

# PROGRAMMING BASIC(DAY 2)



# What we learn today

1. Pseudo Code
2. Why Pseudo Code
3. Variable Type
4. Operator
5. Sequential
6. Condition
7. CALCULATOR program with Pseudo Code
8. Looping
9. Triangle Star Program

# Pseudo Code

Pseudo Code , 伪代码 code 代码 program 程序  
伪代码 代码 基本 Basic 基本  
Pseudo Code, Flow Chart 伪代码, 流程图  
伪代码 流程图

Programming 编程 逻辑 Logic , 逻辑  
编程 语言 language 语言  
编程 逻辑  
编程 语言

Backend 后端 PHP (Laravel),  
Front End 前端 React (Javascript) , 反馈前端系统  
Machine Learning (Python) 机器学习  
编程 语言

language 语言  
编程 语言

# Pseudo Code

```
If student's grade is greater than or equal to 40
    Print "passed"
else
    Print "failed"
```



# Variable

- String
- Integer (-2147483648 through 2147483647)
- Float
- Boolean



# Python Variable Bit

- decimal 52
- dict 144
- float 16
- int 14
- list 32
- object 8
- set 112
- str 26
- tuple 24
- unicode 26

**1 bits = 1/0 (binary system)**

**8 bits = 1 bytes ,**

**1024 bytes = 1 KB ,**

**1024 KB = 1 MB**

**1024 MB = 1 GB**

**1024 GB = 1 TB**

THIS IS A SUBHEADLINE

---

# SEQUENCE

READ height of rectangle

READ width of rectangel

COMPUTE area as height times width

PYTHON

```
height = input ("Enter Height of Rectangle")
width = input ("Enter Width of Rectanble")
area = int(height) * int(width)
print("Area is ",area)
```



THIS IS A SUBHEADLINE

# CONDITION (IF)

```
READ height of rectangle
IF height is less than or equal zero
    Print "please enter more than zero"
ELSE IF height is greater than 100
    Print "please enter less than 101"
ELSE
    Print "Height is " + height
```

## PYTHON

```
height = int(input ("Enter Height of Rectangle"))
if height <= 0 :
    print("please enter more than zero")
elif height > 100 :
    print("please enter less than 100")
else:
    print("Height is ",height)
```

```
def __init__(self, settings):
    self.file = None
    self.fingerprints = set()
    self.logdups = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(settings['job_dir'], path), 'a')
        self.file.seek(0)
        self.fingerprints.update(self._load_file())
@classmethod
def from_settings(cls, settings):
    debug = settings.getbool('general', 'debug')
    return cls(job_dir=settings['job_dir'], debug=debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

THIS IS A SUBHEADLINE

# Operators

- +
- -
- \*
- /
- ==
- !=
- >
- >=
- <
- <=
- and
- or

```
def __init__(self, path=None, debug=False):
    self.file = None
    self.fingerprints = set()
    self.logdups = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(path, 'fingerprint.log'), 'a')
        self.file.seek(0)
        self.fingerprints.update(fingerprint for fingerprint in self.read())
    else:
        self.logdups = False
        self.debug = False
        self.logger.setLevel(logging.INFO)

@classmethod
def from_settings(cls, settings):
    job_dir = settings.get('job_dir')
    debug = settings.getboolean('debug', False)
    return cls(job_dir=job_dir, debug=debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# CONDITION (SWITCH)

```
CASE https_status OF
    200 :
        Print "OK"
        BREAK
    404 :
        Print "NOT FOUND"
        BREAK
    500 :
        Print "Internal Server Error"
        BREAK
    OTHERWISE:
        OUTPUT "Unknown Error"
        BREAK
ENDCASE
```

NO CASE IN PYTHON Until 3.10

```
def __init__(self, settings):
    self.file = None
    self.fingerprints = set()
    self.logdups = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(settings['job_dir'], path), 'a')
        self.file.seek(0)
        self.fingerprints.update(self._load_fingerprints())
@classmethod
def from_settings(cls, settings):
    debug = settings.getbool('debug', False)
    return cls(job_dir=settings['job_dir'], debug=debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Calculator Program

```
PRINT "Enter Number 1"  
INPUT number1  
  
PRINT "Enter Operator + , - , * ,  
/"  
INPUT operator  
  
PRINT "Enter Number 2"  
INPUT number2
```

```
IF operator is +  
    PRINT number1 + number2  
ELSE IF operator is -  
    PRINT number1 - number2  
ELSE IF operator is *  
    PRINT number1 * number2  
ELSE IF operator is /  
    PRINT number1 / number2
```

# LOOPING

□ lopping ☰☐☐

A horizontal row of 20 empty square boxes, intended for children to draw or write in.

A horizontal sequence of 20 empty square boxes, followed by a single asterisk (\*), and then another sequence of 15 empty square boxes.

# Looping while loop , for loop

While loop    condition

# FOR loop

# FOR LOOP

```
FOR counter = 1 TO 100 STEP 1 DO  
    OUTPUT counter  
ENDFOR
```

```
for counter in range(1, 101):  
    print(counter)
```

```
FOR counter = 1 TO 100 STEP 3 DO  
    OUTPUT counter  
ENDFOR
```

```
for counter in range(1, 101, 3):  
    print(counter)
```

# WHILE LOOP

```
counter is 1  
WHILE counter <= 100 THEN  
    OUTPUT counter  
    counter = counter + 1  
END WHILE
```

```
counter = 1  
while counter <= 100:  
    print(counter)  
    counter = counter + 1
```

# TRIANGLE STARS

Program 1

```
*  
**  
***  
****  
*****
```

Program 1

```
FOR row = 1 TO 5 STEP 1 DO  
    FOR col = 1 TO row STEP 1 DO  
        print "*"  
    END FOR  
    print "\n"  
END FOR
```

Program 2

```
*****  
****  
***  
**  
*
```

Python

```
for row in range(1,6):  
    for col in range(1,row + 1):  
        print("*",end="")  
    print("\n",end="")
```

SATURN GOD

---

**THANK YOU**