while loop aggregation (class slides)

CSc 110 - while loop aggregation

`while` loops

- Using an index variable:
  - Defined before the loop
  - Used in the condition of the loop
  - Changed within the loop
- Using a temporary variable for aggregation:
  - Defined before the loop
  - Changed within the loop
  - Returned outside the loop

`while` loops with aggregation

```python
total = 0
index = 1
while index <= 5:
    print('adding ' + str(index))
    total = total + index
    index = index + 1

print(total)
```

```
adding 1
adding 2
adding 3
adding 4
adding 5
15
```

Write a function

1. Its name is `sum_all`
2. It takes two numeric arguments: `low` and `high`
3. It runs a loop that iterates through the values `low` and `high` summing all values (HINT: you need to create a variable that will aggregate or accumulate the sum)
4. It returns the sum of all values between `low` and `high`
5. Use `while` (define an index before the loop, use index in the `while` condition, change the index inside the loop)

Write a function – solution

Write more test cases for this function.

```
def sum_all(low, high):
  total = 0
  current_number = low
  while current_number <= high:
    total += current_number
    current_number += 1
  return total

def main():
  print( sum_all(1, 2) ) # 3
  print( sum_all(0, 5) ) # 15

main()
```

```
3
15
```

Write a function

1. Its name is `vowels_only`
2. It takes a `string` argument
3. It builds a new string containing only the vowels in the `string` argument

4. It returns new string with vowels only (define an `index` before the loop, use index in the `while` condition, change `index` inside the loop)

```
print( vowels_only("banana") ) # "aaa"
print( vowels_only("fly") ) # ""
```

Write a function – solution

Write more test cases for this function.

```
def vowels_only(string):
  new_string = ""
  index = 0
  while index < len(string):
    if string[index] in "aeiou":
      new_string += string[index]
    index += 1
  return new_string

def main():
  print( vowels_only("banana") ) # "aaa"

main()
```

```
aaa
```

Write a function

1. Its name is `factorial`
2. It takes a numeric argument `number`
3. It returns the factorial of `number`
4. The factorial of a number is the product of itself and all the integers below it – factorial of 4 = `1 * 2 * 3 * 4 = 24`
5. Use `while`

```
print( factorial(4) ) # 24
print( factorial(5) ) # 120
print( factorial(0) ) # 1
```

Write a function – solution

Write more test cases for this function.

```
def factorial(number):
  result = 1
  index = 1
  while index <= number:
    result *= index
    index += 1
  return result

def main():
  assert factorial(4) == 24
  assert factorial(5) == 120
  assert factorial(0) == 1

main()
```

Submit code for attendance

Submit your `factorial` function to Gradescope for attendance.

Name your file `factorial.py`

Write a function

1. Its name is `power`
2. It takes two numeric arguments: `base` and `exp`
3. It returns the `base` to the power of `exp`
4. Don't use the `**` operator, use a `while` loop (define an `index` before the loop, use `index` in the `while` condition, change `index` inside the loop)

```
assert power(2, 3) == 8
assert power(3, 3) == 27
```

Write a function – solution

Write more test cases for this function.

```python
def power(base, exp):
  result = 1
  index = 1
  while index <= exp:
    result *= base
    index += 1
  return result

def main():
  print( power(2, 3) ) # 8
  print( power(3, 3) ) # 27

main()
```

```
8
27
```