Open in Colab

chapter 07 calculus problems

In [1]:
```
import math
math.sin(math.pi/2)
```

Out[1]: 1.0

In [ ]:
```
import sympy
sympy.sin(math.pi/2)
```

In [3]:
```
from sympy import sin, solve, Symbol
u = Symbol('u')
t = Symbol('t')
g = Symbol('g')
theta = Symbol('theta')
solve(u*sin(theta)-g*t,t)
```

Out[3]: [u*sin(theta)/g]

assumptions in symPy

In [4]:
```
from sympy import Symbol
x = Symbol('x', positive = True)
if (x+5) > 0:
  print('Do something')
else:
  print('Do something else')
```

```
Do something
```

Finding the limit of Functions

In [7]:
```
from sympy import Limit, Symbol, S
x = Symbol('x')
Limit(1/x,x,S.Infinity)
```

Out[7]: $\lim_{x \to \infty} \frac{1}{x}$

In [8]:
```
l = Limit(1/x,x,S.Infinity)
l.doit()
```

Out[8]: 0

In [9]:
```
Limit(1/x,x,0, dir='+').doit()
```

Out[9]: ∞

In [11]:
```
from sympy import Symbol, sin
Limit(sin(x)/x,x,0).doit()
```

Out[11]: 1

In [13]:
```
from sympy import Symbol, Limit, S
n= Symbol('n')
Limit((1+1/n)**n,n, S.Infinity).doit()
```

Out[13]: e

Continuous Compound Interest

In [14]:
```
from sympy import Symbol, Limit, S
p = Symbol('p', positive=True)
r = Symbol('r', positive=True)
t = Symbol('t', positive=True)
Limit(p*(1+r/n)**(n*t),n,S.Infinity).doit()
```

Out[14]: pe^{rt}

Instantaneous Rate of Change

In [18]:
```
from sympy import Symbol, Limit
t = Symbol('t')
st = 5*t**2 + 2*t + 8

t1 = Symbol('t1')
delta_t = Symbol('delta_t')

st1 = st.subs({t:t1})
st1_delta = st.subs({t:t1+delta_t})
Limit((st1_delta-st1)/delta_t, delta_t, 0).doit()
```

Out[18]: $10t_1 + 2$

finding derivative of functions

In [20]:
```
from sympy import Symbol, Derivative
t = Symbol('t')
st = 5*t**2 + 2*t + 8
Derivative(st,t)
```

Out[20]: $\frac{d}{dt}\left(5t^2 + 2t + 8\right)$

In [21]:
```
d = Derivative(st,t)
```

```
d.doit()
```

Out[21]: $10t + 2$

In [22]:
```
d.doit().subs({t:t1})
```

Out[22]: $10t_1 + 2$

In [23]:
```
d.doit().subs({t:1})
```

Out[23]: 12

In [24]:
```
from sympy import Derivative,Symbol
x = Symbol('x')
f = (x**3 + x**2 + x)*(x**2+x)
Derivative(f, x).doit()
```

Out[24]: $(2x+1)\left(x^3+x^2+x\right)+\left(x^2+x\right)\left(3x^2+2x+1\right)$

A Derivative Calculator

In [33]:
```
from sympy import Derivative, Symbol, sympify, pprint
from sympy.core.sympify import SympifyError

def derivative(f,var):
  var = Symbol(var)
  d = Derivative(f,var).doit()
  pprint(d)


if __name__ == '__main__':

  f = input('enter a function: ')
  var = input('enter the variable to differentiate with respect to:')
  try:
    f = sympify(f)
  except SympifyError:
    print('invalid input')
  else:
    derivative(f,var)
```

```
enter a function: 2*x**2 + 3*x + 1
enter the variable to differentiate with respect to:x
4·x + 3
```

higher-order Derivatives and Finding the Maxima and Minima

In [36]:
```
from sympy import Symbol, solve,Derivative
x = Symbol('x')
f = x**5 - 30*x**3 + 50*x
f = x **5 - 30 * x ** 3 + 50 * x
d1 = Derivative(f,x).doit()
```

```
critical_points = solve(d1)
critical_points
```

Out[36]:
```
[-sqrt(9 - sqrt(71)),
 sqrt(9 - sqrt(71)),
 -sqrt(sqrt(71) + 9),
 sqrt(sqrt(71) + 9)]
```

In [37]:
```
A = critical_points[2]
B = critical_points[0]
C = critical_points[1]
D = critical_points[3]
```

In [38]:
```
d2 = Derivative(f,x,2).doit()
```

In [39]:
```
d2.subs({x:B}).evalf()
```

Out[39]: 127.661060789073

In [40]:
```
d2.subs({x:C}).evalf()
```

Out[40]: -127.661060789073

In [41]:
```
 d2.subs({x:A}).evalf()
```

Out[41]: -703.493179468151

In [42]:
```
d2.subs({x:D}).evalf()
```

Out[42]: 703.493179468151

In [44]:
```
x = min = -5
x_max = 5
f.subs({x:A}).evalf()
```

Out[44]: $-30.0x^3 + 50.0x + x^{4.17446401028639}$

main Doing-math-with-python- / chapter 07 calculus problelms_.ipynb Top

Preview Code Blame Raw

```
def grad_ascent(x0,f1x,x):
  epsilon = 1e-6
  step_size = 1e-4
  x_old = x0
  x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()
  while abs(x_old - x_new) > epsilon:
    x_old = x_new
```

```
    x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()
    return x_new

def find_max_theta(R, theta):
    # Calculate the first derivative
    R1theta = Derivative(R, theta).doit()
    theta0 = 1e-3
    theta_max = grad_ascent(theta0, R1theta, theta)
    return theta_max

if __name__ == '__main__':
  g = 9.8
  #asumme initial velocity
  u = 25
  #expression for range
  theta = Symbol('theta')
  R = u**2*sin(2*theta)/g

  theta_max = find_max_theta(R, theta)
  print('theta :{0}'.format(math.degrees(theta_max)))
  print('maxium Range:{0}:'.format(R.subs({theta:theta_max})))
```

```
theta :1.5186448207743275
maxium Range:3.37920154330593:
```

A Generic Program for Gradient Ascent

In [55]:

```
from sympy import Derivative , Symbol, sympify

def grad_ascent(x0,f1x,x):
  step_size = 1e-3
  x_old = x0
  x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()

  return x_new

if __name__ == '__main__':
  f = input('enter a function in one variable:')
  var = input('enter the variable to differentiate  with respect to:')
  var0 = float(input('enter the inital value of the varible :'))
  try:
   f = sympify(f)
  except SympifyError:
    print('invalid function entered')

  else:
    var = Symbol(var)
    d = Derivative(f,var).doit()
    var_max = grad_ascent(var0,d,var)
    print('{0}:{1}'.format(var.name,var_max))
    print('maxium value:{0}'.format(f.subs({var:var_max})))
```

```
enter a function in one variable:25*25*sin(2*theta)/9.8
enter the variable to differentiate  with respect to:theta
enter the inital value of the varible :0.001
theta:0.128550765306207
```

```
maxium value:16.2167362349436
```

In [56]:

```
from sympy import Derivative , Symbol, sympify

def grad_ascent(x0,f1x,x):
  step_size = 1e-3
  x_old = x0
  x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()

  return x_new

if __name__ == '__main__':
  f = input('enter a function in one variable:')
  var = input('enter the variable to differentiate  with respect to:')
  var0 = float(input('enter the inital value of the varible :'))
  try:
   f = sympify(f)
  except SympifyError:
    print('invalid function entered')

  else:
    var = Symbol(var)
    d = Derivative(f,var).doit()
    var_max = grad_ascent(var0,d,var)
    print('{0}:{1}'.format(var.name,var_max))
    print('maxium value:{0}'.format(f.subs({var:var_max})))
```

```
enter a function in one variable:cos(y) +k
enter the variable to differentiate  with respect to:y
enter the inital value of the varible :0.01
y:0.00999000016666583
maxium value:k + 0.999950100363336
```

A Word of Warning About the Initial Value

In [57]:

```
from sympy import Derivative , Symbol, sympify

def grad_ascent(x0,f1x,x):
  step_size = 1e-3
  x_old = x0
  x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()

  return x_new

if __name__ == '__main__':
  f = input('enter a function in one variable:')
  var = input('enter the variable to differentiate  with respect to:')
  var0 = float(input('enter the inital value of the varible :'))
  try:
   f = sympify(f)
  except SympifyError:
    print('invalid function entered')

  else:
    var = Symbol(var)
    d = Derivative(f,var).doit()
    var_max = grad_ascent(var0,d,var)
    print('{0}:{1}'.format(var.name,var_max))
```

```
        print('{0}:{1}'.format(var.name,var_max))
        print('maxium value:{0}'.format(f.subs({var:var_max})))
```

```
enter a function in one variable:x**5 - 30*x**3 + 50*x
enter the variable to differentiate  with respect to:x
enter the inital value of the varible :-2
x:-2.23000000000000
maxium value:166.039702265700
```

In [58]:

```
from sympy import Derivative , Symbol, sympify

def grad_ascent(x0,f1x,x):
  step_size = 1e-3
  x_old = x0
  x_new = x_old + step_size*f1x.subs({x:x_old}).evalf()

  return x_new

if __name__ == '__main__':
  f = input('enter a function in one variable:')
  var = input('enter the variable to differentiate  with respect to:')
  var0 = float(input('enter the inital value of the varible :'))
  try:
   f = sympify(f)
  except SympifyError:
    print('invalid function entered')

  else:
    var = Symbol(var)
    d = Derivative(f,var).doit()
    var_max = grad_ascent(var0,d,var)
    print('{0}:{1}'.format(var.name,var_max))
    print('maxium value:{0}'.format(f.subs({var:var_max})))
```

```
enter a function in one variable:x**5 -30*x**3 + 50*x
enter the variable to differentiate  with respect to:x
enter the inital value of the varible :0.5
x:0.527812500000000
maxium value:22.0203528721678
```

The Role of the Step Size and Epsilon

In [61]:

```
from sympy import Derivative, Symbol, sympify, solve, SympifyError

def grad_ascent(x0, f1x, x):
    # Check if f1x=0 has a solution
    if not solve(f1x):
        print('cannot continue, solution for {0}=0 does not exist'.format(f1x
        return
    epsilon = 1e-6
    step_size = 1e-4
    x_old = x0
    x_new = x_old + step_size*f1x.subs({x: x_old}).evalf()
    while abs(x_old - x_new) > epsilon:
        x_old = x_new
        x_new = x_old + step_size*f1x.subs({x: x_old}).evalf()  # Fixed here
    return x_new
```

```python
if __name__ == '__main__':
    f = input('enter a function in one variable:')
    var = input('enter the variable to differentiate with respect to: ')
    var0 = float(input('enter the initial value of the variable: '))
    try:
        f = sympify(f)
    except SympifyError:
        print('invalid function entered')
    else:
        var = Symbol(var)
        d = Derivative(f, var).doit()
        var_max = grad_ascent(var0, d, var)
        if var_max:
            print('{0}: {1}'.format(var.name, var_max))
            print('maximum value: {0}'.format(f.subs({var: var_max})))
```

```
enter a function in one variable:log(x)
enter the variable to differentiate with respect to: x
enter the initial value of the variable: 0.1
cannot continue, solution for 1/x=0 does not exist
```

Finding the integrals of Functions

In [64]:

```python
from sympy import Integral, symbols
x = Symbol('x')
k = Symbol('k')
Integral(k*x,x)
```

Out[64]: $\int kx\,dx$

In [65]:

```python
Integral(k*x,x).doit()
```

Out[65]: $\frac{kx^2}{2}$

In [66]:

```python
Integral(k*x,(x,0,2)).doit()
```

Out[66]: $2k$

In [67]:

```python
from sympy import Integral, Symbol
x = Symbol('x')
Integral(x,(x,2,4)).doit()
```

Out[67]: 6

Probability Density Functions

In [68]:

```python
from sympy import Symbol, exp, sqrt, pi,integrals
x = Symbol('x')
p = exp(-(x-10)**2/2)/sqrt(2*pi)
Integral(p,(x,11,12)).doit().evalf()
```

```
Integral(p,(x,11,12)).doit().evalf()
```

Out[68]: 0.135905121983278

In [69]:

```
from sympy import Symbol, exp, sqrt,pi,Integral,S
x =Symbol('x')
p = exp(-(x-10)**2/2)/sqrt(2*pi)
Integral(p,(x,S.NegativeInfinity,S.Infinity)).doit().evalf()
```

Out[69]: 1.0

chapter ends.