Python Programming 101

sunsun-datateathyme\

Basic Python for Data Analyst (Beginner)

- variable
- data type
- data structure
- control flow
- function

```
## 1. variable
x = 100
y = 200
print(x + y)
```

```
300
```

```
my_university_name = "Rangsit"
print(my_university_name)
```

```
Rangsit
```

```
# delete variable
del my_university_name
```

```
## replace value
x = 100
x =150
print(x)
```

```
150
```

```
# ประกาศตัวแปร บรรทัดเดียวกันเลย
x, y, z = 1,"math", 3
print(x, y, z)
```

```
1 math 3
```

```
## R vs. Python
# R for small data, prototyping
# Python for larger data, software, data science, ai, app
# R is statistical language vs. Python is a general language
# R is specialist (stats)
# Python is generalist (app, data, software, ai)
```

```
## variable
my_name = "jane"
age = 25
friends = ["ann", "mae", "joe"] # list
fav_food = ("hotdog", "coke", "french fries") # tuple
```

```
# update value
friends[0] = "kittipong"
```

```
friends
```

```
['kittipong', 'mae', 'joe']
```

```
## 2. data types
## int, float, str, bool
age = 25
type(age)
```

```
int
```

```
gpa = 3.72
type(gpa)
```

```
float
```

```
name = "sunsun"
type(name)
```

```
str
```

```
result = age < 40
print(result, type(result))
```

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

```
my_bool = True, False, True
print(my_bool, type(my_bool))
```

```
(True, False, True) <class 'tuple'>
```

```
my_bool = [True, False, True]
print(my_bool, type(my_bool))
```

```
[True, False, True] <class 'list'>
```

```
## convert type
str(25)
```

```
'25'
```

```
int("25")
```

```
25
```

```
float("25")
```

```
25.0
```

```
## True = 1, False = 0
bool(0)
```

```
False
```

🆎 Working with String

```
## fstring => format string
name = "sunsun"
gpa = 3.72

text = f"{name} graduates from RSU with gpa {gpa}"
print(text)
```

```
sunsun graduates from RSU with gpa 3.72
```

```
## long string
long_str = """
I love McDonald's
Planning to have it for Lunch
Very Cool!
"""
```

```
long_str
```

```
' \nI love McDonald's \nPlanning to have it for Lunch \nVery Cool!\n'
```

```
## function vs. method
text = "a duck walks into a bar"
len(text)
print(text)
```

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

```
## method is a function created specifically to an object
## string method
text.upper()
```

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

```
## replace new value
text = text.replace("duck", "lion")
print(text)
```

```
a lion walks into a bar
```

```
text.count("a")
```

```
4
```

```
"strawberry".count("r")
```

```
3
```

```
result = text.split(" ")
print(result)
```

```
['a', 'lion', 'walks', 'into', 'a', 'bar']
```

```
text = " ".join(result)
print(text)
```

```
a lion walks into a bar
```

```
"-".join(result)
```

```
'a-lion-walks-into-a-bar'
```

```
## index in python starts with 0
text = "python"
text[0]
```

```
'p'
```

```
text[-1]
```

```
'n'
```

```
## slice text
text = "python"
text[3:6]
```

```
'hon'
```

```
text[1: ]
```

```
'ython'
```

```
## text + text
"Python" + " is awesome"+ " and I love it"
```

```
'Python is awesome and I love it'
```

```
## start and jump
"I am learning Python today"[0:15:2]
```

```
'Ia erigP'
```

```
## string is immutable
text = "python"

text[0] = "c"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-3077004063.py in <cell line: 0>()
      2 text = "python"
      3
----> 4 text[0] = "c"

TypeError: 'str' object does not support item assignment
```

Next steps: Explain error

```
## how to solve
text = "python"

print("c" + text[1: ])
```

```
cython
```

Data Structures

1. List
2. Tuple
3. Dictionary
4. Set

1. list, ordered, mutable object

```
## list, ordered, mutable object
shopping_list = ["egg", "milk", "vitamilk", "bread"]
```

```
shopping_list
```

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

```
len(shopping_list)
```

```
4
```

```
shopping_list[-4]
```

```
'egg'
```

```
# list can be update value (replace)
shopping_list[0] = "butter"
shopping_list
```

```
['butter', 'milk', 'vitamilk', 'bread']
```

```
## list method
shopping_list.append("banana")
shopping_list
```

```
['butter', 'milk', 'vitamilk', 'bread', 'banana']
```

```
shopping_list.pop()
```

```
'banana'
```

```
shopping_list
```

```
['butter', 'milk', 'vitamilk', 'bread']
```

```
shopping_list.remove("milk")
```

```
shopping_list
```

```
['butter', 'vitamilk', 'bread']
```

```
shopping_list.insert(1, "milk")
shopping_list
```

```
['butter', 'milk', 'milk', 'vitamilk', 'bread']
```

```
shopping_list.insert(2, "butter")
```

```
shopping_list
```

```
['butter', 'milk', 'butter', 'milk', 'vitamilk', 'bread']
```

```
shopping_list.count("milk")
```

```
2
```

```
shopping_list.reverse()
shopping_list
```

```
['bread', 'vitamilk', 'milk', 'butter', 'milk', 'butter']
```

```
shopping_list.sort()
shopping_list
```

```
['bread', 'butter', 'butter', 'milk', 'milk', 'vitamilk']
```

```
## list + list
["item1" , "item2"] + ["item3", "item4"]
```

```
['item1', 'item2', 'item3', 'item4']
```

```
## loop through shopping list
for item in shopping_list:
    print(item)
```

```
bread
butter
butter
milk
milk
vitamilk
```

```
for item in shopping_list:
    print("I have to buy " + item)
```

```
I have to buy bread
I have to buy butter
I have to buy butter
I have to buy milk
I have to buy milk
I have to buy vitamilk
```

```
for item in shopping_list:
    if len(item) >= 6:
        continue
    else:
        print(f"I need to buy {item}")
```

```
I need to buy bread
I need to buy milk
I need to buy milk
```

```
## average revenue per user (ARPU)
spending = [500, 1200, 800, 300, 900]

for spend in spending:
    if spend >= 900:
        print("high spender")
    else:
```

```
        print("low spender")
```

```
low spender
high spender
low spender
low spender
high spender
```

```
## list comprehension
scores = [80, 90, 75, 60, 59, 82]
for score in scores:
    if score >= 80:
        print(score, "passed")
    else:
        print(score, "failed")
```

```
80 passed
90 passed
75 failed
60 failed
59 failed
82 passed
```

```
## example list comprehension
scores = [80, 90, 75, 60, 59, 82]

new_scores = [score + 5 for score in scores ]
print(new_scores)
```

```
[85, 95, 80, 65, 64, 87]
```

```
grades = ["passed" if score >= 80 else "failed" for score in scores]
print(grades)
```

```
['passed', 'passed', 'failed', 'failed', 'failed', 'passed']
```

2. tuple, ordered, immutable

```
## tuple, ordered, immutable
## tuple unpacking
toy, jane, ann = (36, 29, 32)
print(toy, jane, ann)
```

```
36 29 32
```

```
names = ("toy", "joe", "john")
names.index("joe")
```

```
1
```

```
for name in names:
    print(f"Hello! {name.capitalize()}")
```

```
Hello! Toy
Hello! Joe
Hello! John
```

```
## recap list
complex_list = [
    25, "The Dark Knight",
    [1, 2, 3, 4, 5],
    ("hello", "ni hao", "sawasdee")
]

complex_list
```

```
[25, 'The Dark Knight', [1, 2, 3, 4, 5], ('hello', 'ni hao', 'sawasdee')]
```

```
complex_list[3][1]
```

```
'ni hao'
```

3. Dictionary

```
## Dictionary
## key-value pair (similar to json)

movie = {
    "title": "The Hitchhiker's Guide to the Galaxy",
    "author": "Douglas Adams",
    "publishedYear": 1979,
    "genres": [ "Science fiction","Comedy"],
    "isInPrint": True
}

movie
```

```
{'title': "The Hitchhiker's Guide to the Galaxy",
 'author': 'Douglas Adams',
 'publishedYear': 1979,
 'genres': ['Science fiction', 'Comedy'],
 'isInPrint': True}
```

```
customer_01 = {
    "name": "john wick",
    "age": 50,
    "fav_movies": ["Superman", "Inside out", "Lion King"],
    "gpa": 3.41
}

customer_01
```

```
{'name': 'john wick',
 'age': 50,
 'fav_movies': ['Superman', 'Inside out', 'Lion King'],
 'gpa': 3.41}
```

```
## dictionary is unordered, mutable
customer_01["name"].upper()
```

```
'JOHN WICK'
```

```
customer_01["fav_movies"]
```

```
['Superman', 'Inside out', 'Lion King']
```

```
customer_01["fav_movies"][1]
```

```
'Inside out'
```

```
customer_01["fav_movies"][0::2]
```

```
['Superman', 'Lion King']
```

```
 ## dictionary method
list(customer_01.keys())
```

```
['name', 'age', 'fav_movies', 'gpa']
```

```
list(customer_01.values())
```

```
['john wick', 50, ['Superman', 'Inside out', 'Lion King'], 3.41]
```

```
list(customer_01.items())
```

```
[('name', 'john wick'),
 ('age', 50),
 ('fav_movies', ['Superman', 'Inside out', 'Lion King']),
 ('gpa', 3.41)]
```

```
## create new key
customer = customer_01
customer["city"] = "Bangkok"
customer["nationality"] = "American"
```

```
customer
```

```
{'name': 'john wick',
 'age': 50,
 'fav_movies': ['Superman', 'Inside out', 'Lion King'],
 'gpa': 3.41,
 'city': 'Bangkok',
 'nationality': 'American'}
```

```
## remove the key
del customer["nationality"]
customer
```

```
{'name': 'john wick',
 'age': 50,
 'fav_movies': ['Superman', 'Inside out', 'Lion King'],
 'gpa': 3.41,
 'city': 'Bangkok'}
```

```
## use method to remove
customer.pop("city")
customer
```

```
{'name': 'john wick',
 'age': 50,
 'fav_movies': ['Superman', 'Inside out', 'Lion King'],
 'gpa': 3.41}
```

```
## update value
customer["age"] = 51
customer
```

```
{'name': 'john wick',
 'age': 51,
 'fav_movies': ['Superman', 'Inside out', 'Lion King'],
 'gpa': 3.41}
```

🎯The last data structure: set

```
## The last data structure: set
## set ie used to find distinct/ unique values
set([1, 1, 2, 3, 4])
```

```
{1, 2, 3, 4}
```

```
set(["orange", "orange", "banana"])
```

```
{'banana', 'orange'}
```

```
## set operation
## union and intersection
mary = {"orange", "apple"}
jay = {"orange", "durian"}

mary | jay
```

```
{'apple', 'durian', 'orange'}
```

```
mary & jay
```

```
{'orange'}
```

```
mary - jay
```

```
{'apple'}
```

🎯 Recap Data Structures

1. list
2. tuple
3. dictionary

4. set

```
Start coding or generate with AI.
```

✅ Function

User defined function

```
## the most important thing why we write function
## because they are reusable
def hello():
    print("Hello World!")
```

```
hello()
```

```
Hello World!
```

```
## default argument
def hello2(name="may"):
    print("Hello " + name)

hello2("jay")
```

```
Hello may
```

```
## can we get input from a user?
def greeting():
    username = input("What's your name: ")
    result = f"Hi {username}"
    print(result)
    action = input("What are you goin to do today? ")
    print(f"You're going to {action}. Great!")
```

```
greeting()
```

```
What's your name: Kevin
Hi Kevin
What are you goin to do today? having lunch
You're going to having lunch. Great!
```

```
user_age = int(input("How old are you:  "))
print(user_age, type(user_age))
```

```
How old are you:  25
25 <class 'int'>
```

function can have more than one paremeters

```
## function can have more than one paremeters
def my_power(base=2, power=3):
    return base ** power
```

```
result = my_power(5, 2)
print(result)
```

```
25
```

```
## regular function
# def double(num):
#     return num*2

## lambda function
double = lambda num: num*2

double(3)
```

```
6
```

```
hello = lambda name: f"Hello {name}"
hello("jay")
```

```
'Hello jay'
```

🎯 Control Flow

1. if
2. for
3. while

```
def grading(score):
    if score >= 80:
        return "Passed"
    else:
        return "Failed"

grading(85)
```

```
'Passed'
```

```
def grading(score):
    """
    input: score is a numeric number
    output: grade passed or failed
    """
    if score >= 80:
        return "Passed"
    else:
        return "Failed"

grading(85)
```

```
## multiple if else
def full_grading(score):
    if score >= 80:
        return "A"
    elif score >= 70:
        return "B"
    elif score >= 60:
        return "C"
    elif score >= 50:
        return "D"
    else:
        return "Retry the exam again."

full_grading(45)
```

```
'Retry the exam again.'
```

```
## if multiple condition
# morning weekday => cereal
# morning weekend => hamburger
# else => fasting

time = "morning"
day = "weekend"

if time == "morning" and day == "weekday":
    print("I'm eating cereal")
elif time == "morning" and day == "weekend":
    print("I'm eating hamburger")
else:
    print("I'm eating nothing, I'm fasting")
```

```
I'm eating hamburger
```

```
## recap for
shopping_list = ["egg", "milk", "vitamilk", "bread"]
```

```
shopping_list = [ "egg" , "milk" , "vitamilk" , "bread" ]

for item in shopping_list:
    if len(item) >= 4:
        print(item)
```

```
milk
vitamilk
bread
```

```
## while loop
count = 0
while count < 5:
    print("hello world")
    count += 1
```

```
hello world
hello world
hello world
hello world
hello world
```