Hello World

Basic `Python` for Data Analyst (Beginner)

```
## code
```

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

[ ] ↳ *26 cells hidden*

Working with String

```
## fstring => format string
name = "mary"
gpa = 3.88

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

```
mary graduates from KU with gpa 3.88.
```

```
## python v3
print("hello world")
```

```
hello world
```

```
## 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")
```

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

```
4
```

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

```
result
```

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

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

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

```
## index starts with 0
text = "python"
text[3]
```

```
'h'
```

```
## slice text
text[1:]
```

```
'ython'
```

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

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

```
"hello " * 3
```

```
'hello hello hello '
```

```
"Python"[2:4]
```

```
'th'
```

```
"Python"[::-1]
```

```
'nohtyP'
```

```
"I am learning Python today"[0:15:2]
```

```
'Ta erigP'
```

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

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

```
Cython
```

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

Data Structures

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

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

```
shopping_list[0] = "egg egg"
```

```
shopping_list[2] = "lactasoy"
shopping_list
```

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

```
## list method
shopping_list.append("butter")
len(shopping_list)
```

```
5
```

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

```
shopping_list.append("milk")
shopping_list.append("milk")
```

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

```
2
```

Start coding or generate with AI.

```
# create a new list
shopping = ["milk", "bread", "egg", "creamcheese"]
shopping
```

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

```
shopping.sort(reverse=False)
print(shopping)
```

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

```
shopping.append("strawberry")
shopping
```

```
['bread', 'creamcheese', 'egg', 'milk', 'strawberry']
```

```
shopping.insert(1, "orange") # inplace
shopping
```

```
['bread', 'orange', 'creamcheese', 'egg', 'milk', 'strawberry']
```

```
["orange", "orange", "egg"].count("egg")
```

```
1
```

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

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

```
"Python" + " R"
```

```
'Python R'
```

```
shopping[3:]
```

```
['egg', 'milk', 'strawberry']
```

```
## loop through shopping list
for item in shopping:
    if len(item) <= 4 :
        continue
    else:
        print("I need to buy " + item)
```

```
I need to buy bread
I need to buy orange
I need to buy creamcheese
I need to buy strawberry
```

```
## 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
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']
```

```
## 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[3][1]
```

```
'ni hao'
```

```
("toy", 42, 40, 50, 100, ["hello", "nihao"])
```

```
('toy', 42, 40, 50, 100, ['hello', 'nihao'])
```

```
## 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["fav_movies"][-1]
```

```
'Lion King'
```

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

list(customer_01.values())
```

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

```
for item in list(customer_01.items()):
    print(item)
```

```
('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 gpa key
del customer["gpa"]

## use method
customer.pop("city")

customer
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-267-58324edf80e7> in <cell line: 2>()
      1 ## remove gpa key
----> 2 del customer["gpa"]
      3
      4 ## use method
      5 customer.pop("city")

KeyError: 'gpa'
```

Next steps: Explain error

```
# update value
customer["name"] = "David Beckham"
customer
```

```
{'name': 'David Beckham',
 'age': 50,
 'fav_movies': ['Superman', 'Inside Out', 'Lion King'],
 'nationality': 'American'}
```

```
## the last data structure: set
## set is 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"}
toy = {"apple", "durian"}

mary - toy
```

```
{'orange'}
```

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

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 functions
## because they are reusable
def hello():
    print("hello world")
```

```
## default argument
def hello2(name="toy"):
    print("hello " + name)

hello2("jessica")
```

```
hello jessica
```

```
## can we get input from a user?
def greeting():
    username = input("What's your name: ")
    result = f"Hello {username}!"
    print(result)

    action = input("What are you going to do today?")
    print(f"You're going to {action}. Great!")
```

```
greeting()
```

```
What's your name: david
Hello david!
What are you going to do today?running
You're going to running. Great!
```

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

```
How old are you: 33
33 <class 'int'>
```

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

```
result = my_power(power=2, base=9) ## 9**2
print(result)
```

```
81
```

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

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

double(52)

hello = lambda name: "hello " + name
hello(name="toy")
```

```
'hello toy'
```

Control Flow

1. if
2. for
3. while

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

```

grading(75)
```

```
'Failed'
```

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

full_grading(90)
```

```
'A'
```

```
# if muliple conditions
# morning weekday => cereal
# morning weekend => hamburger
# else => fasting

time = "lunch"
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("No eating, i'm fasting.")
```

```
No eating, i'm fasting.
```

```
## recap for
for item in ["orange", "apple", "grape", "banana"]:
    if len(item) > 5:
        print(item)
```

```
orange
banana
```

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

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

```
## homework
## 1. review methods (list, string)
## 2. create this function: pao ying chub

from random import choice

def pao_ying_chub():
    print("let's play game!")
    hands = ["hammer", "scissor", "paper"]
```

```
    hands = ["hammer", "scissor", "paper"]
    a_hand = choice(hands)
    print("Please choose your hand: [hammer, scissor, paper]")
    user_hand = input("Your hand: ")
    print(a_hand, user_hand)

pao_ying_chub()
```

```
let's play game!
Please choose your hand: [hammer, scissor, paper]
Your hand: paper
hammer paper
```

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