Computational Physics Weekly Assessments
Week 2: Numerical Integration

Weekly assessment task & hints

1. Create a module, named '`cp_2.py`'

2. Implement a function 'f(x)' that returns x^2 sin(x). This is the function to be integrated

3. Implement a function 'g(x)' that is the indefinite integral of f(x).

- This function is *not* meant to analytically integrate any expression, the idea is that you find the indefinite integral from part 2 *on paper* and then type that in as a function.

3. Create a function, '`integrate_numeric(x0, x1, n_panels)`' that calculates the definite integral of the function `f(x)` over the interval [x0,x1] using Simpson's rule by splitting it up into n_panels.

- Every time you need to evaluate the function being integrated, call the function f(x).

4. Create a function, '`integrate_analytic(x0, x1)`' that uses an analytical formula to integrate the function over the same interval.

- You may wish to use a CAS such as Wolfram Alpha to generate the analytical expression.

5. Plot a graph of the percentage error in the numerical method vs the number of panels used.

- Use a log/log plot – see the example code below.
- To determine the error, look at the fractional between the analytical and numerical values as a percentage. Use the absolute of this (we don't care about the sign, just the magnitude.)
- If the numerical value of the integral is `area_n` and the analytical value is `area_a` then the error we wish to plot is `error = abs(area_a-area_n)/area_a`

6. Answer a question on numerical differentiation

- The question is "What effect(s) does changing the number of panels used have on the accuracy off the numerical method, and why?"
- Answer the question in no more than 60 words. Place the answer as a string variable in the global scope off your code (i.e. not in a function). Call the variable ANSWER1

6. Answer another question on numerical differentiation

- The question is "If the trapezium rule was being used, how would the panel count affect accuracy? Briefly explain your reasoning."
- Answer the question in no more than 60 words. Place the answer as a string variable in the global scope off your code (i.e. not in a function). Call the variable ANSWER2

General comments:

- Place the code for parts (5) and (6) above in the global scope
- Place two variables at the start of the code,
  - USER=”your name”
  - USER_ID = “your CIS login”
- Your module must run in order to be awarded any marks
- Your solution should contain no more than about 35 lines of code and perhaps 10 lines of comments. Excessively long submissions will be penalised.
- A partial example is shown below, using the function $f(x) = x^2$. Obviously you need to change this to match part (2) above and actually implement the area calculation for each panel.

compphys_assessment_2.py

compphys_assessment_2.py 〉 No Selection

```
from __future__ import division

import numpy
import matplotlib.pyplot as pyplot

USER     = "David Levenson"
USER_ID  = "eyed4x"

def f(x):
    return x**2

def compute_numeric_integral(x0, x1, n_panels):
    panel_width = (x1-x0) / n_panels
    area = 0

    for ix in range(n_panels):
        # Find the left edge of this panel
        a = x0 + ix * panel_width
        # Do some maths
        area = area + ???
    return area


def compute_analytic_integral(x0, x1):
    y0 = x0**3/3
    y1 = x1**3/3
    return y1 - y0

# Range of panel sizes to be evaluated
PANEL_COUNTS = [4, 8, 16, 32, 64, 128, 256, 512, 1024]

# Bounds to integrate
X0, X1 = 0, 2

# Evaluate error for various panel counts
y_data = []
ref = compute_analytic_integral(X0, X1)

for n in PANEL_COUNTS:
    s = compute_numeric_integral(0, 2, n)
    error = abs((s-ref)/ref)
    y_data.append(error)

pyplot.figure(figsize=(6,6))
pyplot.loglog()
# Scatter plot of data points
pyplot.scatter(PANEL_COUNTS, y_data)
pyplot.show()

ANSWER1 = '''I've got a cold'''
```