import sympy as sym

from calcgen import tools, validations, images

from calcgen.tools import unique, Printer, Template, Problem, random

from itertools import count

import os

import sys

count = count(1)

def alt\_text(m, a, f\_at\_a):

 short\_text = f"The graph of the function f of x and its tangent line at the point x = {a}."

 long\_text = f"The function f of x is graphed in the x y-coordinate plane along with its " \

 f"tangent line at the point ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)}). " \

 f"The point ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)})" \

 f" is marked on the graph. The tangent line is shown as a straight line passing through the point" \

 f" ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)}) with a slope of {tools.num\_to\_words(m)}."

 return f"{short\_text}|{long\_text}"

def alt\_text2(m, m2, a, f\_at\_a):

 short\_text = f"The graph of the function f of x, its tangent line, and the line normal " \

 f"to the tangent at the point x = {a}."

 long\_text = f"The function f of x is graphed in the x y-coordinate plane along with its " \

 f"tangent line at the point ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)}). " \

 f"The point ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)})" \

 f" is marked on the graph. The tangent line is shown as a straight line passing through the point" \

 f" ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)}) with a slope of {tools.num\_to\_words(m)}. " \

 f"The line normal to the tangent is shown as a straight line passing through the point" \

 f" ({tools.num\_to\_words(a)}, {tools.num\_to\_words(f\_at\_a)}) with a slope of {tools.num\_to\_words(m2)}."

 return f"{short\_text}|{long\_text}"

class Template1(Template):

 # f is a monomial like ax^n, x is a point. a can be a reasonable fraction or an integer (non-zero),

 # n can be a positive integer > 1 and < 5. Your answer should be less than 100 or so, so choose x wisely.

 @unique

 def variables(self):

 # Define any variables you need here. The @unique decorator will ensure your variables are unique

 a = sym.sympify(tools.non\_zero\_select(-2, 2))

 b = tools.non\_zero\_select(-8, 8)

 p = tools.non\_zero\_select(-4, 4)

 n = random.randint(2, 5)

 # You can add any additional constraints on your variables as assertions

 # limit location of point

 assert b != 1

 assert n \* a / b != 1

 assert a / b != int(a / b)

 assert -5 <= a / b \* p \*\* n < 5

 return a, b, p, n

 def template(self):

 x = sym.symbols('x')

 a, b, p, n = self.variables()

 f = (a / b) \* x \*\* n

 f\_disp = tools.polytex(f)

 fxp = f.subs(x, p)

 f\_diff = sym.diff(f, x)

 a\_diff = n \* (a / b)

 slope = f\_diff.subs(x, p)

 # get signs right for equations of line in slope-intercept and point-slope forms

 if fxp < 0:

 y2 = f"+{sym.latex(-fxp)}"

 else:

 y2 = f"-{sym.latex(fxp)}"

 if p < 0:

 x2 = f"+{-p}"

 else:

 x2 = f"-{p}"

 y\_int = -slope \* p + fxp

 if y\_int < 0:

 bb = f"{sym.latex(y\_int)}"

 else:

 bb = f"+{sym.latex(y\_int)}"

 g = slope \* x + y\_int

 g\_disp = tools.polytex(g)

 image\_name = f"find\_the\_equation\_of\_a\_tangent\_line\_using\_basic\_derivative\_rules\_{next(count)}"

 graph = images.Graph(name=image\_name)

 graph.set\_axis(style="standard", x\_step=1, x\_min=-6, x\_max=6, x\_label\_step=1, y\_min=-6, y\_max=6)

 graph.addplot("restrict y to domain = -10:10", function=f, samples=901, domain=[-10, 10])

 graph.addplot(function=g, samples=101, domain=[-10, 10], color="red")

 graph.addplot("mark size=2pt", coordinates=[(p, fxp)], color="black", mark="\*")

 # graph.addplot(coordinates=[(p, fxp - 1)], mark="none", color="black",

 # label=f"$\\large({p},{fxp})$", label\_position=["below", "pos=.5", "fill=white"])

 # graph.addplot(coordinates=[(p, fxp + 1)], mark="none", color="black",

 # label=f"$\\large y={g\_disp}$", label\_position=["above", "pos=.5", "fill=white"])

 graph.print\_graph()

 metadata1 = "CC\_BY\_NC\_ND|Knewton|http://www.knewton.com|" + alt\_text(slope, p, fxp)

 question\_stem = f"Find the equation of the line tangent to the graph of $\_f(x) =" \

 f" \\displaystyle{f\_disp}$\_ at $\_x = {p}$\_. <p>Please submit your answer in slope-intercept form.</p>"

 explanation = f"<p>To find the equation of the line tangent to the graph of $\_f(x)$\_ " \

 f"we will need to recall: " \

 f"<ul><li>The <b>extended power rule</b> for derivatives, which states " \

 f"that for a function $\_f(x) = x^{{n}}$\_, given $\_n$\_ is a non-zero integer, then " \

 f"$$f^{{\\prime}}(x) = nx^{{n-1}}.$$</li>" \

 f"<li>The <b>equation of the tangent line</b> to a function, $\_f(x)$\_, at the point " \

 "$\_x = a$\_ is given by $\_y = mx+b$\_ where $\_m$\_ is the value of the derivative of " \

 f"$\_f$\_ evaluated at $\_a$\_. In other words" \

 "$$m\_{\\text{tan}} = f'(a).$$</li>" \

 f"</ul></p>" \

 f"<p>To find the equation of a line, we need a point on the line and the slope of the line. To find a " \

 f"point on the line, we will evaluate the function $\_f(x)$\_ at $\_x = {p}$\_. To find the " \

 f"slope of the line at $\_x = {p}$\_, we will find $\_f^{{\\prime}}({p})$\_.</p>" \

 f"<p>Evaluating the function $\_f(x)$\_ at $\_x = {p}$\_ gives " \

 f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f\\color{{red}}{{({p})}}&={sym.latex(a / b)}\\color{{red}}{{({p})}}^{n}\\\\[5pt]" \

 f"&={sym.latex(fxp)}." \

 f"\\end{{align}}$$</p>" \

 f"<p>This gives us the point $\_\\left({p},{sym.latex(fxp)}\\right)$\_. " \

 f"Now we need to define the slope of the tangent line at our point. " \

 f"We will first define the derivative of $\_f(x)$\_, then evaluate the derivative at $\_x={p}$\_ to find the slope."

 if n - 1 != 1:

 explanation += f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f^{{\\prime}}(x)&={sym.latex(a / b)}\\left({n}x^{{ {n} - 1 }}\\right)\\\\[5pt]" \

 f"&={tools.polytex(f\_diff)}\\\\[5pt]" \

 f"f^{{\\prime}}\\color{{red}}{{({p})}}&={sym.latex(a\_diff)}\\color{{red}}{{({p})}}^{{{n-1}}}\\\\[5pt]" \

 f"&={sym.latex(slope)}" \

 f"\\end{{align}}$$</p>"

 else:

 explanation += f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f^{{\\prime}}(x)&={sym.latex(a / b)}\\left({n}x^{{ {n} - 1 }}\\right)\\\\[5pt]" \

 f"&={tools.polytex(f\_diff)}\\\\[5pt]" \

 f"f^{{\\prime}}\\color{{red}}{{({p})}}&={sym.latex(a\_diff)}\\color{{red}}{{({p})}}\\\\[5pt]" \

 f"&={sym.latex(slope)}" \

 f"\\end{{align}}$$</p>"

 if slope == 1:

 explanation += f"<p>Using the point-slope formula, we can define the equation of the line tangent to" \

 f" $\_f(x)$\_ at $\_x = {p}$\_ as " \

 f"$$y{y2}=(x{x2}).$$</p>" \

 f"<p>Putting the equation of the line in slope-intercept form, we obtain " \

 f"$$y=x{bb}.$$</p>" \

 f"${{images/{image\_name}.svg|{metadata1}}}$"

 else:

 explanation += f"<p>Using the point-slope formula, we can define the equation of the line tangent to" \

 f" $\_f(x)$\_ at $\_x = {p}$\_ as " \

 f"$$y{y2}={sym.latex(slope)}(x{x2}).$$</p>" \

 f"<p>Putting the equation of the line in slope-intercept form, we obtain " \

 f"$$y={sym.latex(slope)}x{bb}.$$</p>" \

 f"${{images/{image\_name}.svg|{metadata1}}}$"

 correct\_answer = f"y={sym.latex(g)}"

 question\_template = "y={{response}}"

 json\_blob = validations.lea\_blob(template=question\_template,

 response=correct\_answer,

 validation="equivSymbolic")

 return Problem(concepts="Find the equation of a line tangent to a monomial function",

 question\_stem=question\_stem,

 explanation=explanation,

 correct\_answer=correct\_answer,

 json\_blob=json\_blob)

class Template2(Template):

 # f = ax^n + bx^m + c, all integers. At least two of a, b, c should be non-zero. n,m <= 4.

 @unique

 def variables(self):

 # Define any variables you need here. The @unique decorator will ensure your variables are unique

 a = tools.non\_zero\_select(-2, 2)

 b = tools.non\_zero\_select(-5, 5)

 c = random.randint(-5, 6)

 xx = tools.non\_zero\_select(-3, 3)

 n, m = sorted(random.choice(list(range(1, 4)), 2, replace=False))

 # You can add any additional constraints on your variables as assertions

 assert abs(a \* xx \*\* m + b \* xx \*\* n + c) < 5

 assert a != 1

 assert b != 1

 return a, b, c, xx, n, m

 def template(self):

 x, y = sym.symbols('x y')

 a, b, c, xx, n, m = self.variables()

 f = a \* x \*\* m + b \* x \*\* n + c

 df = sym.diff(f)

 f\_at\_xx = f.subs(x, xx)

 f\_string = tools.add\_terms(a \* x \*\* m, b \* x \*\* n, c)

 df\_string = tools.add\_terms(a \* m \* x \*\* (m - 1), b \* n \* x \*\* (n - 1))

 slope = df.subs(x, xx)

 ans = tools.polytex(sym.expand(slope \* (x - xx) + f.subs(x, xx)))

 g = slope \* (x - xx) + f.subs(x, xx)

 image\_name = f"find\_the\_equation\_of\_a\_tangent\_line\_using\_basic\_derivative\_rules\_{next(count)}"

 metadata2 = "CC\_BY\_NC\_ND|Knewton|http://www.knewton.com|" + alt\_text(slope, xx, f\_at\_xx)

 graph = images.Graph(name=image\_name)

 graph.set\_axis(style="standard", x\_step=1, x\_min=-6, x\_max=6, x\_label\_step=1, y\_min=-6, y\_max=6)

 graph.addplot("restrict y to domain = -10:10", function=f, samples=901, domain=[-10, 10])

 graph.addplot(function=g, samples=101, domain=[-10, 10], color="red")

 graph.addplot("mark size=2pt", coordinates=[(xx, f\_at\_xx)], color="black", mark="\*")

 # graph.addplot(coordinates=[(xx, f\_at\_xx - 1)], mark="none", color="black",

 # label=f"$\\large({xx},{f\_at\_xx})$",

 # label\_position=["below", "pos=.5", "fill=white", "font=\\large"])

 # graph.addplot(coordinates=[(xx, f\_at\_xx + 1)], mark="none", color="black",

 # label=f"$\\large y={ans}$", label\_position=["above", "pos=.5", "fill=white", "font=\\large"])

 graph.print\_graph()

 question\_stem = f"Find the equation of the line tangent to the graph of $\_f(x)={f\_string}$\_ at $\_x = {xx}$\_."

 explanation = "<p>To find the equation of the tangent line to the function at the given point, " \

 "we will need to recall:" \

 "<ul>" \

 "<li>The <b>equation of the tangent line</b> to a function, $\_f(x)$\_, at the point " \

 "$\_x = a$\_ is given by $\_y = mx+b$\_ where $\_m$\_ is the value of the derivative of " \

 f"$\_f$\_ evaluated at $\_a$\_. In other words" \

 "$$m\_{\\text{tan}} = f'(a).$$</li>" \

 "<li>The <b>sum and difference rules</b> for derivatives state that the derivative of the " \

 "sum or difference of two functions is equal to the sum or difference of their derivative. " \

 "In math notation this means that " \

 "$$\\frac{{d}}{{dx}}\\left(f(x) \\pm g(x) \\right)" \

 " = \\frac{{d}}{{dx}}f(x) \\pm \\frac{{d}}{{dx}}g(x).$$</li>" \

 "<li>The <b>extended power rule</b> for derivatives states that if " \

 "$\_n$\_ is a non-zero real number, then " \

 "$$\\frac{{d}}{{dx}}\\left(x^n \\right) = nx^{n-1}.$$</li>" \

 "<li>The <b>constant multiple rule</b> for derivatives states that the derivative of a " \

 "constant times a function is equal to the constant times the derivative of the function. " \

 "In math notation this means if $\_c$\_ is a constant then " \

 "$$\\frac{{d}}{{dx}}\\left(cf(x)\\right) = c \\frac{{d}}{{dx}}f(x).$$</li>" \

 "</ul>" \

 "</p>"

 df\_dx\_expanded = tools.operator\_expand\_string('\\frac{{d}}{{dx}}', a \* x \*\* m, b \* x \*\* n, c,

 include\_zeros=False)

 constants\_pulled\_out = tools.operator\_expand\_string('\\frac{{d}}{{dx}}', a \* x \*\* m, b \* x \*\* n,

 include\_zeros=False,

 pull\_out\_const=True)

 explanation += f"<p>To define the derivative of $\_f(x)$\_, we first apply the sum and difference " \

 f"rules to $\_f(x)$\_ to give " \

 f"$$\\begin{{align}}" \

 f"f(x) &= {f\_string} \\\\[5pt]" \

 f"\\frac{{d}}{{dx}}f(x) &= \\frac{{d}}{{dx}}\\left({f\_string}\\right) \\\\[5pt]" \

 f" &= {df\_dx\_expanded}. " \

 f"\\end{{align}}$$</p>"

 extra\_bit\_of\_text = "" if c == 0 else "and use the fact that the derivative of a constant is zero, "

 explanation += f"<p>Next, we apply the constant rule to take the constants outside of the derivatives" \

 f" {extra\_bit\_of\_text} to give " \

 f"$$\\frac{{d}}{{dx}}f(x) = {constants\_pulled\_out}" \

 f"{tools.pmsign(c / abs(c)) + '0' if c != 0 else ''}.$$</p>"

 explanation += f"<p>The last step in finding the derivative of $\_f(x)$\_ is to use the power rule to find" \

 f"$$\\begin{{align}}" \

 f"\\frac{{d}}{{dx}}f(x) &= {tools.pmsign(a, leading=True)} " \

 f"\\left({sym.latex(m)}x^{{{m}-1}}\\right) " \

 f"{tools.pmsign(b)}\\left({sym.latex(n)}x^{{{n}-1}}\\right) \\\\[5pt]" \

 f"&= {df\_string}." \

 f"\\end{{align}}$$</p>"

 explanation += "<p>To find the slope of the tangent line to $\_f(x)$\_ at the point " \

 f"$\_x = {xx}$\_, we evaluate the formula for $\_f'({xx})$\_ and " \

 f"simplify." \

 f"$$\\begin{{align}}" \

 f"f'(x) &= {df\_string} \\\\[5pt]" \

 f"f'\\color{{red}}{{({xx})}} &= {tools.substitute(xx, df\_string, color='red')} \\\\[5pt]" \

 f"&= {tools.terms\_string(a \* m \* xx \*\* (m - 1), b \* n \* xx \*\* (n - 1))} \\\\[5pt]" \

 f"&= {sym.latex(df.subs(x, xx))}" \

 f"\\end{{align}}$$</p>"

 explanation += f"<p>We can find a point on the tangent line by substituting the given value of $\_x$\_ into " \

 f"$\_f$\_. " \

 f"$$\\begin{{align}}" \

 f"f\\color{{red}}{{({xx})}} &= {tools.substitute(xx, f, color='red')} \\\\[5pt]" \

 f"&= {sym.latex(f.subs(x, xx))}" \

 f"\\end{{align}}$$</p>" \

 f"<p>This gives us the point $\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_.</p>"

 explanation += f"<p>At this point, we know the slope of the tangent line is $\_{sym.latex(slope)}$\_ and " \

 f"that the point $\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_ is on the tangent line. " \

 f"This is what we need to use the point-slope form of a line " \

 f"$$y - y\_0 = m(x - x\_0).$$"

 explanation += f"<p>Using the point-slope formula with " \

 f"$$\\begin{{align}}" \

 f"m &= {sym.latex(slope)} \\\\[5pt] " \

 f"(x\_0, y\_0) &= \\left({xx}, {sym.latex(f.subs(x, xx))}\\right)," \

 f"\\end{{align}}$$</p>" \

 f"<p>we find that the equation of the tangent line is" \

 f"$$\\begin{{align}}" \

 f"{sym.latex(y - f.subs(x, xx))} " \

 f"&= {sym.latex(slope)}\\left({sym.latex(x - xx)}\\right) \\\\[5pt]" \

 f"y &= {ans}." \

 f"\\end{{align}}$$</p>" \

 f"${{images/{image\_name}.svg|{metadata2}}}$"

 correct\_answer = f"y = {sym.latex(ans)}"

 question\_template = "y = {{response}}"

 json\_blob = validations.lea\_blob(template=question\_template,

 response=correct\_answer,

 validation="equivSymbolic")

 return Problem(concepts="Find the equation of a line tangent to a polynomial function",

 question\_stem=question\_stem,

 explanation=explanation,

 correct\_answer=correct\_answer,

 json\_blob=json\_blob)

class Template3(Template):

 # f = ax^n + bx^m + c, all integers. At least two of a, b, c should be non-zero. n,m <= 4.

 @unique

 def variables(self):

 # Define any variables you need here. The @unique decorator will ensure your variables are unique

 a = tools.non\_zero\_select(-2, 2)

 b = tools.non\_zero\_select(-5, 5)

 c = random.randint(-6, 7)

 xx = tools.non\_zero\_select(-3, 3)

 n, m = sorted(random.choice(list(range(1, 4)), 2, replace=False))

 x = sym.symbols('x')

 f = a \* x \*\* m + b \* x \*\* n + c

 df = sym.diff(f)

 slope = df.subs(x, xx)

 intercept = -slope \* xx + a \* xx \*\* m + b \* xx \*\* n + c

 # You can add any additional constraints on your variables as assertions

 assert abs(a \* xx \*\* m + b \* xx \*\* n + c) < 7

 assert xx != a \* xx \*\* m + b \* xx \*\* n + c

 assert slope != 0

 assert abs(intercept) < 9

 return a, b, c, xx, n, m

 def desmos\_blob(self, f, xx, slope, intercept):

 x = sym.symbols('x')

 blob = r'''{

 "ui\_style": {

 "fontsize": "large"

 },

 "validation": {

 "valid\_response": {

 "score": 1,

 "value": true

 }

 },

 "type": "custom",

 "js": "https://questions-va.learnosity.com/latest/customassets/redirect?customType=desmos\_draggable-v2&assetType=js&clientReference=988e92a0",

 "custom\_type": "desmos\_blank\_graph\_question",

 "version": "0.0.1",

 "graph\_width": 500,

 "correct\_answer": {

 "variable\_values": {},

 "slider\_values": {

 "a": %s,

 "b": %s,

 "c": 0,

 "d": %s

 }

 },

 "success\_condition": "\\left\\{m=%s\\right\\}\\left\\{G=1\\right\\}",

 "initial\_state": {

 "version": 7,

 "graph": {

 "viewport": {

 "xmin": -10,

 "ymin": -10,

 "xmax": 10,

 "ymax": 10

 }

 },

 "expressions": {

 "list": [{

 "type": "expression",

 "id": "1",

 "color": "#2d70b3",

 "latex": "f\\left(x\\right)=%s"

 }, {

 "type": "folder",

 "id": "10",

 "title": "Tangent Line"

 }, {

 "type": "expression",

 "id": "8",

 "folderId": "10",

 "color": "#c74440",

 "latex": "y-d=m\\left(x-c\\right)"

 }, {

 "type": "expression",

 "id": "11",

 "folderId": "10",

 "color": "#c74440",

 "latex": "x=a\\left\\{c=a\\right\\}"

 }, {

 "type": "expression",

 "id": "2",

 "folderId": "10",

 "color": "#c74440",

 "latex": "\\left(a,b\\right)",

 "showLabel": true

 }, {

 "type": "expression",

 "id": "4",

 "folderId": "10",

 "color": "#c74440",

 "latex": "\\left(c,d\\right)",

 "showLabel": true

 }, {

 "type": "expression",

 "id": "3",

 "folderId": "10",

 "color": "#388c46",

 "latex": "a=0",

 "hidden": true,

 "slider": {

 "step": ".5"

 }

 }, {

 "type": "expression",

 "id": "17",

 "folderId": "10",

 "color": "#6042a6",

 "latex": "b=0",

 "hidden": true,

 "slider": {

 "step": ".5"

 }

 }, {

 "type": "expression",

 "id": "5",

 "folderId": "10",

 "color": "#000000",

 "latex": "c=1",

 "hidden": true,

 "slider": {

 "step": ".5"

 }

 }, {

 "type": "expression",

 "id": "6",

 "folderId": "10",

 "color": "#c74440",

 "latex": "d=1",

 "hidden": true,

 "slider": {

 "step": ".5"

 }

 }, {

 "type": "expression",

 "id": "7",

 "folderId": "10",

 "color": "#2d70b3",

 "latex": "m=\\frac{\\left(d-b\\right)}{c-a}",

 "displayEvaluationAsFraction": true

 }, {

 "type": "folder",

 "id": "12",

 "title": "Grade"

 }, {

 "type": "text",

 "id": "14",

 "folderId": "12",

 "text": "We can check that the tangent line is correct by just checking that m is correct

 and that the given point (x\_0, y\_0) is on the line."

 }, {

 "type": "expression",

 "id": "20",

 "folderId": "12",

 "color": "#388c46",

 "latex": "x\_0=%s"

 }, {

 "type": "expression",

 "id": "21",

 "folderId": "12",

 "color": "#6042a6",

 "latex": "y\_0=%s"

 }, {

 "type": "expression",

 "id": "19",

 "folderId": "12",

 "color": "#2d70b3",

 "latex": "G=\\left\\{y\_0-d=m\\left(x\_0-c\\right):1,0\\right\\}"

 }]

 }

 },

 "graph\_height": 500,

 "graph\_url": ""

 }''' % (sym.latex(xx), sym.latex(f.subs(x, xx)), sym.latex(intercept),

 sym.latex(slope), tools.polytex(f), sym.latex(xx), sym.latex(f.subs(x, xx)))

 return blob.replace("\n", "")

 def template(self):

 x, y = sym.symbols('x y')

 a, b, c, xx, n, m = self.variables()

 f = a \* x \*\* m + b \* x \*\* n + c

 df = sym.diff(f)

 f\_string = tools.terms\_string(a \* x \*\* m, b \* x \*\* n, c)

 df\_string = tools.terms\_string(a \* m \* x \*\* (m - 1), b \* n \* x \*\* (n - 1))

 question\_stem = f"The graph of $\_f(x) = {sym.latex(f)}$\_ is shown below. " \

 f"Use the tools to graph the line tangent to $\_f(x)$\_ at $\_x = {xx}$\_."

 explanation = "<p>To find the equation of the tangent line to the function at the given point, " \

 "we will need to recall:" \

 "<ul>" \

 "<li>The <b>equation of the tangent line</b> to a function, $\_f(x)$\_, at the point " \

 "$\_x = a$\_ is given by $\_y = mx+b$\_ where $\_m$\_ is the value of the derivative of " \

 f"$\_f$\_ evaluated at $\_a$\_. In other words" \

 "$$m\_{\\text{tan}} = f'(a).$$</li>" \

 "<li>The <b>sum and difference rules</b> for derivatives state that the derivative of the " \

 "sum or difference of two functions is equal to the sum or difference of their derivative. " \

 "In math notation this means that " \

 "$$\\frac{{d}}{{dx}}\\left(f(x) \\pm g(x) \\right)" \

 " = \\frac{{d}}{{dx}}f(x) \\pm \\frac{{d}}{{dx}}g(x).$$</li>" \

 "<li>The <b>extended power rule</b> for derivatives states that if " \

 "$\_n$\_ is a non-zero real number, then " \

 "$$\\frac{{d}}{{dx}}\\left(x^n \\right) = nx^{n-1}.$$</li>" \

 "<li>The <b>constant multiple rule</b> for derivatives states that the derivative of a " \

 "constant times a function is equal to the constant times the derivative of the function. " \

 "In math notation this means if $\_c$\_ is a constant then " \

 "$$\\frac{{d}}{{dx}}\\left(cf(x)\\right) = c \\frac{{d}}{{dx}}f(x).$$</li>" \

 "</ul>" \

 "</p>"

 df\_dx\_expanded = tools.operator\_expand\_string('\\frac{{d}}{{dx}}', a \* x \*\* m, b \* x \*\* n, c,

 include\_zeros=False)

 constants\_pulled\_out = tools.operator\_expand\_string('\\frac{{d}}{{dx}}', a \* x \*\* m, b \* x \*\* n,

 include\_zeros=False,

 pull\_out\_const=True)

 explanation += f"<p>To define the derivative of $\_f(x)$\_, we first apply the sum and difference " \

 f"rules to $\_f(x)$\_ to give " \

 f"$$\\begin{{align}}" \

 f"f(x) &= {f\_string} \\\\[5pt]" \

 f"\\frac{{d}}{{dx}}f(x) &= {df\_dx\_expanded}" \

 f"\\end{{align}}$$</p>"

 extra\_bit\_of\_text = "" if c == 0 else "and use the fact that the derivative of a constant is zero, "

 explanation += f"<p>Next, we apply the constant rule to take the constants outside of the derivatives" \

 f" {extra\_bit\_of\_text} to find " \

 f"$$\\frac{{d}}{{dx}}f(x) = {constants\_pulled\_out}" \

 f"{tools.pmsign(c / abs(c)) + '0' if c != 0 else ''}.$$</p>"

 explanation += f"<p>The last step in finding the derivative of $\_f(x)$\_is to use the power rule to find" \

 f"$$\\begin{{align}}" \

 f"\\frac{{d}}{{dx}}f(x) &= {tools.pmsign(a, leading=True)} " \

 f"\\left({sym.latex(m)}x^{{{m}-1}}\\right) " \

 f"{tools.pmsign(b)}\\left({sym.latex(n)}x^{{{n}-1}}\\right) \\\\[5pt]" \

 f"&= {df\_string}." \

 f"\\end{{align}}$$</p>"

 slope = df.subs(x, xx) # the slope of the tangent line

 explanation += f"<p>To find the slope of the tangent line to $\_f(x)$\_ at the point " \

 f"$\_x = {xx}$\_, we evaluate the formula for $\_f'(x)$\_ at $\_x = \\color{{red}}{{{xx}}}$\_ and " \

 f"simplify." \

 f"$$\\begin{{align}}" \

 f"f'(x) &= {df\_string} \\\\[5pt]" \

 f"f'\\color{{red}}{{({xx})}} &= {tools.substitute(xx, df\_string, color='red')} \\\\[5pt]" \

 f"&= {tools.terms\_string(a \* m \* xx \*\* (m - 1), b \* n \* xx \*\* (n - 1))} \\\\[5pt]" \

 f"&= {sym.latex(df.subs(x, xx))}" \

 f"\\end{{align}}$$</p>"

 ans = tools.polytex(sym.expand(slope \* (x - xx) + f.subs(x, xx)))

 explanation += f"<p>We can find a point on the tangent line by substituting the given value of $\_x$\_ into " \

 f"$\_f$\_. " \

 f"$$\\begin{{align}}" \

 f"f\\color{{red}}{{({xx})}} &= {tools.substitute(xx, f, color='red')} \\\\[5pt]" \

 f"&= {sym.latex(f.subs(x, xx))}" \

 f"\\end{{align}}$$</p>" \

 f"<p>This gives us the point $\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_.</p>"

 explanation += f"<p>At this point, we know the slope of the tangent line is $\_{sym.latex(slope)}$\_ and " \

 f"that the point $\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_ is on the tangent line. " \

 f"This is what we need to use the point-slope form of a line " \

 f"$$y - y\_0 = m(x - x\_0).$$"

 explanation += f"<p>Using the point-slope formula with " \

 f"$$\\begin{{align}}" \

 f"m &= {sym.latex(slope)} \\\\[5pt] " \

 f"(x\_0, y\_0) &= \\left({xx}, {sym.latex(f.subs(x, xx))}\\right), " \

 f"\\end{{align}}$$</p>" \

 f"<p>we find that the equation of the tangent line is" \

 f"$$\\begin{{align}}" \

 f"{sym.latex(y - f.subs(x, xx))} " \

 f"&= {sym.latex(slope)}\\left({sym.latex(x - xx)}\\right) \\\\[5pt]" \

 f"y &= {ans}." \

 f"\\end{{align}}$$</p>"

 intercept = sym.latex(-slope \* xx + a \* xx \*\* m + b \* xx \*\* n + c)

 explanation += f"<p>Finally, to plot the line tangent to $\_f(x)$\_ at " \

 f"$\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_ move one of the points to " \

 f"$\_\\left({xx}, {sym.latex(f.subs(x, xx))}\\right)$\_ and the other to the $\_y$\_-intercept " \

 f"$\_(0, {intercept})$\_."

 json\_blob = self.desmos\_blob(f, xx, slope, -slope \* xx + f.subs(x, xx))

 return Problem(concepts="Graph a line tangent to a polynomial function",

 question\_stem=question\_stem,

 explanation=explanation,

 correct\_answer="",

 json\_blob=json\_blob)

class Template4(Template):

 # f = ax^n + bx^m + c, all integers. At least two of a, b, c should be non-zero. n,m <= 4. Keep x

 # reasonable so your final answer is < 100 or so

 @unique

 def variables(self):

 # Define any variables you need here. The @unique decorator will ensure your variables are unique

 a = tools.non\_zero\_select(-2, 2)

 b = tools.non\_zero\_select(-5, 5)

 c = random.randint(-5, 6)

 xx = random.randint(-3, 3)

 n, m = sorted(random.choice(list(range(1, 4)), 2, replace=False))

 assert abs(a \* xx \*\* m + b \* xx \*\* n + c) < 5

 assert abs(b) != 1

 return a, b, c, xx, n, m

 def template(self):

 x = sym.symbols('x')

 a, b, c, xx, n, m = self.variables()

 f = a \* x \*\* m + b \* x \*\* n + c

 f\_disp = tools.polytex(f)

 fxp = f.subs(x, xx)

 f\_at\_xx = tools.substitute\_unsimplified(xx, f\_disp, x=sym.symbols('x'), include\_parentheses=True, order='lex',

 color='red')

 f\_diff = sym.diff(f, x)

 f\_diff\_disp = tools.polytex(f\_diff)

 f\_diff\_subs = tools.substitute\_unsimplified(xx, f\_diff\_disp, x=sym.symbols('x'), include\_parentheses=True,

 order='lex',

 color='red')

 slope = f\_diff.subs(x, xx)

 y\_int = -(xx \* slope) + fxp

 g = slope \* x + y\_int

 image\_name = f"find\_the\_equation\_of\_a\_tangent\_line\_using\_basic\_derivative\_rules\_{next(count)}"

 metadata4 = "CC\_BY\_NC\_ND|Knewton|http://www.knewton.com|" + alt\_text(slope, xx, fxp)

 graph = images.Graph(name=image\_name)

 graph.set\_axis(style="standard", x\_step=1, x\_min=-6, x\_max=6, x\_label\_step=1, y\_min=-6, y\_max=6)

 graph.addplot("restrict y to domain = -10:10", function=f, samples=901, domain=[-10, 10])

 graph.addplot(function=g, samples=101, domain=[-10, 10], color="red")

 graph.addplot("mark size=2pt", coordinates=[(xx, fxp)], color="black", mark="\*")

 # graph.addplot(coordinates=[(xx, fxp - 1)], mark="none", color="black",

 # label=f"$\\large({xx},{fxp})$", label\_position=["below", "pos=.5", "fill=white", "font=\\large"])

 # graph.addplot(coordinates=[(xx, fxp + 1)], mark="none", color="black",

 # label=f"$y={sym.latex(g)}$", label\_position=["above", "pos=.5", "fill=white", "font=\\large"])

 graph.print\_graph()

 question\_stem = f"Find slope of the line tangent to the graph of $\_f(x) = {f\_disp}$\_ at $\_x = {xx}$\_."

 extra\_bit\_of\_text = "" if c == 0 else "and using the fact that the derivative of a constant is zero "

 first\_term = f"{tools.pmsign(a, leading=True)}\\left({sym.latex(m \* x)}^{{{m}-1}}\\right)"

 middle\_term = f"{tools.pmsign(b)}" \

 f"\\left({sym.latex(n \* x)}^{{{n}-1}}\\right)"

 last\_term = f"+0" if c != 0 else ""

 explanation = f"<p>To find the slope of the line tangent to the graph of $\_f(x) = {f\_disp}$\_ at $\_x = {xx}$\_, " \

 f"we will need to recall" \

 "<ul>" \

 "<li>The <b>sum and difference rules</b> for derivatives state that the derivative of the " \

 "sum or difference of two functions is equal to the sum or difference of their derivative. " \

 "In math notation this means that " \

 "$$\\frac{{d}}{{dx}}\\left(f(x) \\pm g(x) \\right) = \\frac{{d}}{{dx}}f(x)" \

 f"\\pm \\frac{{d}}{{dx}}g(x).$$</li>" \

 "<li>The <b>extended power rule</b> for derivatives states that if " \

 "$\_n$\_ is a non-zero real number, then " \

 "$$\\frac{{d}}{{dx}}\\left(x^n \\right) = nx^{n-1}.$$</li></ul>" \

 "</p>" \

 f"<p>To find the slope of the line tangent at $\_x = {xx}$\_, we will find " \

 f"$\_f^{{\\prime}}({xx})$\_.</p>" \

 f"<p>Evaluating the function $\_f(x) = {f\_disp}$\_ at $\_x = {xx}$\_ gives " \

 f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f\\color{{red}}{{({xx})}}&={f\_at\_xx}\\\\[5pt]" \

 f"&={sym.latex(fxp)}." \

 f"\\end{{align}}$$</p>" \

 f"<p>This gives us the point $\_\\left({xx},{sym.latex(fxp)}\\right)$\_. " \

 f"<p>Applying the sum and difference rules with the extended " \

 f"power rule {extra\_bit\_of\_text} gives " \

 f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f^{{\\prime}}(x)&={first\_term}{middle\_term}{last\_term}\\\\[5pt]" \

 f"&={sym.latex(f\_diff)}.\\\\[5pt]" \

 f"\\end{{align}}$$</p>" \

 f"<p>Evaluating the derivative of $\_f(x)$\_ at $\_x={xx}$\_ gives " \

 f"$$\\begin{{align}}" \

 f"f^{{\\prime}}(x)&={sym.latex(f\_diff)}\\\\[5pt]" \

 f"f^{{\\prime}}\\color{{red}}{{({xx})}} &= {f\_diff\_subs}\\\\[5pt]" \

 f"&={slope}.\\\\[5pt]" \

 f"\\end{{align}}$$</p>" \

 f"<p>We have found that the slope of the line tangent to the graph of" \

 f" $\_f(x) = {f\_disp}$\_ at $\_x = {xx}$\_ is $\_{slope}$\_.</p>" \

 f"${{images/{image\_name}.svg|{metadata4}}}$"

 correct\_answer = f"slope = {slope}"

 question\_template = "slope = {{response}}"

 json\_blob = validations.lea\_blob(template=question\_template,

 response=correct\_answer,

 validation="equivSymbolic")

 return Problem(concepts="Find the slope of the line tangent to a polynomial function",

 question\_stem=question\_stem,

 explanation=explanation,

 correct\_answer=correct\_answer,

 json\_blob=json\_blob)

class Template5(Template):

 # f = ax^n + bx^m + c, all integers. At least two of a, b, c should be non-zero. n,m <= 4.

 @unique

 def variables(self):

 # Define any variables you need here. The @unique decorator will ensure your variables are unique

 a = tools.non\_zero\_select(-2, 2)

 b = tools.non\_zero\_select(-5, 5)

 c = random.randint(-5, 6)

 p = tools.non\_zero\_select(-3, 3)

 n, m = sorted(random.choice(list(range(1, 4)), 2, replace=False))

 # You can add any additional constraints on your variables as assertions

 assert abs(a \* p \*\* m + b \* p \*\* n + c) < 5

 assert abs(b) != 1

 return a, b, c, p, n, m

 def template(self):

 x = sym.symbols('x')

 y = sym.symbols('y')

 a, b, c, p, n, m = self.variables()

 f = a \* x \*\* m + b \* x \*\* n + c

 f\_disp = tools.polytex(f)

 fxp = f.subs(x, p)

 f\_subs\_p = tools.substitute\_unsimplified(p, f\_disp, x=sym.symbols('x'), include\_parentheses=True, order='lex',

 color='red')

 f\_diff = sym.diff(f, x)

 f\_diff\_disp = tools.polytex(f\_diff)

 f\_diff\_subs = tools.substitute\_unsimplified(p, f\_diff\_disp, x=sym.symbols('x'),

 include\_parentheses=True, order='lex', color='red')

 slope = f\_diff.subs(x, p)

 y\_int = -(p \* slope) + fxp

 g = slope \* x + y\_int

 extra\_bit\_of\_text = "" if c == 0 else "and using the fact that the derivative of a constant is zero "

 first\_term = f"{tools.pmsign(a, leading=True)}\\left({sym.latex(m \* x)}^{{{m}-1}}\\right)"

 middle\_term = f"{tools.pmsign(b)}" \

 f"\\left({sym.latex(n \* x)}^{{{n}-1}}\\right)"

 last\_term = f"+0" if c != 0 else ""

 if slope == 1:

 normal\_slope = -1

 point\_slope\_form = f"{sym.latex(y-fxp)}=({sym.latex(x-p)})"

 h = normal\_slope \* x + (p / slope + fxp)

 point\_slope\_form\_normal = f"{sym.latex(y-fxp)}=-({sym.latex(x-p)})"

 elif slope == -1:

 normal\_slope = 1

 point\_slope\_form = f"y{tools.pmsign(fxp)}=-({sym.latex(x-p)})"

 point\_slope\_form\_normal = f"{sym.latex(y - fxp)}=({sym.latex(x - p)})"

 h = normal\_slope \* x + (p / slope + fxp)

 elif slope == 0:

 normal\_slope = f"\\frac{{-1}}{{0}}"

 point\_slope\_form = f"{sym.latex(y - fxp)}=0({sym.latex(x - p)})"

 else:

 normal\_slope = sym.Rational(-1, slope)

 point\_slope\_form = f"y{tools.pmsign(fxp)}={sym.latex(slope)}({sym.latex(x-p)})"

 point\_slope\_form\_normal = f"{sym.latex(y - fxp)}={sym.latex(normal\_slope)}({sym.latex(x-p)})"

 h = normal\_slope \* x + (p / slope + fxp)

 image\_name = f"find\_the\_equation\_of\_a\_tangent\_line\_using\_basic\_derivative\_rules\_{next(count)}"

 metadata5 = "CC\_BY\_NC\_ND|Knewton|http://www.knewton.com|" + alt\_text2(slope, normal\_slope, p, fxp)

 graph = images.Graph(name=image\_name)

 graph.set\_axis(style="standard", x\_step=1, x\_min=-6, x\_max=6, x\_label\_step=1, y\_min=-6, y\_max=6)

 graph.addplot("restrict y to domain = -10:10", function=f, samples=901, domain=[-10, 10])

 graph.addplot(function=g, samples=101, domain=[-10, 10], color="red", \_dashed="dashed")

 if slope != 0:

 graph.addplot(function=h, samples=101, domain=[-10, 10], color="red")

 else:

 graph.addplot(coordinates=[(p, -6), (p, 6)], color="red")

 graph.addplot("mark size=2pt", coordinates=[(p, fxp)], color="black", mark="\*")

 # if slope == 0:

 # coords = [(p, fxp - 2)]

 # line\_label = f"$x={p}$"

 # graph.addplot(coordinates=[(p, fxp), (p, 10)], color="red")

 # graph.addplot(coordinates=[(p, fxp), (p, -10)], color="red")

 #

 # else:

 # coords = [(p, h.subs(x, p) - 2)]

 # line\_label = f"$y={h}$"

 # graph.addplot(function=h, samples=101, domain=[-10, 10], color="red")

 # graph.addplot(coordinates=coords, color="red", mark="", label=line\_label,

 # label\_position=["right", "pos=.5", "fill=white", "font=\\large"])

 # graph.addplot(coordinates=[(p, fxp)], color="black", mark="\*", label=f"$({p},{fxp})$",

 # label\_position=["right", "pos=.5", "fill=white", "font=\\large"])

 graph.print\_graph()

 question\_stem = f"Find the equation of the line normal to the graph of $\_f(x) = {f\_disp}$\_ at $\_x = {p}$\_." \

 f" Please give your answer in slope-intercept form, or if applicable, in the form $\_x = a$\_. "

 explanation = f"<p>To find the equation of the line normal to the graph of $\_f(x)$\_ at $\_x = {p}$\_, " \

 f"we will first find the equation of the line tangent to the graph, " \

 f"then we will find the equation of the line normal to it. " \

 f"To do this we will need to recall" \

 "<ul>" \

 "<li>The <b>sum and difference rules</b> for derivatives state that the derivative of the " \

 "sum or difference of two functions is equal to the sum or difference of their derivative. " \

 "In math notation this means that " \

 "$$\\frac{{d}}{{dx}}\\left(f(x) \\pm g(x) \\right) = \\frac{{d}}{{dx}}f(x)" \

 f"\\pm \\frac{{d}}{{dx}}g(x).$$</li>" \

 "<li>The <b>extended power rule</b> for derivatives states that if " \

 "$\_n$\_ is a non-zero real number, then " \

 "$$\\frac{{d}}{{dx}}\\left(x^n \\right) = nx^{n-1}.$$</li>" \

 "<li>The <b>equation of the tangent line</b> to a function, $\_f(x)$\_, at the point " \

 "$\_x = a$\_ is given by $\_y = mx+b$\_ where $\_m$\_ is the value of the derivative of " \

 f"$\_f$\_ evaluated at $\_a$\_. In other words" \

 "$$m\_{\\text{tan}} = f'(a).$$</li>" \

 "<li>Lines that are <b>normal</b>, or perpendicular, to each other have slopes that are " \

 "negative reciprocals." \

 "</li></ul>" \

 "</p>" \

 f"<p>To find the equation of a line, we need a point on the line and the slope of the line. " \

 f"To find a " \

 f"point on the line, we will evaluate the function $\_f(x)$\_ at $\_x = {p}$\_. To find the " \

 f"slope of the line at $\_x = {p}$\_, we will find $\_f^{{\\prime}}({p})$\_.</p>" \

 f"<p>Evaluating the function $\_f(x)$\_ at $\_x = {p}$\_ gives " \

 f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f\\color{{red}}{{({p})}}&={f\_subs\_p}\\\\[5pt]" \

 f"&={sym.latex(fxp)}." \

 f"\\end{{align}}$$</p>" \

 f"<p>This gives us the point $\_\\left({p},{sym.latex(fxp)}\\right)$\_. " \

 f"<p>Now we need to define the slope of the tangent line at our point by finding the " \

 f"derivative of $\_f(x)$\_ and evaluating it for $\_x={p}$\_.</p>" \

 f"<p>Applying the sum and difference rules with the extended " \

 f"power rule {extra\_bit\_of\_text}gives " \

 f"$$\\begin{{align}}" \

 f"f(x)&= {f\_disp}\\\\[5pt]" \

 f"f^{{\\prime}}(x)&={first\_term}{middle\_term}{last\_term}\\\\[5pt]" \

 f"&={sym.latex(f\_diff)}.\\\\[5pt]" \

 f"\\end{{align}}$$</p>" \

 f"<p>Evaluating the derivative of $\_f(x)$\_ at $\_x={p}$\_ gives " \

 f"$$\\begin{{align}}" \

 f"f^{{\\prime}}(x)&={sym.latex(f\_diff)}\\\\[5pt]" \

 f"f^{{\\prime}}\\color{{red}}{{({p})}} &= {f\_diff\_subs}\\\\[5pt]" \

 f"&={slope}.\\\\[5pt]" \

 f"\\end{{align}}$$</p>" \

 f"<p>Using the point-slope formula, we can define the equation of the line tangent to" \

 f" $\_f(x)$\_ at $\_x = {p}$\_ as $${point\_slope\_form}.$$" \

 f"<p>Putting the equation of the line tangent to the graph of $\_f(x)$\_ at $\_x={p}$\_ " \

 f"in slope-intercept form, we obtain $$y={sym.latex(g)}.$$</p>" \

 f"<p>Now we can find the line normal to the graph of $\_f(x) = {f\_disp}$\_ at $\_x = {p}$\_. " \

 f"We know the line will pass through the point $\_\\left({p}, {slope}\\right)$\_. " \

 f"Therefore, we can write the equation of the line normal to the graph of " \

 f"$\_f(x)$\_ at $\_x={p}$\_ will be $\_{sym.latex(normal\_slope)}$\_."

 if slope == 0:

 explanation += f"This means the slope of the line normal to $\_f(x)$\_ is undefined," \

 f" and the equation of the line normal to $\_f(x)$\_ is therefore $\_x={p}$\_.</p>" \

 f"${{images/{image\_name}.svg|{metadata5}}}$"

 correct\_answer = f"x={p}"

 question\_template = ""

 else:

 explanation += f" We know the line will pass through the point " \

 f"$\_({p},{fxp})$\_, and we know the slope, therefore we can write the equation of the line" \

 f" normal to $\_f(x)$\_ using point-slope form." \

 f"$${point\_slope\_form\_normal}$$</p>" \

 f"<p>Putting the equation of the line normal to the graph of $\_f(x)$\_ at $\_x={p}$\_ " \

 f"in slope-intercept form we obtain $$y={sym.latex(h)}.$$</p>" \

 f"${{images/{image\_name}.svg|{metadata5}}}$"

 correct\_answer = f"y={sym.latex(h)}"

 question\_template = ""

 correct\_answer = correct\_answer

 question\_template = question\_template

 json\_blob = validations.lea\_blob(template=question\_template,

 response=correct\_answer,

 validation="equivSymbolic")

 return Problem(concepts="Find the equation of a line tangent to a polynomial function",

 question\_stem=question\_stem,

 explanation=explanation,

 correct\_answer=correct\_answer,

 json\_blob=json\_blob)

if \_\_name\_\_ == '\_\_main\_\_':

 os.chdir(sys.path[0])

 adaptive\_printer = Printer("Find the equation of a tangent line using basic derivative rules")

 quiz\_printer = Printer("Find the equation of a tangent line using basic derivative rules", is\_quiz=True)

 algo\_printer = Printer("Find the equation of a tangent line using basic derivative rules", is\_algo=True)

 formative\_printer = Printer("Find the equation of a tangent line using basic derivative rules", is\_formative=True)

 Templates = [Template1, Template2, Template3, Template4, Template5]

 adaptive\_printer.print\_all(\*[Template().take(10) for Template in Templates])

 quiz\_printer.print\_all(\*[Template().take(3) for Template in Templates])

 algo\_printer.print\_all(\*[Template().take(10) for Template in Templates])

 formative\_printer.print\_all(\*[Template().take(3) for Template in Templates])