Essential Python 101

Today we are learning Python 101 for beginners.

- variables
- data types
- data structures
- function
- control flow
- OOP

```
print("hello world")
print("the world is mine")
```

```
hello world
the world is mine
```

```
print("I'm leraning python 101") # Go
```

```
I'm leraning python 101
```

```
# Basic calculation
print(2 +2)
print(2 - 2)
print(4 / 2)
print(5 * 2)
```

```
4
0
2.0
10
```

```
print(7 // 2)
print(pow(5, 4))
print(abs(-234)) # absolute
print(5 % 2) # modula
```

```
3
625
234
1
```

```
# 5 building blocks
# 1. variables
# 2. data types
# 3. data structures
# 4. function
# 5. control flow
# 6. OOP
```

```
# variables
my_name = "nay"
age = 27
gpa = 3.28
dog_lover = True
```

```
print(my_name, my_name)
gpa
```

```
nay nay
3.28
```

```
# over wirte a value
age = 27
new_age = age + 50
print(age, new_age)
```

```
27 77
```

```
iphone_price = 39990
discout = 0.2
new_iphone_price = iphone_price * (1-discout)
print(new_iphone_price)
```

```
31992.0
```

```
# remove variable
del new_ihpone_price
```

```
# count variable
age = 27
age += 1
age -= 2
age *= 2
age /= 3
print(age)
```

```
17.333333333333332
```

```
# data types
age = 27 #int
gpa = 3.28 #float
school = "Kasetsart" #str
dog_lover = True #bool
```

```
print( type(age))
print( type(gpa))
print( type(school))
print( type(dog_lover))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

```
# convert type
x = 100
x = str(x)
print(x, type(x))
```

```
100 <class 'str'>
```

```
y = False # T=1, F=0
y = int(y)
print(y, type(y))
```

```
0 <class 'int'>
```

```
r = 1
r = bool(r)
print(r, type(r))
```

```
True <class 'bool'>
```

```
age = 27
print(age+age, age*2, age/2)
```

```
54 54 13.5
```

```
text = "what's up"
text2 = '"wowwwww"'
print(text, text2)
```

```
what's up "wowwwww"
```

```
print(text*4)
```

```
what's upwhat's upwhat's upwhat's up
```

```
# type hint
age: int = 27
```

```
my_name: str = "Nay"
gpa: float = 3.28
seafood: bool = True
```

```
# function
print("hello, world")
print(pow(5, 3), abs(-4))
```

```
hello, world
125 4
```

```
#greeting()
def greeting(name="Nay", location="London"):
    print("Hello!", name)
    print("He is at", location)
```

```
greeting(location="Japan", name="Bob")
```

```
Hello! Bob
He is at Japan
```

```
def add_two_nums(num1, num2, num3):
    print("the world")
    return num1 + num2 + num3
```

```
x = add_two_nums(15, 5, 6)
print(x)
```

```
the world
26
```

```
# work with string
text = "Hello world"

long_text = """
this is a
very long text
this is a new line"""

print(text)
print(long_text)
```

```
Hello world

this is a
very long text
this is a new line
```

```
# string template : fstrings
my_name = "John"
location = "London"

text = f"Hi! my name is {my_name} and I live in {location}"

print(text)
```

```
Hi! my name is John and I live in London
```

```
text = "a duck walks into a bar"
print(text)
```

```
a duck walks into a bar
```

```
len(text)
```

```
23
```

```
# slicing, index start with 0
print(text[0], text[22], text[-1])
```

```
a r r
```

```
print(text[2:6])
print(text[7: ])
print(text[-3: ])
```

```
duck
walks into a bar
bar
```

```
# string is immutable
name = "Python" # - > Cython
name = "C" + name[1: ]
print(name)
```

```
Cython
```

```
text = "a duck walks into a bar"
len(text)
```

```
23
```

```
# function vs. method
# string methods
text = text.upper()
print(text)
```

```
A DUCK WALKS INTO A BAR
```

```
text.lower()
```

```
'a duck walks into a bar'
```

```
text.title()
```

```
'A Duck Walks Into A Bar'
```

```
text.replace("DUCK", "LION")
```

```
'A LION WALKS INTO A BAR'
```

```
text.split(" ")
```

```
['A', 'DUCK', 'WALKS', 'INTO', 'A', 'BAR']
```

```
words = text.split(" ")
print(words, type(words))
```

```
['A', 'DUCK', 'WALKS', 'INTO', 'A', 'BAR'] <class 'list'>
```

```
" ".join(words)
```

```
'A DUCK WALKS INTO A BAR'
```

```
# data structures
# 1. list []
# 2. tuple ()
# 3. dictionary {}
# 4. set {unique}
```

```
# list
shopping_item = ["banana", "eeg", "milk"]
print(shopping_item[0])
print(shopping_item[1])
print(shopping_item[2])
print(shopping_item[1:])
print(len(shopping_item))
```

```
banana
eeg
milk
['eeg', 'milk']
3
```

```
shopping_item = ["banana", "eeg", "milk"]

shopping_item[0] = "pine apple"
shopping_item[1] = "cheese"

print(shopping_item)
```

```
['pine apple', 'cheese', 'milk']
```

```
# list methods
shopping_item.append("egg")
print(shopping_item)
```

```
['pine apple', 'cheese', 'milk', 'egg', 'egg']
```

```
# sort items (ascending order, A-Z)
shopping_item.sort()
print(shopping_item)
```

```
['cheese', 'egg', 'egg', 'milk', 'pine apple']
```

```
shopping_item.sort(reverse=True) # descending order
print(shopping_item)
```

```
['pine apple', 'milk', 'egg', 'egg', 'cheese']
```

```
# reuseable
def mean(scores):
    return sum(scores) / len(scores)
```

```
scores = [90, 88, 85, 92, 75]
print(len(scores), sum(scores), min(scores), max(scores), mean(scores))
```

```
5 430 75 92 86.0
```

```
# remove last item in list
shopping_item.pop()
print(shopping_item)
```

```
['pine apple', 'milk', 'egg']
```

```
shopping_item.append("egg")
shopping_item
```

```
['pine apple', 'milk', 'egg', 'egg']
```

```
shopping_item.remove("milk")
shopping_item
```

```
['pine apple', 'egg', 'egg']
```

```
# .insert()
shopping_item.insert(1, "milk")
shopping_item
```

```
['pine apple', 'milk', 'egg', 'egg']
```

```
# list + list
item1 = ["egg", "milk"]
item2 = ["banana", "bread"]

print(item1 + item2)
```

```
['egg', 'milk', 'banana', 'bread']
```

```
# tuple () is immutable
tup_items = ("egg", "bread", "pepsi", "egg", "egg")
tup_items
```

```
('egg', 'bread', 'pepsi', 'egg', 'egg')
```

```
tup_items.count("egg")
```

```
3
```

```
# username password
# student1, student2
s1 = ("id001", "123456")
s2 = ("id002", "654321")

user_pw = (s1, s2)
print(user_pw)
```

```
(('id001', '123456'), ('id002', '654321'))
```

```
# tuple unpacking
username, password = s1

print(username, password)
```

```
id001 123456
```

```
# tuple unpacking 3 values
name, age, _ = ("John", 38, 3.4)
print(name, age)
```

```
John 38
```

```
# set {unique}
courses = ["Python", "Python", "R", "SQL",]
```

```
set(courses)
```

```
{'Python', 'R', 'SQL'}
```

```
# dictionary key: value pairs
course = {
    "name": "Data Science Bootcamp",
    "duration": "4 months",
    "students": 400,
    "replay": True,
    "skills": ["Google Sheets", "SQL", "R", "Python", "Stats", "ML", "Dashboard", "Data tranformation"]
}
```

```
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 400,
 'replay': True,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data tranformation']}
```

```
course["name"]
```

```
'Data Science Bootcamp'
```

```
course["start_time"] = "9am"

course["language"] = "Thai"

course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 400,
 'replay': True,
 'skills': ['Google Sheets',
  'SQL',
```

```
          'R',
          'Python',
          'Stats',
          'ML',
          'Dashboard',
          'Data tranformation'],
         'start_time': '9am',
         'language': 'Thai'}
```

```
1 # delete
2 del course["start_time"]
3 course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 400,
 'replay': True,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data tranformation']}
```

```
1 # update data
2 course["replay"] = False
3 course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 400,
 'replay': False,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data tranformation']}
```

```
1 course["skills"][0:3]
```

```
['Google Sheets', 'SQL', 'R']
```

```
1 course["skills"][-3:]
```

```
['ML', 'Dashboard', 'Data tranformation']
```

```
1 list( course.keys() )
```

```
['name', 'duration', 'students', 'replay', 'skills']
```

```
1 list( course.values())
```

```
['Data Science Bootcamp',
 '4 months',
 400,
 False,
 ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data tranformation']]
```

```
1 list( course.items() )
```

```
[('name', 'Data Science Bootcamp'),
 ('duration', '4 months'),
 ('students', 400),
 ('replay', False),
 ('skills',
  ['Google Sheets',
```

```
        'SQL',
        'R',
        'Python',
        'Stats',
        'ML',
        'Dashboard',
        'Data tranformation'])]
```

```
course.get("replay")
```

```
False
```

```
# Recap
# list, dictionary = mutable
# tuple, string = immutable
```

```
# control flow
# if for while
```

```
# final exam 150 questions, pass>=120
score = 125
if score >=120:
    print("passed")
else :
    print("failed")
```

```
passed
```

```
def grade(score):
    if score >=120:
        return "passed"
    else :
        return "failed"
```

```
result = grade(119)
print(result)
```

```
failed
```

```
def grade(score):
    if score >=120:
        return "Excellent"
    elif score >= 100:
        return "Good"
    elif score >= 80:
        return "Okay"
    else :
        return "Need to read more!"
```

```
result = grade(95)
print(result)
```

```
Okay
```

```
# use and, or in condition
# course == data science, score >= 80 passed
# course == english, score >=70 passed
def grade(course, score):
    if course == "data science" and score >= 80:
        return "passed"
    if course == "english" and score >= 70:
        return "passed"
    else :
        return "failed"
```

```
result = grade("english ", 60)
print(result)
```

```
failed
```

```
# for loop
# if socre >= 80 passed
```

```
scores = [88, 90, 75]

for score in scores:
    print(score)
```

```
88
90
75
```

```
new_scores = []

for score in scores:
    new_scores.append(score-2)
print(new_scores)
```

```
[86, 88, 73]
```

```
def grading_all(scores):
    new_scores = []
    for score in scores:
        new_scores.append(score+2)
    return new_scores
```

```
grading_all([75, 88, 90, 95, 52])
```

```
[77, 90, 92, 97, 54]
```

```
# list comprehension
scores = [75, 88, 90, 95, 52]
```

```
for s in scores:
    print(s*2)
```

```
150
176
180
190
104
```

```
new_scores = [s*2 for s in scores]
new_scores
```

```
[150, 176, 180, 190, 104]
```

```
friends = ["toy", "ink", "bee", "zue", "yos"]
for f in friends:
    print(f.upper())
```

```
TOY
INK
BEE
ZUE
YOS
```

```
[f.upper() for f in friends]
```

```
['TOY', 'INK', 'BEE', 'ZUE', 'YOS']
```

```
# while loop
count = 0

while count < 5:
    print("hello")
    count += 1
```

```
hello
hello
hello
hello
hello
```

```
# chatbot for fruit order
user_name = input("What is your name? ")
```

```
What is your name? John Wick
```

```
user_name
```

```
'John Wick'
```

```
def chatbot():
    fruits = []
    while True:
        fruit = input("What fruit do you want to order? ")
        if fruit == "exit":
            return fruits
        fruits.append(fruit)
```

```
chatbot()
```

```
What fruit do you want to order? banana
What fruit do you want to order? mango
What fruit do you want to order? grape
What fruit do you want to order? exit
['banana', 'mango', 'grape']
```

```
age = input("How old are you? ")
```

```
How old are you? 27
```

```
type(age)
```

```
str
```

```
age = int( input("How old are you? "))
```

```
How old are you? 27
```

```
type(age)
```

```
int
```

```
# OOP - Object Oriented Programming
# Dog class  __ # dunder
```

```
class Dog:
    def __init__(self, name, age, breed):
        self.name = name
        self.age = age
        self.breed = breed
```

```
dog1 = Dog("billy", 2, "chihuahua")
dog2 = Dog("van", 5, "golden retriver")
dog3 = Dog("smurf", 10, "bulldog")
```

```
print(dog3.name, dog3.age, dog3.breed)
```

```
smurf 10 bulldog
```

```
class Employee:
    def __init__(self, id, name, dept, pos):
        self.id = id
        self.name = name
        self.dept = dept
        self.pos = pos

    def hello(self):
        print(f"Hello! my name is {self.name}")

    def work_hours(self, hours):
        print(f"{self.name} work for {hours} hours.")

    def change_dept(self, new_dept):
        self.dept = new_dept
        print(f"{self.name} is now in {self.dept}.")
```

```
emp1 = Employee(1, "John", "Finace", "Financial Analyst")
```

```
print(emp1.name, emp1.pos)
```

```
John Financial Analyst
```

```
emp1.hello()
```

```
Hello! my name is John
```

```
emp1.work_hours(10)
```

```
John work for 10 hours.
```

```
emp1.dept
```

```
'Finace'
```

```
emp1.change_dept("Marketing")
```

```
John is now in Marketing.
```

```
emp1.dept
```

```
'Marketing'
```

```
# Object: attribute => name, id, dept, pos
# Object: method => hello(), change_dept()
```

```
# create new ATM class

class ATM:
    def __init__(self, name, bank, balance):
        self.name = name
        self.bank = bank
        self.balance = balance
    def deposit(self, amt):
        self.balance += amt
    def withdraw(self, amt):
        self.balance -= amt
```

```
scb = ATM("Nay", "scb", 500)
```

```
scb.balance
```

```
500
```

```
scb.deposit(100)
```

```
scb.balance
```

```
600
```

```
scb.withdraw(200)
```

```
scb.balance
```

```
400
```

✓ 0 วินาที เสร็จสมบูรณ์เมื่อ 19:03