MODUL PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

INF1008

Penyusun :

Naufal Azmi Verdikha, M.Eng.

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2018

Praktikum 6:
Struktur Data Dasar (1)

Pokok Bahasan:

- Stack.
- Infix, Prefix, dan Postfix.

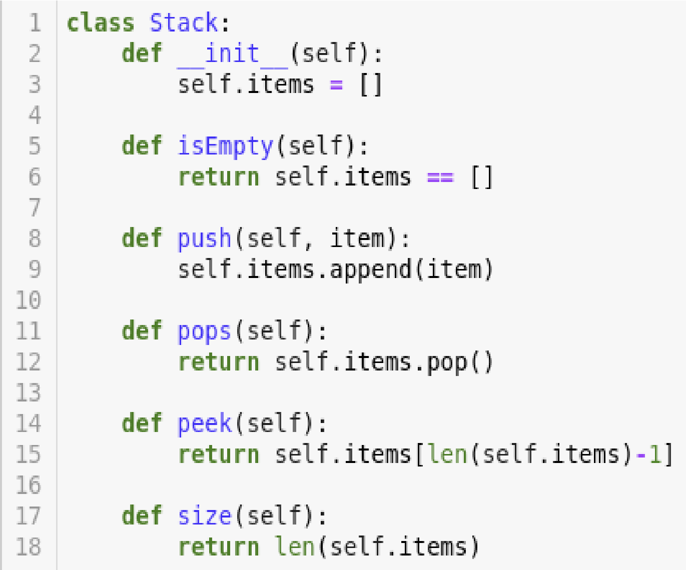
Tujuan Pembelajaran:

- ✓ Memahami implementasi *Stack* pada struktur data *Python.*
- ✓ Memahami penulisan ekspresi aritmatika di *Python.*

Stack:

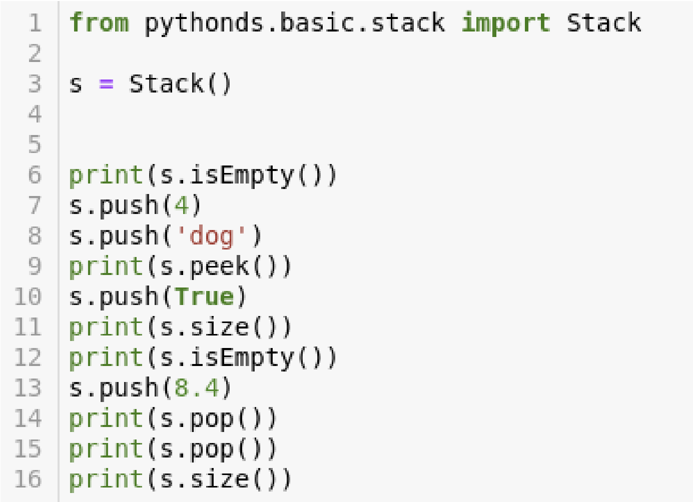
Percobaan & Latihan 6.1

Buatlah class berikut!



```python
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pops(self):
        return self.items.pop()

    def peek(self):
        return self.items[len(self.items)-1]

    def size(self):
        return len(self.items)
```

Jalankan class diatas dengan perintah berikut!



```python
from pythonds.basic.stack import Stack

s = Stack()


print(s.isEmpty())
s.push(4)
s.push('dog')
print(s.peek())
s.push(True)
print(s.size())
print(s.isEmpty())
s.push(8.4)
print(s.pop())
print(s.pop())
print(s.size())
```

Jika terjadi ModuleNotFoundError, jalankan perintah berikut:

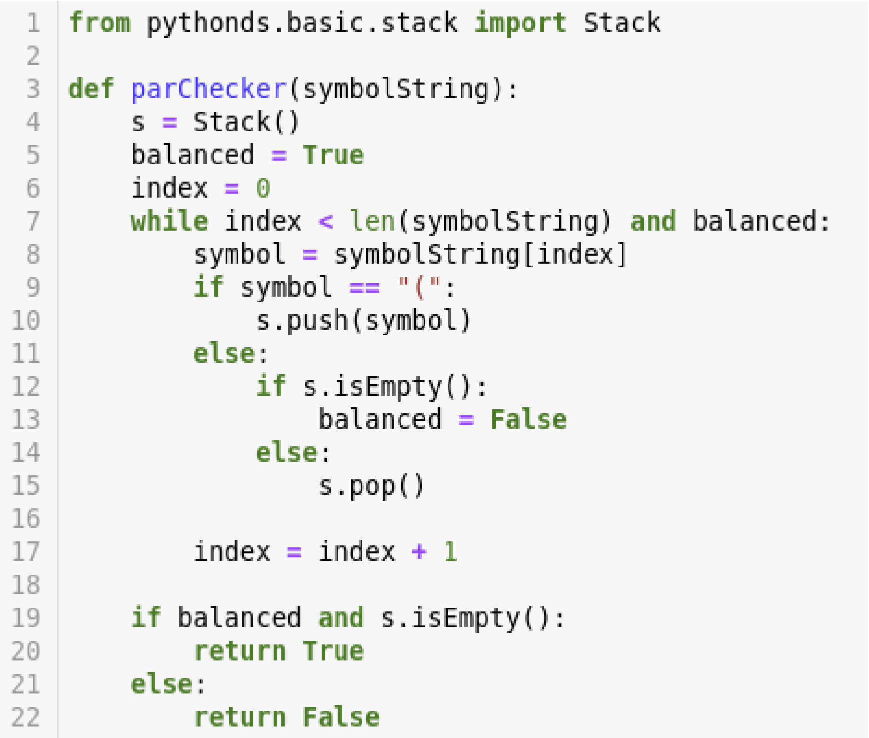
1. Buka command prompt windows.
2. Kemudian ketik *pip install pythonds* (pastikan komputer anda terkoneksi internet)

Soal :

a) Berikan tampilan output dari perintah diatas!
b) Analisa hasil dari perintah diatas!

Percobaan & Latihan 6.2

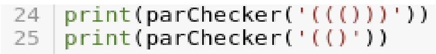
Buatlah perintah *Stack* berikut!



```python
from pythonds.basic.stack import Stack

def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol == "(":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                s.pop()

        index = index + 1

    if balanced and s.isEmpty():
        return True
    else:
        return False
```

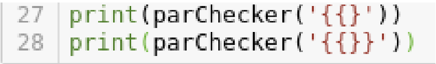
Soal :

a) Jalankan perintah berikut dan analisa hasil output.



```python
print(parChecker('((()))'))
print(parChecker('(()'))
```

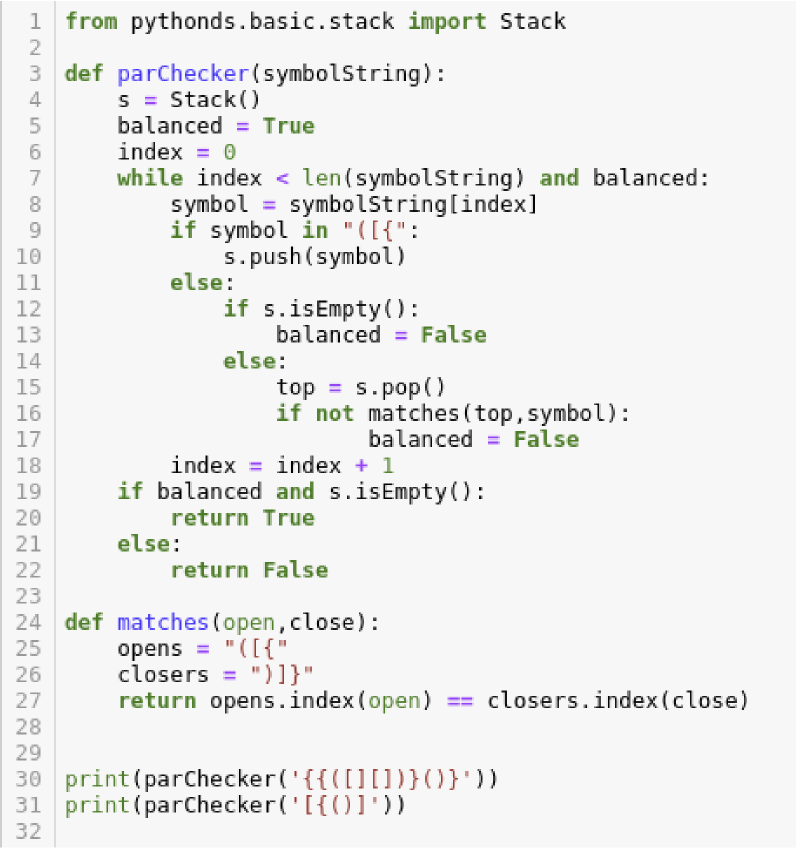
b) Jalankan perintah berikut dan berikan alasan mengapa menghasilkan ouput False!



```python
print(parChecker('{{}'))
print(parChecker('{{}}'))
```

Percobaan & Latihan 6.3

Jalankan perintah berikut!



```python
from pythonds.basic.stack import Stack

def parChecker(symbolString):
    s = Stack()
    balanced = True
    index = 0
    while index < len(symbolString) and balanced:
        symbol = symbolString[index]
        if symbol in "([{":
            s.push(symbol)
        else:
            if s.isEmpty():
                balanced = False
            else:
                top = s.pop()
                if not matches(top,symbol):
                    balanced = False
        index = index + 1
    if balanced and s.isEmpty():
        return True
    else:
        return False

def matches(open,close):
    opens = "([{"
    closers = ")]}"
    return opens.index(open) == closers.index(close)


print(parChecker('{{([][])}()}'))
print(parChecker('[{()]'))

```

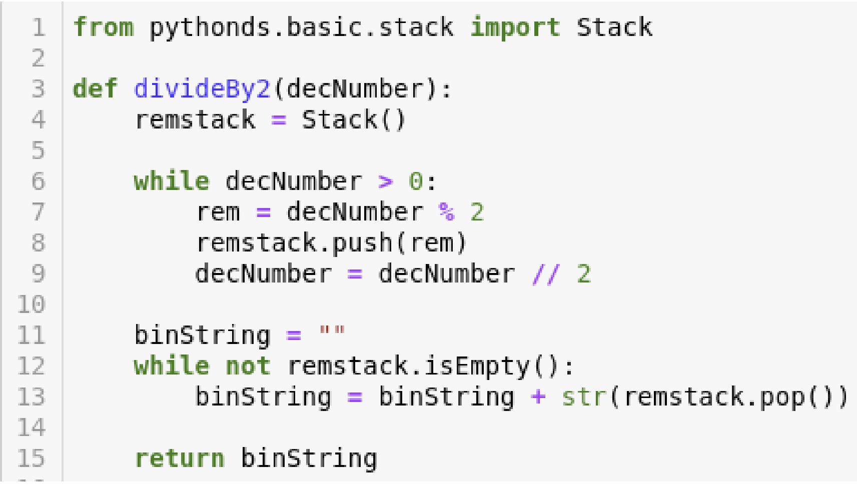
Soal :

a) Berikan tampilan ouput dari perintah diatas!

b) Jelaskan perbedaan fungsi diatas dengan fungsi yang terdapat di **Percobaan & Latihan 6.2**!

Percobaan & Latihan 6.4

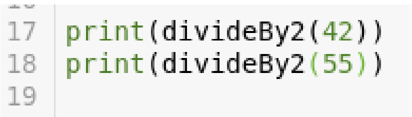
Jalankan fungsi berikut!



```python
from pythonds.basic.stack import Stack

def divideBy2(decNumber):
    remstack = Stack()

    while decNumber > 0:
        rem = decNumber % 2
        remstack.push(rem)
        decNumber = decNumber // 2

    binString = ""
    while not remstack.isEmpty():
        binString = binString + str(remstack.pop())

    return binString
```

Soal:

a) Jalankan fungsi diatas menggunakan perintah berikut!

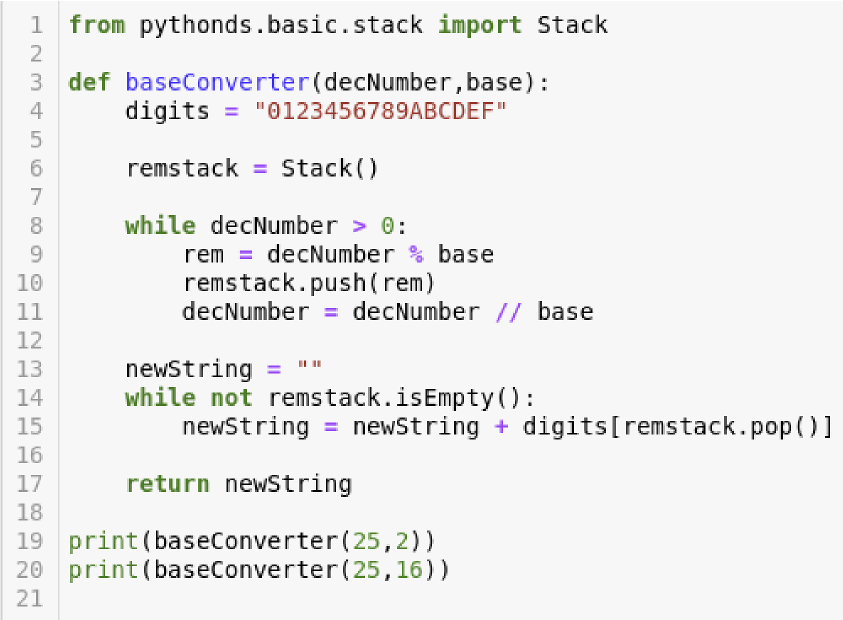


```python
print(divideBy2(42))
print(divideBy2(55))
```

b) Analisa hasil ouput diatas!

Percobaan & Latihan 6.5

Jalankan fungsi dan perintah berikut!



```python
from pythonds.basic.stack import Stack

def baseConverter(decNumber,base):
    digits = "0123456789ABCDEF"

    remstack = Stack()

    while decNumber > 0:
        rem = decNumber % base
        remstack.push(rem)
        decNumber = decNumber // base

    newString = ""
    while not remstack.isEmpty():
        newString = newString + digits[remstack.pop()]

    return newString

print(baseConverter(25,2))
print(baseConverter(25,16))
```

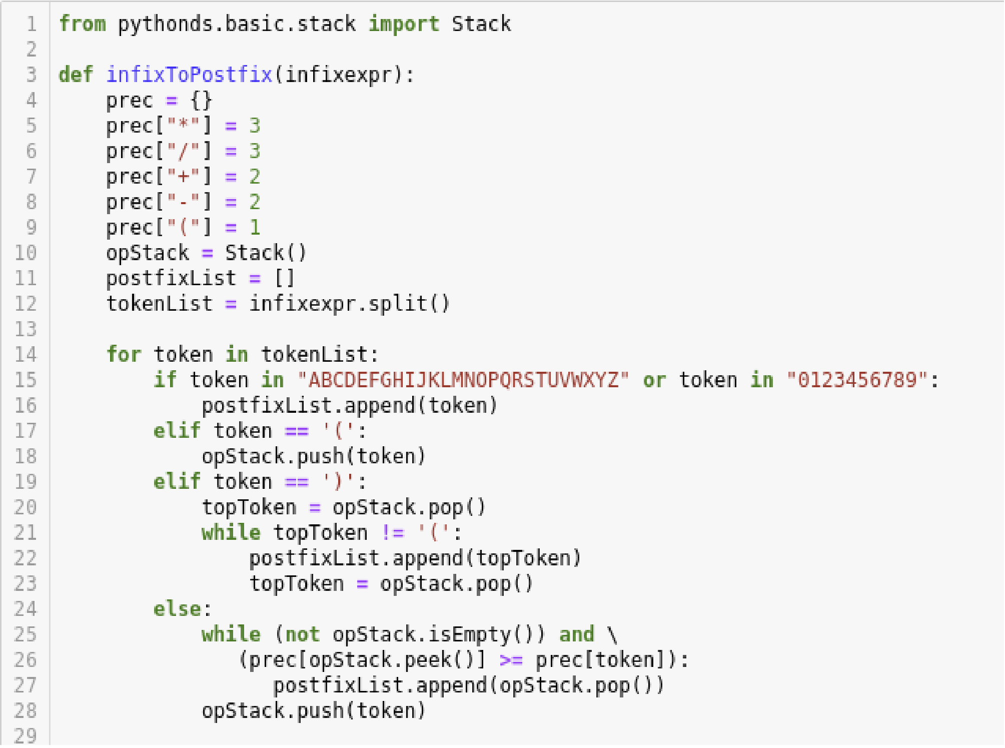
Soal:

a) Jelaskan perbedaan ouput dari kedua perintah diatas (ouput baris 19 dan 20)!

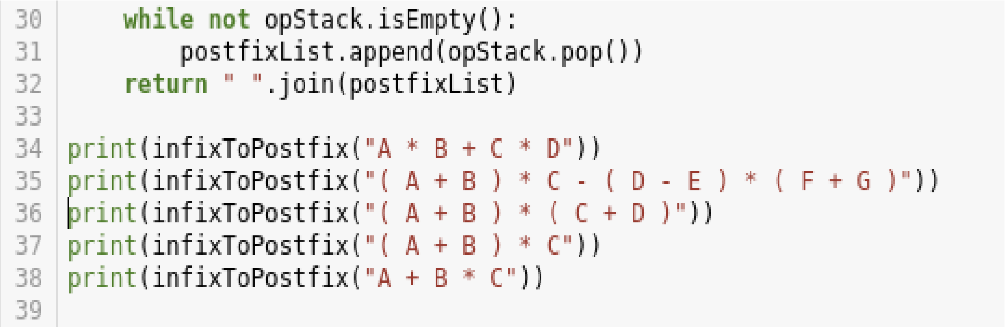
Infix, Prefix, dan Postfix:

Percobaan & Latihan 6.6

Jalankan perintah berikut!



```python
from pythonds.basic.stack import Stack

def infixToPostfix(infixexpr):
    prec = {}
    prec["*"] = 3
    prec["/"] = 3
    prec["+"] = 2
    prec["-"] = 2
    prec["("] = 1
    opStack = Stack()
    postfixList = []
    tokenList = infixexpr.split()

    for token in tokenList:
        if token in "ABCDEFGHIJKLMNOPQRSTUVWXYZ" or token in "0123456789":
            postfixList.append(token)
        elif token == '(':
            opStack.push(token)
        elif token == ')':
            topToken = opStack.pop()
            while topToken != '(':
                postfixList.append(topToken)
                topToken = opStack.pop()
        else:
            while (not opStack.isEmpty()) and \
               (prec[opStack.peek()] >= prec[token]):
                  postfixList.append(opStack.pop())
            opStack.push(token)

```



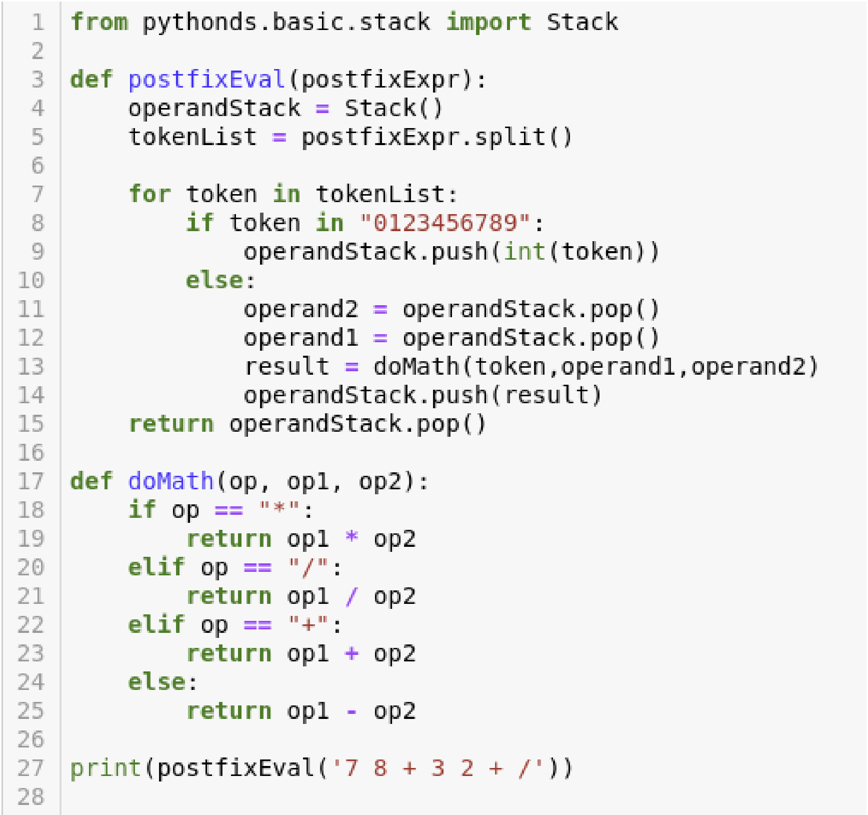
```python
    while not opStack.isEmpty():
        postfixList.append(opStack.pop())
    return " ".join(postfixList)

print(infixToPostfix("A * B + C * D"))
print(infixToPostfix("( A + B ) * C - ( D - E ) * ( F + G )"))
print(infixToPostfix("( A + B ) * ( C + D )"))
print(infixToPostfix("( A + B ) * C"))
print(infixToPostfix("A + B * C"))

```

Soal :

a) Berikan hasil output dari perintah diatas!
b) Berikan penjelasan tiap baris pada coding diatas!

Percobaan & Latihan 6.7

Jalankan fungsi dan perintah berikut!



```python
from pythonds.basic.stack import Stack

def postfixEval(postfixExpr):
    operandStack = Stack()
    tokenList = postfixExpr.split()

    for token in tokenList:
        if token in "0123456789":
            operandStack.push(int(token))
        else:
            operand2 = operandStack.pop()
            operand1 = operandStack.pop()
            result = doMath(token,operand1,operand2)
            operandStack.push(result)
    return operandStack.pop()

def doMath(op, op1, op2):
    if op == "*":
        return op1 * op2
    elif op == "/":
        return op1 / op2
    elif op == "+":
        return op1 + op2
    else:
        return op1 - op2

print(postfixEval('7 8 + 3 2 + /'))

```

Soal :

a) Berikan hasil output dari perintah diatas!

b) Berikan penjelasan tiap baris pada coding diatas!

Laporan Resmi:

1. Buatlah summary dan analisa dari **Percobaan & Latihan** pada pratikum ini.
2. Berikan kesimpulan dari praktikum ini.