LAPORAN UJIAN AKHIR SEMESTER

PEMROGRAMAN BERORIENTASI OBJEK

DISUSUN OLEH :

Kelompok 10:

1. Imelda Cyntia (G1A022022)
2. Resyaliana Esa Putri (G1A022038)
3. Anissa Shanniyah Aprilia (G1A022044)

Dosen Pengampu :

1. Arie Vatresia, S.T,.M.TI, Ph.D.
2. Mochammad Yusa, S. Kom., M. Kom.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2023

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunianya, sehingga kami dapat menyelesaikan Laporan Pelaksanaan Ujian Akhir Semester (UAS) mata kuliah Pemprograman Berorientasi Objek (PBO).

Laporan ini merupakan salah satu matakuliah yang wajib ditempuh di Prodi Informatika Fakultas Teknik Universitas Bengkulu. Laporan ini disusun sebagai nilai akhir mata kuliah PBO yang telah dilaksanakan selama satu semester (enam bulan).

Dengan selesainya laporan ini tidak terlepas dari bantuan banyak pihak yang telah memberikan masukan-masukan kepada kami. Untuk itu kami mengucapkan banyak terimakasih.

Kami menyadari bahwa penyusunan laporan ini masih jauh dari sempurna. Mengingat keterbatasan waktu dan kemampuan kami. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang bersifat membangun, demi kesempurnaan laporan ini. Akan tetapi, kami berharap semoga laporan ini dapat bermanfaat, khususnya bagi kami.

Bengkulu, 6 Juni 2023

Kelompok 10

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI ii
DAFTAR GAMBAR iii
BAB I 1
PENDAHULUAN 1
- **A. Latar Belakang** 1
- **B. Rumusan Masalah** 1
- **C. Tujuan Dan Manfaat** 1

BAB II 2
LANDASAN TEORI 2
BAB III 4
PEMBAHASAN 4
- **A. Algoritma** 4
- **B. Source Code** 4
- **C. Analisis Program** 10

BAB IV 17
KESIMPULAN DAN SARAN 17
- **A. Kesimpulan** 17
- **B. Saran** 17

DAFTAR PUSTAKA 18

DAFTAR GAMBAR

Gambar 3.1 *Source Code*……5
Gambar 3.1 *Source Code*……6
Gambar 3.1 *Source Code*……6
Gambar 3.2 *Output* Jika Menang……13
Gambar 3.3 *Output* Jika Menang dan Ada Kesalahan……..14
Gambar 3.4 *Output* Jika Kalah……..14
Gambar 3.4 *Output* Jika Kalah……..15

BAB I

PENDAHULUAN

A. Latar Belakang

Permainan Hangman adalah salah satu permainan tebak kata yang populer di seluruh dunia. Permainan ini biasanya dimainkan oleh satu atau lebih pemain yang harus menebak kata yang tersembunyi dengan menebak satu huruf pada setiap giliran. Setiap kali pemain menebak huruf yang salah, gambar hangman digambar satu langkah lebih lengkap. Pemain harus menebak kata sebelum gambar hangman lengkap digambar untuk memenangkan permainan.

B. Rumusan Masalah

1. Bagaimana cara membuat permainan Hangman dengan menggunakan bahasa pemrograman Python?
2. Bagaimana cara menyediakan kata-kata yang dapat ditebak secara acak dalam permainan Hangman?
3. Bagaimana cara menampilkan visualisasi gambar hangman yang berbeda pada setiap kesalahan tebakan huruf?

C. Tujuan Dan Manfaat

Tujuan:

1. Membuat permainan Hangman yang interaktif dan menarik menggunakan bahasa pemrograman Python.
2. Menyediakan kata-kata acak untuk ditebak dalam permainan Hangman agar permainan menjadi lebih menantang dan beragam.
3. Menampilkan visualisasi gambar hangman yang terus berubah setiap kali pemain menebak huruf yang salah.
4. Menyediakan logika permainan yang memungkinkan pemain menebak kata yang tersembunyi dan menentukan kapan permainan berakhir dengan kekalahan atau kemenangan.

Manfaat:

1. Mengembangkan keterampilan pemrograman Python dalam mengimplementasikan logika permainan.
2. Meningkatkan pemahaman tentang algoritma pemrograman, string handling, dan operasi dasar dalam Python.

BAB II

LANDASAN TEORI

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Sebagai bahasa pemrograman tinggi, python dapat dipelajari dengan mudah karena telah dilengkapi dengan manajemen memori otomatis. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python dirancang untuk memberikan kemudahan bagi programmer melalui segi efisiensi waktu, kemudahan dalam pengembangan dan kompatibilitas dengan sistem. Python bisa digunakan untuk membuat aplikasi standalone (berdiri sendiri) dan pemrograman script (*scripting programming*).

Bahasa pemrograman Python dibuat oleh Guido Van Rossum diawal tahun 1990. Guio Van Rossum merupakan seorang programmer asal Belanda. Python pertama kali dibuat di Centrum Wiskunde Informatica (CWI) di Negara Belanda. Python bukanlah nama ular, Guido menamai bahasa pemrograman buatannya Python karena dia adalah fans dari grup komedi Monty Python Flying Circus yang berasal dari Inggris. Pembuatan Python dilanjutkan pada tahun 1995 di Corporation for National Research Iniative (CNRI) yang terletak di Virginia Amerika. Guido dan tim python kemudian pindah ke BeOpen.com pada bulan Mei 2000, mereka membentuk tim BeOpen PythonLabs. Kemudian tim Python pindah ke Digital Creation (sekarang Zope Corporation) pada bulan Oktober 2000. Organisasi Python Software Foundation (PSF) dibentuk pada tahun 2001. Tentunya organisasi ini memiliki misi yang luar biasa untuk bahasa pemrograman Python. Misinya adalah melindungi, mempromosikan, melindungi, membuat kemajuan pada bahasa pemrograman Python, semendukung serta memfasilitasi pertumbuhan komunitas developer Python internasional.

Bahasa pemprograman Python memiliki kelebihan dan kekurangan. Kelebihan Bahasa pemprograman python yaitu mudah digunakan karena Bahasa python menyerupai Bahasa manusia dari pada Bahasa mesin, Python mempunyai Kompabilitas dan Kemampuan, Mendukung OOP, *Platform Independent* Maksud

dari *platform independent* adalah program yang di buat bisa dijalankan di sistem operasi apa saja selama di sistem operasi tersebut terdapat *platform* Python. Adapun kekurangannya yaitu Beberapa penugasan terdapat di luar dari jangkauan kemampuan Python, seperti bahasa pemrograman dinamis lainnya, Python tidak secepat atau efisien sebagai statis. Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) adalah paradigma pemrograman yang penting dalam pengembangan perangkat lunak modern. Python merupakan bahasa pemrograman yang kuat dan mendukung penuh konsep-konsep OOP. Dalam Python, segala sesuatu dianggap sebagai objek. Objek adalah representasi nyata dari konsep di dunia nyata. Objek memiliki atribut (data) yang menyimpan informasi tentang objek tersebut, serta metode (fungsi) yang mendefinisikan perilaku objek.

Pada dasarnya, pemrograman berorientasi objek di Python melibatkan pembuatan kelas (*class*) sebagai blueprint atau cetak biru untuk menciptakan objek. Kelas merupakan struktur data yang menggabungkan data dan fungsi terkait dalam satu entitas. Objek kemudian dibuat berdasarkan kelas dengan menggunakan proses yang disebut instansiasi. Dalam pemrograman berorientasi objek, konsep-konsep seperti enkapsulasi, pewarisan, dan polimorfisme digunakan. Enkapsulasi adalah konsep yang memungkinkan penyembunyian detail internal objek dan hanya mengekspos metode dan atribut yang diperlukan. Pewarisan memungkinkan pembuatan kelas baru yang mewarisi atribut dan metode dari kelas yang sudah ada, memungkinkan penggunaan kembali dan perluasan kode. Polimorfisme memungkinkan objek dari kelas yang berbeda untuk merespons metode dengan cara yang berbeda, tetapi dengan menggunakan antarmuka yang sama.

Dalam Python, pewarisan dilakukan dengan menunjukkan kelas dasar (superclass) saat mendefinisikan kelas turunan (subclass) baru. Kelas turunan dapat menambahkan atribut dan metode tambahan, serta mengganti atau memperluas perilaku yang diwarisi dari kelas dasar. Python juga mendukung konsep abstraksi, di mana kelas abstrak dapat menyediakan kerangka kerja umum untuk kelas turunannya. Pemrograman berorientasi objek dalam Python memungkinkan pembuatan kode yang lebih terstruktur, modular, dan dapat diorganisir dengan baik. Dengan menggunakan konsep-konsep OOP, kita dapat memecah kompleksitas program.

BAB III

PEMBAHASAN

A. Algoritma

Algoritma yang digunakan untuk memainkan permainan Hangman ini, pertama-tama kode akan memilih secara acak sebuah kata dari kamus kata yang telah ditentukan sebelumnya. Kata terpilih akan ditampilkan dalam bentuk garis-garis bawah, dengan setiap hurufnya digantikan oleh underscore. Jumlah underscore yang ditampilkan akan sesuai dengan panjang kata yang dipilih.

Setelah itu, pemain diminta untuk menebak satu huruf. Jika huruf tersebut sudah pernah ditebak sebelumnya, pesan "Letters Already Guessed Earlier." akan ditampilkan. Jika huruf yang ditebak benar, huruf tersebut akan ditambahkan ke daftar huruf yang sudah ditebak. Kata terpilih yang ditampilkan akan diperbarui, dengan huruf yang ditebak menggantikan underscore pada posisi yang sesuai. Jumlah huruf yang ditebak dengan benar akan ditampilkan.

Namun, jika huruf yang ditebak salah, jumlah kesalahan akan bertambah. Gambar Hangman yang ditampilkan juga akan diubah sesuai dengan jumlah kesalahan yang terjadi. Selain itu, kata terpilih yang ditampilkan juga akan diperbarui dengan huruf yang sudah ditebak. Proses ini akan terus berlanjut hingga salah satu dari dua kondisi terpenuhi: jumlah kesalahan mencapai 6 (Hangman lengkap) atau semua huruf telah ditebak dengan benar.

Jika semua huruf telah ditebak dengan benar, pesan "Congrats! U WINNN!" akan ditampilkan sebagai pemenang. Namun, jika jumlah kesalahan mencapai 6 sebelum semua huruf ditebak dengan benar, pesan "Game is over! YOU LOSE:(" akan ditampilkan sebagai kekalahan.

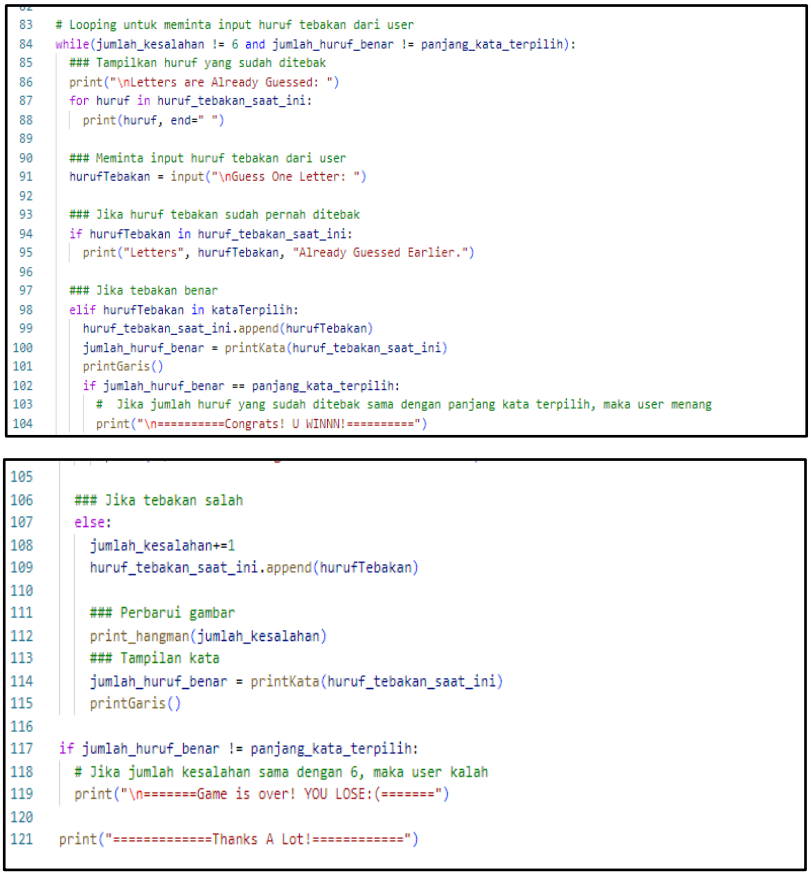
Setelah permainan selesai, pesan "Thanks A Lot!" akan ditampilkan sebagai penutup dari permainan Hangman. Perlu dicatat bahwa urutan pesan dan gambar Hangman yang ditampilkan dapat berbeda dalam setiap permainan, tergantung pada tebakan huruf yang dimasukkan oleh pemain.

B. Source Code

Printscreen Source Code:

```python
import random

# Tampilan header game
print("WELCOME TO GAME HANGMAN")
print("==================================")
print("==================================")

# Daftar kata yang akan digunakan dalam game
kamusKata = ["sunflower", "table", "chair", "glass", "house", "diamond", "jokes","yeet","hello", "rose", "howdy", "grape", "apple", "hoax", "shoes"]

# Pilih kata secara acak dari kamus kata
kataTerpilih = random.choice(kamusKata)

# Tampilan awal kata dengan garis bawah
for x in kataTerpilih:
  print("_", end=" ")

# Fungsi untuk menampilkan gambar hangman
def print_hangman(salah):
  if(salah == 1):
    print("\n+---+")
    print("O   |")
    print("    |")
    print("    |")
    print("   ===")
  elif(salah == 2):
    print("\n+---+")
    print("O   |")
    print("|   |")
    print("    |")
    print("   ===")
  elif(salah == 3):
    print("\n+---+")
    print(" O  |")
    print("/|  |")
    print("    |")
    print("   ===")
  elif(salah == 4):
    print("\n+---+")
    print(" O  |")
    print("/|\ |")
    print("    |")
    print("   ===")
  elif(salah == 5):
    print("\n+---+")
    print(" O  |")
    print("/|\ |")
    print("/   |")
    print("   ===")
```

```python
  elif(salah == 6):
    print("\n+---+")
    print(" O   |")
    print("/|\  |")
    print("/ \  |")
    print("    ===")

# Fungsi untuk menampilkan kata yang sudah ditebak
def printKata(hurufTebakan):
  counter=0
  jumlahBenar=0
  for char in kataTerpilih:
    if(char in hurufTebakan):
      print(kataTerpilih[counter], end=" ")
      jumlahBenar+=1
    else:
      print("_", end=" ")
    counter+=1
  return jumlahBenar

# Fungsi untuk menampilkan garis bawah kata yang belum ditebak
def printGaris():
  print("\r")
  for char in kataTerpilih:
    print("\u203E", end=" ")

# Inisialisasi variabel
panjang_kata_terpilih = len(kataTerpilih)
jumlah_kesalahan = 0
posisi_tebakan_sekarang = 0
huruf_tebakan_saat_ini = []
jumlah_huruf_benar = 0

```



```
# Looping untuk meminta input huruf tebakan dari user
while(jumlah_kesalahan != 6 and jumlah_huruf_benar != panjang_kata_terpilih):
  ### Tampilkan huruf yang sudah ditebak
  print("\nLetters are Already Guessed: ")
  for huruf in huruf_tebakan_saat_ini:
    print(huruf, end=" ")

  ### Meminta input huruf tebakan dari user
  hurufTebakan = input("\nGuess One Letter: ")

  ### Jika huruf tebakan sudah pernah ditebak
  if hurufTebakan in huruf_tebakan_saat_ini:
    print("Letters", hurufTebakan, "Already Guessed Earlier.")

  ### Jika tebakan benar
  elif hurufTebakan in kataTerpilih:
    huruf_tebakan_saat_ini.append(hurufTebakan)
    jumlah_huruf_benar = printKata(huruf_tebakan_saat_ini)
    printGaris()
    if jumlah_huruf_benar == panjang_kata_terpilih:
      #  Jika jumlah huruf yang sudah ditebak sama dengan panjang kata terpilih, maka user menang
      print("\n==========Congrats! U WINNN!==========")

  ### Jika tebakan salah
  else:
    jumlah_kesalahan+=1
    huruf_tebakan_saat_ini.append(hurufTebakan)

    ### Perbarui gambar
    print_hangman(jumlah_kesalahan)
    ### Tampilan kata
    jumlah_huruf_benar = printKata(huruf_tebakan_saat_ini)
    printGaris()

if jumlah_huruf_benar != panjang_kata_terpilih:
  # Jika jumlah kesalahan sama dengan 6, maka user kalah
  print("\n=======Game is over! YOU LOSE:(=======")

print("============Thanks A Lot!============")
```

Gambar 3.1 *Source Code*

Source Code:

```
import random

# Tampilan header game
print("WELCOME TO GAME HANGMAN")
print("=====================================")
print("=====================================")

# Daftar kata yang akan digunakan dalam game
kamusKata = ["sunflower", "table", "chair", "glass", "house", "diamond",
"jokes","yeet","hello", "rose", "howdy", "grape", "apple", "hoax", "shoes"]

# Pilih kata secara acak dari kamus kata
kataTerpilih = random.choice(kamusKata)
```

```
# Tampilan awal kata dengan garis bawah
for x in kataTerpilih:
  print("_", end=" ")

# Fungsi untuk menampilkan gambar hangman
def print_hangman(salah):
  if(salah == 1):
    print("\n+---+")
    print("O   |")
    print("    |")
    print("    |")
    print("   ===")
  elif(salah == 2):
    print("\n+---+")
    print("O   |")
    print("|   |")
    print("    |")
    print("   ===")
  elif(salah == 3):
    print("\n+---+")
    print(" O  |")
    print("/|  |")
    print("    |")
    print("   ===")
  elif(salah == 4):
    print("\n+---+")
    print(" O  |")
    print("/|\ |")
    print("    |")
    print("   ===")
  elif(salah == 5):
```

```
    print("\n+---+")
    print(" O  |")
    print("/|\ |")
    print("/   |")
    print("   ===")
  elif(salah == 6):
    print("\n+---+")
    print(" O   |")
    print("/|\  |")
    print("/ \  |")
    print("    ===")

# Fungsi untuk menampilkan kata yang sudah ditebak
def printKata(hurufTebakan):
  counter=0
  jumlahBenar=0
  for char in kataTerpilih:
    if(char in hurufTebakan):
      print(kataTerpilih[counter], end=" ")
      jumlahBenar+=1
    else:
      print("_", end=" ")
    counter+=1
  return jumlahBenar

# Fungsi untuk menampilkan garis bawah kata yang belum ditebak
def printGaris():
  print("\r")
  for char in kataTerpilih:
    print("\u203E", end=" ")

# Inisialisasi variabel
```

```
panjang_kata_terpilih = len(kataTerpilih)
jumlah_kesalahan = 0
posisi_tebakan_sekarang = 0
huruf_tebakan_saat_ini = []
jumlah_huruf_benar = 0

# Looping untuk meminta input huruf tebakan dari user
while(jumlah_kesalahan != 6 and jumlah_huruf_benar != panjang_kata_terpilih):
  ### Tampilkan huruf yang sudah ditebak
  print("\nLetters are Already Guessed: ")
  for huruf in huruf_tebakan_saat_ini:
    print(huruf, end=" ")

  ### Meminta input huruf tebakan dari user
  hurufTebakan = input("\nGuess One Letter: ")

  ### Jika huruf tebakan sudah pernah ditebak
  if hurufTebakan in huruf_tebakan_saat_ini:
    print("Letters", hurufTebakan, "Already Guessed Earlier.")

  ### Jika tebakan benar
  elif hurufTebakan in kataTerpilih:
    huruf_tebakan_saat_ini.append(hurufTebakan)
    jumlah_huruf_benar = printKata(huruf_tebakan_saat_ini)
    printGaris()
    if jumlah_huruf_benar == panjang_kata_terpilih:
      #  Jika jumlah huruf yang sudah ditebak sama dengan panjang kata terpilih,
maka user menang
      print("\n==========Congrats! U WINNN!==========")

  ### Jika tebakan salah
  else:
```

```
    jumlah_kesalahan+=1
    huruf_tebakan_saat_ini.append(hurufTebakan)

    ### Perbarui gambar
    print_hangman(jumlah_kesalahan)
    ### Tampilan kata
    jumlah_huruf_benar = printKata(huruf_tebakan_saat_ini)
    printGaris()

if jumlah_huruf_benar != panjang_kata_terpilih:
  # Jika jumlah kesalahan sama dengan 6, maka user kalah
  print("\n=======Game is over! YOU LOSE:(=======")

print("=============Thanks A Lot!============")
```

C. Analisis Program

Penjelasan Source Code:

Program di atas adalah sebuah game sederhana yang disebut Hangman. Game ini akan mengacak sebuah kata dari daftar kata yang telah didefinisikan, dan user harus menebak huruf yang ada dalam kata tersebut. User hanya diberi kesempatan menebak sebanyak 6 kali, dan jika gagal menebak maka game akan berakhir dan pemain kalah. Pada program di atas merupakan implementasi sederhana dari game hangman menggunakan bahasa pemrograman Python. Pertama, terdapat modul random yang diimpor untuk memilih kata secara acak dari daftar kata yang disediakan. Selanjutnya, terdapat sebuah tampilan header game yang ditampilkan dengan print("WELCOME TO GAME HANGMAN"), print("=================================="), dan print("=================================="). Terdapat daftar kata yang akan digunakan pada game dalam kamusKata = ["sunflower", "table", "chair", "glass", "house", "diamond", "jokes","yeet","hello", "rose", "howdy", "grape", "apple", "hoax", "shoes"]. Setelah itu, sebuah kata dipilih secara acak dan disimpan ke dalam variabel kataTerpilih = random.choice(kamusKata).

Kemudian, garis-garis bawah (_) sebanyak huruf-huruf dalam kata tersebut dicetak sebagai petunjuk tebakan yang harus ditebak oleh user.

Terdapat beberapa fungsi dalam program ini, seperti print_hangman yang digunakan untuk mencetak gambar Hangman berdasarkan jumlah kesalahan yang telah dilakukan oleh pemain, apabila user menebak satu angka yang salah maka luaran yang dihasilkan yaitu berupa angka 0 yang melambangkan gambar kepala hangman if(salah == 1): print("\n+---+") print("O |") print(" |") print(" |") print(" ==="), kemudian jika salah dua huruf, maka luaran yang dihasilkan yaitu garis untuk membentuk tangan sebelah kiri hangman elif(salah == 2): print("\n+---+") print("O |") print("| |") print(" |") print(" ==="), jika salah tiga huruf, maka luaran yang dihasilkan yaitu garis yang membentuk badan hangman elif(salah == 3): print("\n+---+") print(" O |") print("/| |") print(" |") print(" ==="), jika salah empat huruf, maka luaran yang dihasilkan berupa garis yang membentuk tangan sebelah kanan hangman elif(salah == 4): print("\n+---+") print(" O |") print("/|\ |") print(" |") print(" ==="), jika salah lima huruf, maka luaran yang dihasilkan berupa garis yang membentuk kaki sebelah kiri hangman elif(salah == 5): print("\n+---+") print(" O |") print("/|\ |") print("/ |") print(" ==="), dan algoritma yang terakhir yaitu, jika salah enam huruf , maka luaran yang dihasilkan berupa garis yang membentuk kaki sebelah kanan hangman elif(salah == 6): print("\n+---+") print(" O |") print("/|\ |") print("/ \ |") print(" ==="). Setelah, user menebak angka yang salah sebanyak enam kali, maka permainan tersebut selesai dan tampilan yang dihasilkan berupa gambar hangman yang lengkap dan permainan menampilkan print("\n=======Game is over! YOU LOSE:(=======") print("============Thanks A Lot!============").

Pada fungsi printKata yang mencetak kata yang sedang ditebak, menggantikan huruf-huruf yang telah ditebak, dan menghitung jumlah huruf yang benar ditebak, jika huruf telah ditebak , hurugf tersebut akan ditampilkan, sedangkan huruf-huruf yang belum ditebak akan ditampikan sebagai garis. Selain itu, terdapat juga fungsi printGaris yang mencetak garis-garis horizontal sebagai indikator jumlah huruf dalam kata yang harus ditebak, pada print("\r")

digunakan untuk mencetak karakter *carriage return* (CR) yang berfungsi untuk mengembalikan kursor ke awal baris yang sama. Hal ini dilakukan agar garis-garis yang dicetak berada di baris yang sama dengan kata yang sedang ditebak, pada baris kedua fungsi, terdapat loop `for` yang akan mengiterasi setiap karakter (`char`) dalam kata yang telah terpilih (`kataTerpilih`). Di dalam loop, print("\u203E", end=" ") digunakan untuk mencetak karakter unicode \u203E, yang merupakan karakter garis mendatar (overline), dengan menggunakan *escape sequence* \u diikuti dengan kode unicode karakter. Dalam hal ini, karakter garis mendatar digunakan untuk menunjukkan posisi dari setiap huruf yang telah ditebak, Argument end=" " pada print("\u203E", end=" ") menentukan bahwa setiap karakter yang dicetak akan diakhiri dengan satu spasi. Hal ini dilakukan agar setiap karakter garis mendatar tercetak dengan jarak yang sama dan sejajar dengan huruf-huruf dalam kata yang sedang ditebak.

Pada kode program juga terdapat variable-variabel yang digunakan untuk menginisialisasikan permainan, # Inisialisasi variable, panjang_kata_terpilih = len(kataTerpilih) variable ini digunakan untuk menentukan seberapa Panjang huruf pada kata yang terpilih secara random oleh program, jumlah_kesalahan = 0 variabel ini diguanakan untuk menghitung berapa banyak kesalahan huruf yang terdapat dalam permainan, posisi_tebakan_sekarang = 0 variabel ini digunakan untuk menampilkan posisi huruf yang telah ditebak, huruf_tebakan_saat_ini = [] variable ini digunakan untuk menampilkan huruf-huruf yang sudah tertebak oleh pemain, jumlah_huruf_benar = 0 variabel ini digunakan untuk melihat seberapa banyak jumlah huruf yang sudah tertebak.

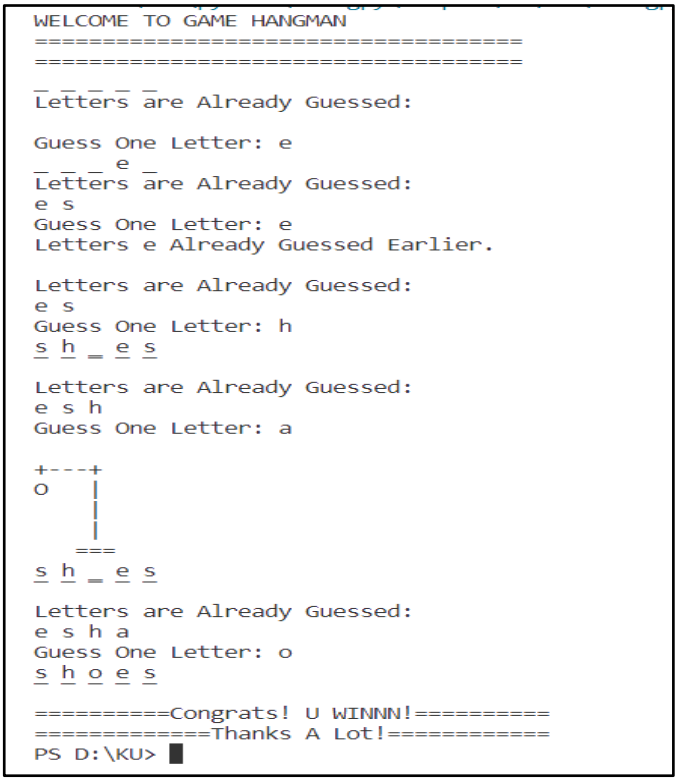
Dengan demikian, fungsi printGaris() akan mencetak garis mendatar di bawah kata yang sedang ditebak, menunjukkan posisi dari setiap huruf yang telah benar ditebak. Program kemudian melakukan perulangan while (jumlah_kesalahan != 6 and jumlah_huruf_benar != panjang_kata_terpilih): perulangan ini merupakan perulangan utama game hangman ini, perulangan ini akan terus berjalan hingga jumlah kesalahan mencapai batas maksimum, yaitu enam atau semua huruf dalam kata telah benar ditebak. hurufTebakan = input("\nGuess One Letter: ") Pemain diminta untuk menebak satu huruf setiap

kali perulangan berlangsung. if hurufTebakan in huruf_tebakan_saat_ini: print("Letters", hurufTebakan, "Already Guessed Earlier.") Jika huruf yang ditebak sudah pernah ditebak sebelumnya, pesan akan dicetak. Jika huruf tersebut benar, huruf tersebut ditambahkan ke dalam list `huruf_tebakan_saat_ini`, kata yang sedang ditebak dicetak ulang, dan jika semua huruf telah benar ditebak, pemain memenangkan permainan if jumlah_huruf_benar == panjang_kata_terpilih: print("\n==========Congrats! U WINNN!==========). Jika huruf yang ditebak salah, jumlah kesalahan akan ditambah, print_hangman(jumlah_kesalahan) gambar Hangman akan diperbarui, kata yang sedang ditebak dicetak ulang, dan if jumlah_huruf_benar != panjang_kata_terpilih: print("\n=======Game is over! YOU LOSE:(=======") print("=============Thanks A Lot!============") jika jumlah kesalahan mencapai batas maksimum sebelum semua huruf benar ditebak, pemain kalah. Terakhir, pesan terima kasih kepada pemain dicetak.

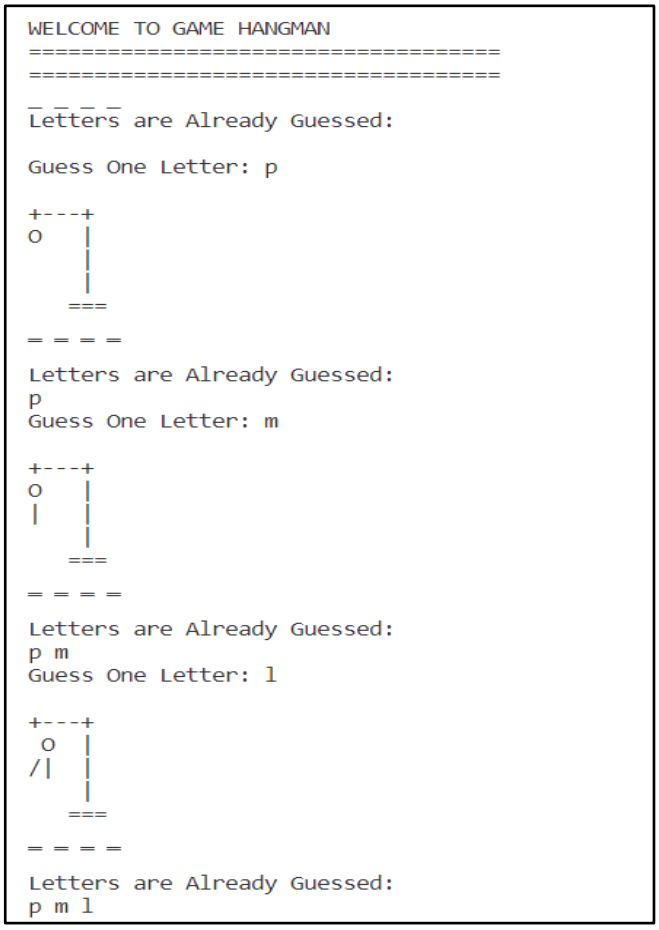
Printscreen Output:

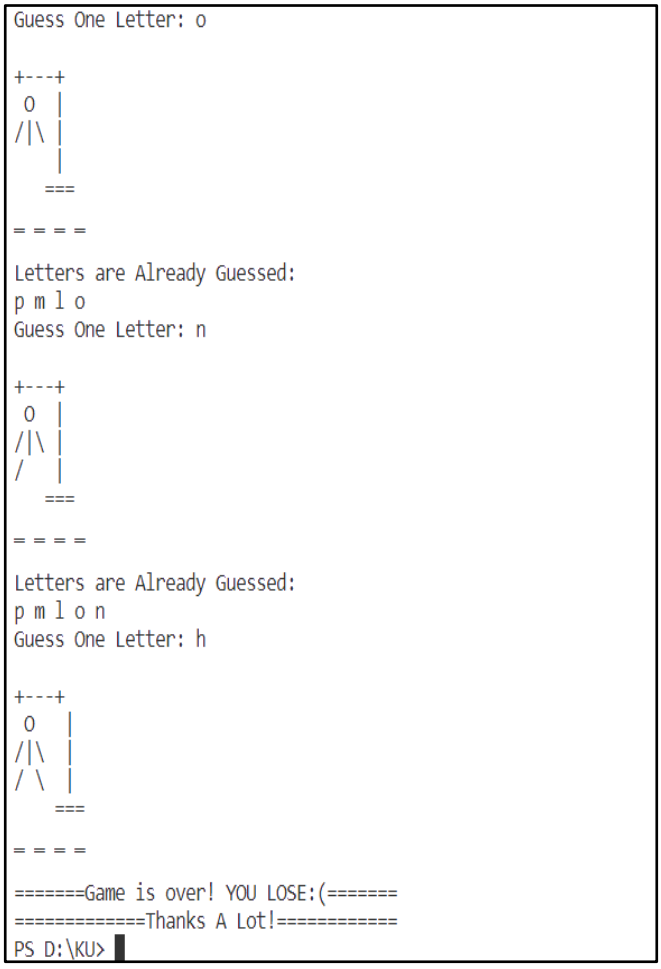
```
WELCOME TO GAME HANGMAN
===================================
===================================

_ _ _ _
Letters are Already Guessed:

Guess One Letter: a
_ _ a _

Letters are Already Guessed:
a
Guess One Letter: h
h _ a _

Letters are Already Guessed:
a h
Guess One Letter: o
h o a _

Letters are Already Guessed:
a h o
Guess One Letter: x
h o a x

==========Congrats! U WINNN!==========
=============Thanks A Lot!============
PS D:\KU>
```

Gambar 3.2 *Output* Jika Menang



Gambar 3.3 *Output* Jika Menang dan Ada Kesalahan





Gambar 3.4 *Output* Jika Kalah

Penjelasan Output:

Setelah menjalankan kode pada Gambar 1, output yang dihasilkan akan memulai dengan pesan "WELCOME TO GAME HANGMAN" yang mencetak judul game. Kemudian, dua baris garis horizontal akan ditampilkan sebagai pemisah sebelum game hangman dimulai. Selanjutnya, kode akan memilih secara acak sebuah kata dari kamus kata yang telah ditentukan sebelumnya. Kata terpilih akan ditampilkan dalam bentuk garis-garis bawah, dengan setiap hurufnya digantikan oleh underscore. Jumlah underscore yang ditampilkan akan sesuai dengan panjang kata yang dipilih.

Seperti pada Gambar 2 terdapat 4 underscore yang berarti kata yang harus ditebak terdapat 4 huruf. Setelah itu, user diminta untuk menebak satu huruf. Jika huruf yang ditebak benar, huruf tersebut akan ditambahkan ke daftar huruf yang

sudah ditebak, huruf tersebut diletakkan sesuai dengan posisinya di atas underscore. Kata terpilih yang ditampilkan akan diperbarui, dengan huruf yang ditebak menggantikan underscore pada posisi yang sesuai. Jumlah huruf yang ditebak dengan benar akan ditampilkan. Jika semua huruf telah ditebak dengan benar dan sesuai dengan kata yang sesuai dengan kamus program, program akan menampilkan pesan "Congrats! U WINNN!" yang menandakan user sebagai pemenang dan permainan telah selesai.

Pada Gambar 3, merupakan contoh jika huruf yang ditebak oleh user tersebut sudah pernah ditebak sebelumnya, pesan "Letters Already Guessed Earlier." akan ditampilkan untuk mengingatkan user bahwa huruf tersebut telah ditebak sebelumnya. Namun, jika huruf yang ditebak salah, jumlah kesalahan akan bertambah. Sehingga gambar Hangman yang ditampilkan juga akan diubah sesuai dengan jumlah kesalahan yang terjadi, contohnya seperti pada gambar 3 user salah dalam menebak huruf sehingga luaran yang dihasilkan berupa kepala hangman. Proses ini akan berlangsung terus menerus. Selain itu, kata terpilih yang ditampilkan juga akan diperbarui dengan huruf yang sudah ditebak. Proses ini akan terus berlanjut hingga salah satu dari dua kondisi terpenuhi, yaitu kalah jika huruf yang salah ditebak mencapai batas maksimum yang telah ditentukan oleh program yaitu enam atau permainan akan berhenti karena user telah berhasil menebak huruf sesuai dengan kata yang telah dipilih secara random oleh program.

Pada Gambar 4 terdapat 4 underscore yang ditampilkan oleh program untuk ditebak oleh user, user diberikan kesempatan sebanyak 6 kali untuk menebak huruf yang benar untuk menyelesaikan permainan ini. Pada kesempatan pertama user memasukkan huruf yang tidak ada pada kata yang telah terpilih sehingga luaran yang dihasilkan berupa kepala hangman, pada kesempatan kedua user juga melakukan kesalahan sehingga program menampilkan luaran berupa garis yang membentuk tangan bagian kiri hangman, pemainan berlanjut hingga kesempatan user telah habis atau telah sampai batas maksimum dari 4 huruf yang harus ditebak user memasukkan huruf yang salah sebanyak 6 kali. Sehingga, gambar hangman akan ditampilkan. Pesan "Game is over! YOU LOSE:(" ditampilkan. Setelah permainan selesai, pesan "Thanks A Lot!" akan ditampilkan sebagai penutup dari permainan Hangman.

BAB IV

KESIMPULAN DAN SARAN

A. Kesimpulan

Python adalah bahasa pemrograman interpretatif multiguna yang dirancang dengan fokus pada keterbacaan kode. Bahasa ini memiliki sintaksis yang jelas dan mudah dipahami, serta dilengkapi dengan pustaka standar yang luas. Python dapat digunakan untuk membuat berbagai jenis aplikasi, baik itu standalone maupun scripting. Python menggunakan manajemen memori otomatis, sehingga memudahkan programmer dalam pengembangan dan mengurangi risiko kesalahan yang terkait dengan pengelolaan memori.

Pada Python mendukung paradigma pemrograman berorientasi objek (OOP), di mana segala sesuatu dianggap sebagai objek yang memiliki atribut dan metode. OOP memungkinkan pembuatan struktur data yang kompleks dan pengorganisasian kode yang lebih terstruktur, modular, dan dapat diorganisir dengan baik. Secara keseluruhan, Python adalah bahasa pemrograman yang populer, mudah dipelajari, dan memiliki banyak kegunaan dalam teknologi saat ini. Dengan kemampuan OOP, sintaksis yang jelas, dan dukungan komunitas yang kuat, Python menjadi pilihan yang menarik untuk pengembangan perangkat lunak.

B. Saran

Pada laporan Ujian Akhir Semester ini yang membahas tentang pembuatan program game hangman. Agar program game hangman ini berjalan dengan baik, diperlukan desain yang matang dan implementasi yang tepat. Pastikan untuk memperhatikan setiap *clue* yang diberikan sehingga dapat menebak kata dengan benar. Untuk meningkatkan kinerja pada program, pertimbangkan untuk mengimplementasikan fitur-fitur baru agar memperindah tampilan game sehingga *user* lebih tertarik dengan game ini. Dengan menerapkan saran-saran di atas, kode program game Hangman dapat ditingkatkan untuk memberikan pengalaman permainan yang lebih baik, menarik, dan menghibur bagi *user*.

DAFTAR PUSTAKA

Josikie. (2021, Juli 26). Sejarah Lahirnya Bahasa Pemprograman Python. Retrieved 2 Juni 2023, from https://josikie.com/sejarah-lahirnya-bahasa-pemrograman-python/

Saragih, R. R. (2016). Pemprograman dan Bahasa Pemprograman.Retrieved 2 Juni 2023, from https://www.researchgate.net/publication/329885312_PEMROGRAMAN_DAN_BAHASA_PEMROGRAMAN

Yuniar,M. (2022, Januari 20). Bahasa Pemprograman Python : 6 Kelebihan Dan Kekurangannya. Retrieved 2 Juni 2023, from https://www.ekrut.com/media/4-kelebihan-bahasa-pemrograman-python