

IN IT TO WIN IT

SIVE BOBI

NTOKOZO PAUL

NASIPHI VUNDLE

AYANDA MTYEKWANA

NDUVHO TSHISHIVHIRI



python

```
from watson.framework import *
from watson.http.messages import *
from watson.common.imports import *
from watson.common.contextmanager import *

ACCEPTABLE_RETURN_TYPES = (str, int, float)

class Base(ContainerAware, metaclass=abc.ABCMeta):
    """The base class for all clients.

    Attributes:
        ``__action__`` (string): The last action performed by the client.

    """

    def execute(self, **kwargs):
        method = self.get_execute()
        self.__action__ = method
        return method(**kwargs)

    @abc.abstractmethod
    def get_execute(self):
        raise NotImplementedError("Subclasses must implement this method")
```

CONTENTS

- INTRODUCTION TO PYTHON
- VARIABLES
- DATA TYPES
- OPERATORS



PYTHON

**Python is a general purpose
programming language that can be used
on any modern computer operating
system**



VARIABLES

```
print("*****")
print("Welcome to BOB")
print("*****")

while True:
    choice = input("Would you like to make a transaction? (Enter the corresponding number)\n1. Yes\n2.

    if choice == "2":
        break

    if choice == "1":
        transaction_type = input("Would you like to make a deposit or withdrawal? (Enter the correspon

        if transaction_type == "1":
            amount = input("How much would you like to deposit? ")
            make_deposit(amount)
        elif transaction_type == "2":
            amount = input("How much would you like to withdraw? ")
            make_withdrawal(amount)
        else:
            print("You provided an invalid input.")
            continue

        current_balance = get_balance()
        print(f"Current balance: {current_balance}")
```

DATA TYPES

Data types are used to define the type of a variable.

```
    except ValueError:  
        print("You provided an invalid input.")  
        return
```

```
29 |         amount = float(amount)  
30 |         if amount <= 0:  
31 |             raise ValueError
```

```
6  |     def update_balance(new_balance):  
7  |         with open("Bank Data.txt", "w") as file:  
8  |             file.write(str(new_balance))  
9  |
```

OPERATORS

The screenshot shows a Python code editor with a dark theme. The code is part of a banking application, specifically handling deposits and withdrawals.

```
10     1 usage
11     def make_deposit(amount):
12         try:
13             amount = float(amount)
14             if amount <= 0:
15                 raise ValueError
16         except ValueError:
17             print("You provided an invalid input.")
18             return
19
20             current_balance = get_balance()
21             new_balance = current_balance + amount
22             update_balance(new_balance)
23             log_transaction("Deposit", amount)
24             print("Deposit successful.")
25
26     1 usage
27     def make_withdrawal(amount):
28         try:
29             amount = float(amount)
30             if amount <= 0:
31                 raise ValueError
32
33             current_balance = get_balance()
34             new_balance = current_balance - amount
35             update_balance(new_balance)
36             log_transaction("Withdrawal", amount)
37             print("Withdrawal successful.")
38
39             while True:
```

The code uses several operators:

- Assignment operator (=) to assign values to variables like `amount` and `new_balance`.
- Comparison operators (==, <=) to check if `amount` is valid or if it's less than or equal to zero.
- Arithmetic operators (+, -) to calculate new balances.
- Logical operators (if, try, except) to handle errors and control flow.

Annotations with yellow circles highlight the assignment operator (=) in line 13 and the arithmetic operator (+) in line 21.

THANK YOU