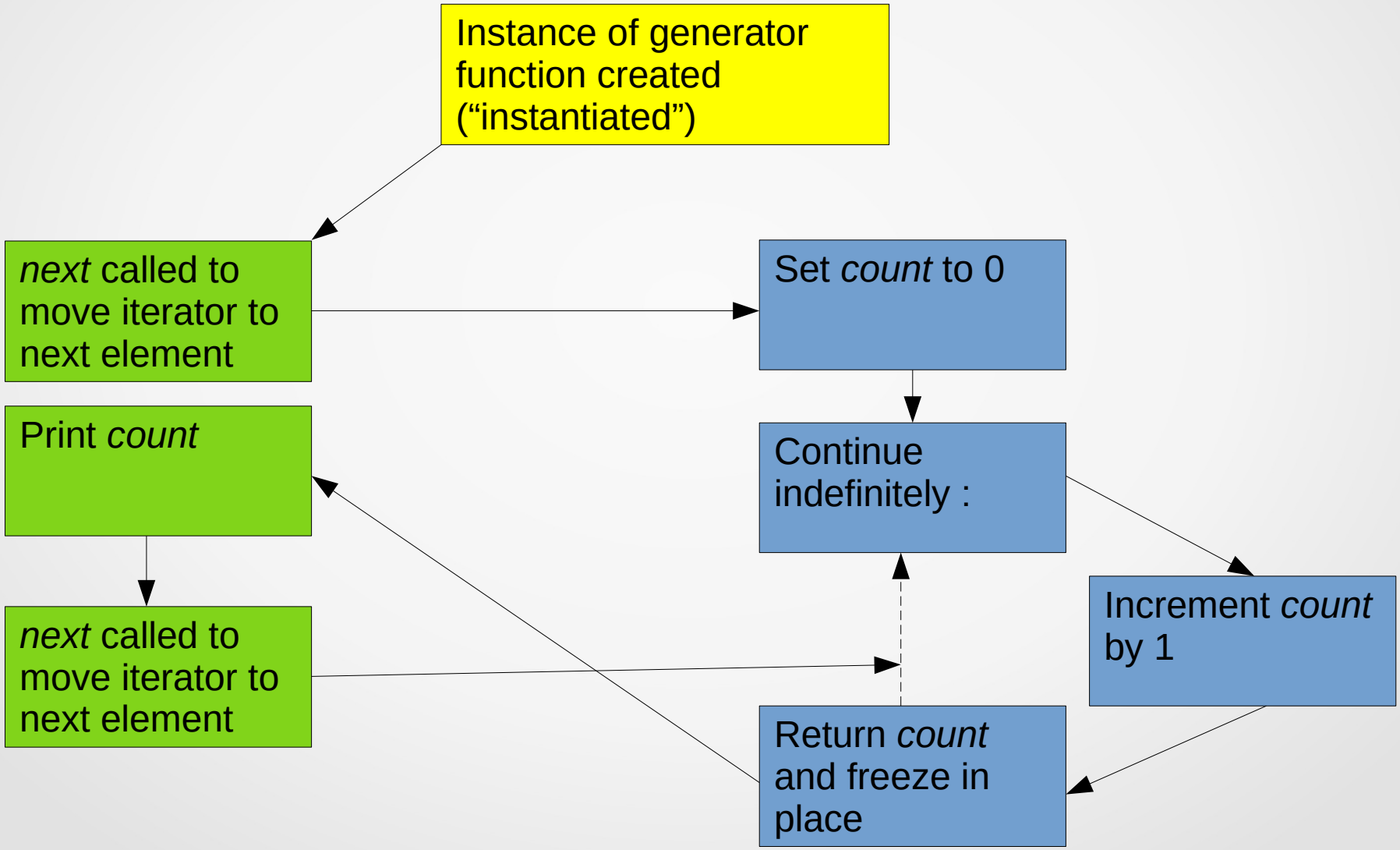
Let’s look at a very simple example of a generator function to see how they work.

```python
# We define a generator function in the same way as a conventional function
def keep_count_generator():
    # We'll set count to 0
    count = 0

    # Keep doing this indefinitely
    while True:
        # Add 1 to the count
        count += 1

        # The 'yield' statement identifies this as a generator function.
        # Yield is like return, but it tells the generator function to freeze
        # in place and remember where it was ready for the next time it's
        # called
        yield count

# We run a generator function a little differently than a conventional
# function.  Here, we create an 'instance' of the generator function, and then
# we can work with that instance.  This also means we can have multiple
# instances of the same generator function, all in different "states"
my_generator = keep_count_generator()

# Let's print the output of the generator function 5 times.  The 'next'
# statement is used to move to the next element of the iterator.  Here, that
# means the generator unfreezes and carries on until it hits another yield
# statement.
# I've not put these print statements in a for loop to make it clear what's
# happening.
print (next(my_generator))
print (next(my_generator))
print (next(my_generator))
print (next(my_generator))
print (next(my_generator))

# The above wouldn't work with a conventional function - every time the
# function is called, it would reset the count to 0, add 1 and then return
# a value of 1
def conventional_keep_count():
    count = 0

    while True:
        count += 1
        return count

a = conventional_keep_count()
print(a)
a = conventional_keep_count()
print(a)
a = conventional_keep_count()
print(a)
a = conventional_keep_count()
print(a)
a = conventional_keep_count()
print(a)
```

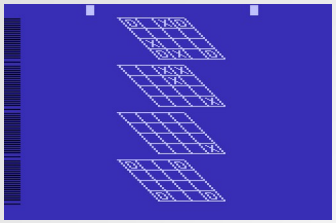
```
1
2
3
4
5
```

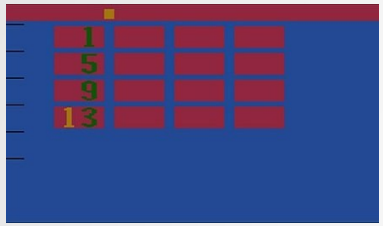
```
1
1
1
1
1
```

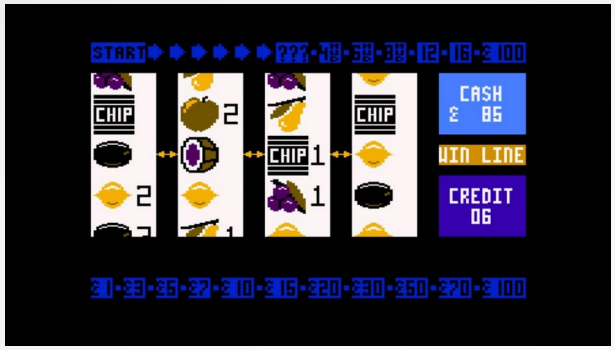
Generator Functions

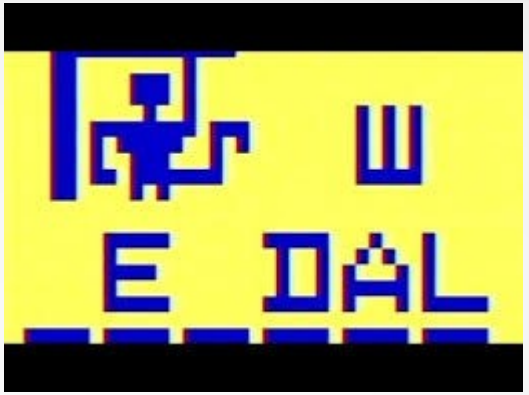


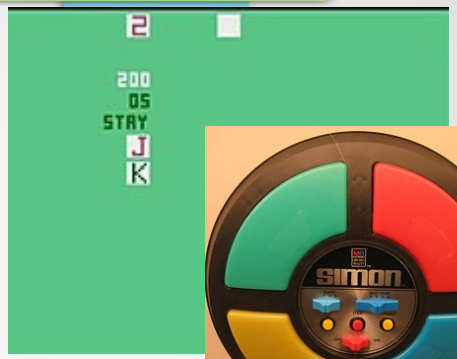
Your task











You are now going to, in your group(s), design and write a game that uses *object oriented code* and which contains at least one *generator function*.

At the end of the session, we'll ask the group(s) to share their game(s) by uploading them to the GitHub repo for this coding club https://github.com/hsma5-student/coding_club_2 and we will play them live!

We'll dip in and out of the breakout rooms to listen in – let us know if you need some help!