**TUGAS 2**

**NAMA : YANUAR EKO ADI L**

**KELAS : B**

**MENAMPILKAN DATA ASCII**

1. **List of Code**

import string

g =[]

for i in range (100):

a = string.printable

e = len(a)

f = a[i]

g.append(f)

index\_ascii = []

for i in range(len(g)):

h = int(ord(g[i]))

index\_ascii.append(h)

j = {}

for i in range (len(g)):

j[int(index\_ascii[i])] = str(g[i])

print ("--------------------------")

print (" No |" + " oktal |"+ " hex |"+" sim |")

print ("--------------------------")

for i in range(128):

b = str(bin(i))

c = str(hex(i))

d = str(oct(i))

if int(i) in index\_ascii:

if len(str(i)) == 3 :

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(i)) == 2:

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(i)) == 1:

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else:

print (" "+str(i)+" | %s | %s | %s | ")%(d,c,j[i])

else :

if len(str(i)) == 3 :

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | | ")%(d,c)

else:

print (str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | | ")%(d,c)

else:

print (str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | | ")%(d,c)

else:

print (str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (str(i)+" | %s | %s | | ")%(d,c)

else:

print (str(i)+" | %s | %s | | ")%(d,c)

elif len(str(i)) == 2:

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(i)) == 1:

if len(str(d)) == 4 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 3 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

elif len(str(d)) == 2 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

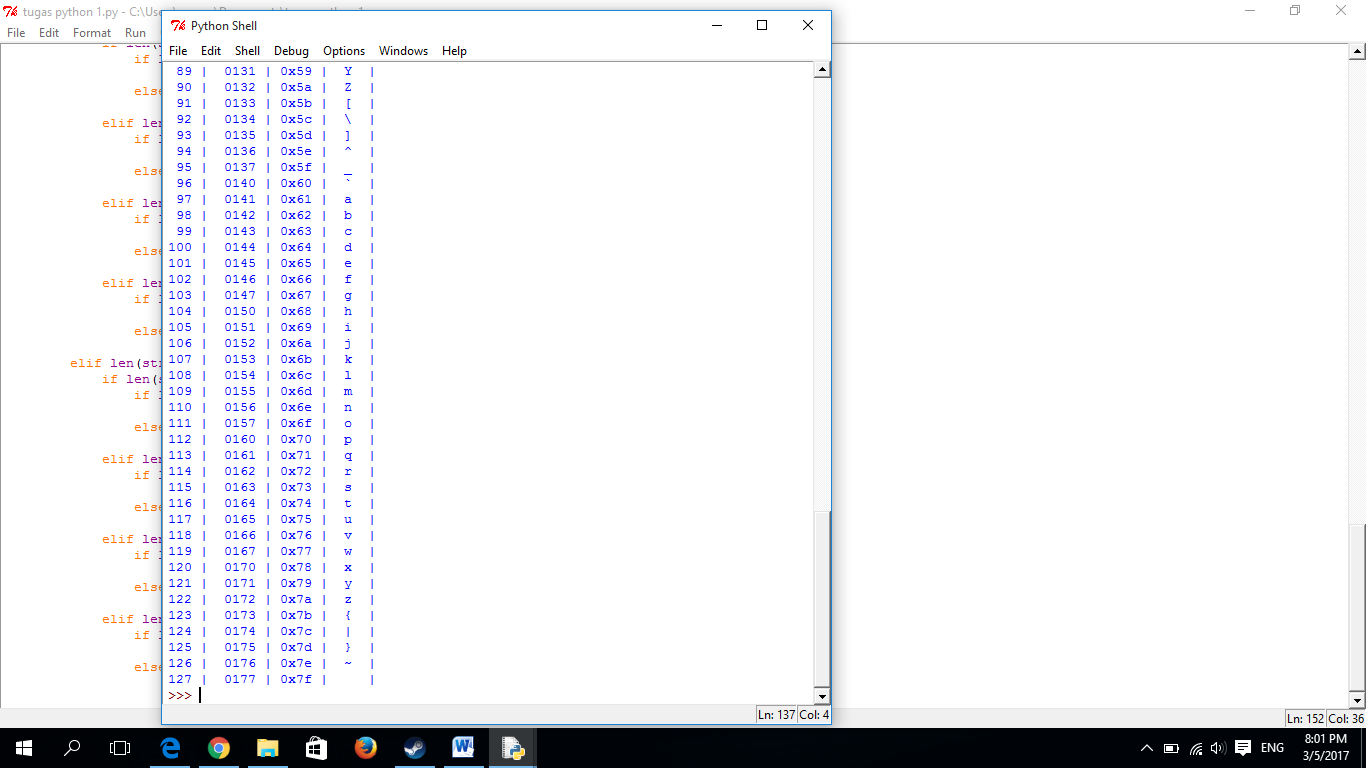
elif len(str(d)) == 1 :

if len(str(c)) == 3:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

else:

print (" "+str(i)+" | %s | %s | | ")%(d,c)

1. **Screenshot**

**TUGAS 2**

**FUNGSI DALAM MODUL MATH**

acos(...)

acos(x)

Return the arc cosine (measured in radians) of x.

acosh(...)

acosh(x)

Return the hyperbolic arc cosine (measured in radians) of x.

asin(...)

asin(x)

Return the arc sine (measured in radians) of x.

asinh(...)

asinh(x)

Return the hyperbolic arc sine (measured in radians) of x.

atan(...)

atan(x)

Return the arc tangent (measured in radians) of x.

atan2(...)

atan2(y, x)

Return the arc tangent (measured in radians) of y/x.

Unlike atan(y/x), the signs of both x and y are considered.

atanh(...)

atanh(x)

Return the hyperbolic arc tangent (measured in radians) of x.

ceil(...)

ceil(x)

Return the ceiling of x as a float.

This is the smallest integral value >= x.

copysign(...)

copysign(x, y)

Return x with the sign of y.

cos(...)

cos(x)

Return the cosine of x (measured in radians).

cosh(...)

cosh(x)

Return the hyperbolic cosine of x.

degrees(...)

degrees(x)

Convert angle x from radians to degrees.

erf(...)

erf(x)

Error function at x.

erfc(...)

erfc(x)

Complementary error function at x.

exp(...)

exp(x)

Return e raised to the power of x.

expm1(...)

expm1(x)

Return exp(x)-1.

This function avoids the loss of precision involved in the direct evaluation of exp(x)-1 for small x.

fabs(...)

fabs(x)

Return the absolute value of the float x.

factorial(...)

factorial(x) -> Integral

Find x!. Raise a ValueError if x is negative or non-integral.

floor(...)

floor(x)

Return the floor of x as a float.

This is the largest integral value <= x.

fmod(...)

fmod(x, y)

Return fmod(x, y), according to platform C. x % y may differ.

frexp(...)

frexp(x)

Return the mantissa and exponent of x, as pair (m, e).

m is a float and e is an int, such that x = m \* 2.\*\*e.

If x is 0, m and e are both 0. Else 0.5 <= abs(m) < 1.0.

fsum(...)

fsum(iterable)

Return an accurate floating point sum of values in the iterable.

Assumes IEEE-754 floating point arithmetic.

gamma(...)

gamma(x)

Gamma function at x.

hypot(...)

hypot(x, y)

Return the Euclidean distance, sqrt(x\*x + y\*y).

isinf(...)

isinf(x) -> bool

Check if float x is infinite (positive or negative).

isnan(...)

isnan(x) -> bool

Check if float x is not a number (NaN).

ldexp(...)

ldexp(x, i)

Return x \* (2\*\*i).

lgamma(...)

lgamma(x)

Natural logarithm of absolute value of Gamma function at x.

log(...)

log(x[, base])

Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.

log10(...)

log10(x)

Return the base 10 logarithm of x.

log1p(...)

log1p(x)

Return the natural logarithm of 1+x (base e).

The result is computed in a way which is accurate for x near zero.

modf(...)

modf(x)

Return the fractional and integer parts of x. Both results carry the sign

of x and are floats.

pow(...)

pow(x, y)

Return x\*\*y (x to the power of y).

radians(...)

radians(x)

Convert angle x from degrees to radians.

sin(...)

sin(x)

Return the sine of x (measured in radians).

sinh(...)

sinh(x)

Return the hyperbolic sine of x.

sqrt(...)

sqrt(x)

Return the square root of x.

tan(...)

tan(x)

Return the tangent of x (measured in radians).

tanh(...)

tanh(x)

Return the hyperbolic tangent of x.

trunc(...)

trunc(x:Real) -> Integral

Truncates x to the nearest Integral toward 0. Uses the \_\_trunc\_\_ magic method.